Classification of Synthesised ISAR Images of Small Complex Targets

Cullen Stewart-Burger

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electronic) in the Faculty of Engineering at Stellenbosch University.

Supervisor: Dr D. J. Ludick
Department of Electrical and Electronic Engineering
Co-supervisor: Dr M. Potgieter
Radar and Electronic Warfare, Defence and Security, CSIR

March 2023

Acknowledgements

I would firstly like to thank my supervisor, Dr D.J. Ludick, and co-supervisor, Dr M. Potgieter, for their continued help and guidance throughout this project. I appreciate you making time in your busy schedules to assist and encourage me. Additionally, I would like to express my gratitude towards the CSIR for their financial support towards my studies. Thanks should also go to Dr T.L. Grobler for his input regarding image classification techniques, and for pointing me in the direction of capsule networks.
I would be remiss in not mentioning my flatmate, Henlo, for the constant supply of coffee during the write-up process. This was invaluable.
And finally, I am extremely grateful to my parents for their unwavering love and support.

UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / *Plagiarism Declaration*

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2023

Abstract

English

This study examines the application and comparison of several machine learning algorithms to the problem of classifying inverse synthetic aperture radar (ISAR) images of electrically small, geometrically complex targets. These algorithms include k-nearest neighbours, logistic regression, a fully connected neural network and a capsule network. A novel classifier is proposed, utilizing a capsule network with a reconstruction sub-network, to perform open-set classification. A dataset of synthetic ISAR images was created from simulated electromagnetic (EM) target returns and used to train and test the models. The EM simulation process was performed using a method of moments solver to compute the backscattering from models of the targets, which are represented as triangular meshes. Specifically considering geometrically small targets with low radar cross-sections, particular attention is paid to the performance of the classifiers when signals are received in the low signal-to-noise ratio regime. The use of a capsule network is found to be highly effective for both closed-set and open-set classification tasks, out-performing the other traditional machine learning and deep learning based classifiers investigated in this study (logistic regression, support vector machines, k-nearest neighbours and fully connected neural networks). The proposed method of comparing the images formed by the capsule network's reconstruction subnetwork to the input image is demonstrated to be an effective technique for identifying observations of ISAR images of targets that do not belong to any known classes, i.e. targets which are "unknown" to the classifier. Additionally, it is demonstrated that the use of the zero-mean normalised cross-correlation coefficient to compare the input and reconstructed images makes the proposed open-set recognition method more resilient to noisy inputs when compared to the use of the mean-squared error between the images. This addresses a commonly overlooked problem that an operational radar's automatic target recognition algorithm is not guaranteed to have been trained for all the possible target types that it will sense in the surveillance volume. The proposed classifier achieves an F1-score of greater than 0.9 for a test set containing two known and two unknown classes with signal-to-noise ratios of 6 dB and above.

Afrikaans

Hierdie studie ondersoek die toepassing en vergelyking van verskeie masjienleer-algoritmes wat gebruik word om ISAR beelde van elektries klein dog geometries kompleks teikens te herken. Hierdie algoritmes sluit die volgende tegnieke in: logistiese regressie, ondersteuningsvektor-masjiene, k-naaste bure en volledig gekoppelde neurale netwerke. 'n Nuwe klassifiseerder word voorgestel wat gebruik maak van 'n kapsule netwerk met 'n rekonstruksie-subnetwerk om oopstel klassifikasie uit te voer. ‘n Datastel van sintetiese ISAR beelde is geskep wat gebaseer is op die gesimuleerde electromagnetiese teiken refleksies en is gebruik vir die opleiding en toets can die herkennings algoritmes. Die EM simulasie proses het gebruik gemaak van die “method of moments” oplossings tegniek om die terug gekaatste sein vanaf die teikens te bereken. Die geometrie van die teikens in hierdie simulasie is voorgestel as ‘n stel gekoppelde driehoekies. Daar word spesifiek oorweeging gegee aan elektriese klein teikens met laë radardeursnitte, en daar word aandag geskenk aan die prestasie van die klassifiseerders wanneer hulle seine met laë sein-tot-ruis verhoudings ontvang as inset. Resulte dui aan dat die gebruik van 'n kapsule netwerk baie doeltreffend is vir beide geslotestel en oopstel klassifikasietake, en beter presteer as die ander tradisionele masjienleer- en diep-leer gebaseerde klassifiseerders wat in hierdie studie ondersoek is (logistiese regressie, ondersteuningsvektor-masjiene, k-naaste bure en volle gekoppelde neurale netwerke). Die voorgestelde tegniek om die beelde te vergelyk wat deur die kapsule netwerk se rekonstruksiesubnetwerk gevorm word met die invoerbeeld, is bewys om doeltreffend te wees om ISAR beelde van teikens te identifiseer wat nie tot enige bekende klasse behoort nie. Dit wil sê teikens wat onbekend is vir herkennings algoritmes hiermee identifiseer word. Boonop word daar aangetoont dat die gebruik van die nulgemiddeld-genormaliseerde kruiskorrelasie-koëffisiënt om die invoer en rekonstrureerde beelde te vergelyk, die voorgestelde oop-stel erkenning meer bestand maak teen ruiserige insette in vergelyking met die gebruik van die gemiddelde-kwadraat-fout tussen die beelde. Hierdeur word 'n algemene uitdaging in outomatiese teiken-herkenning aangespreuk waar 'n operasionele radarstelsel nie gewaarborg is om opgelei te wees op alle tiepes teikens wat in die die sensor se ruimte gewaar kan word nie. Die voorgestelde klassifiseerder behaal 'n F1-telling van meer as 0.9 vir 'n toetsstel wat twee bekende en twee onbekende klasse bevat, met sein-tot-ruis verhoudings van 6 dB en hoër.

Contents

List of Figures

List of Tables

Nomenclature

Variables and functions

λ	Wavelength.
f	Frequency.
θ	Elevation angle, measured from the horizontal.
ϕ	Azimuth angle.
ω	Angular frequency.
c	Speed of light.
Ω	Rotation angle.
E_s	Scattered electric field.
β	Bandwidth.
R_d	Down-range distance.
R_c	Cross-range distance.
U_d	Unambiguous down-range extent.
U_c	Unambiguous cross-range extent.
$\sigma(a)$	Logistic sigmoid function.

Acronyms and abbreviations

3D	Three-dimensional
ANN	Artificial neural network
ATR	Automatic target recognition
AV	Activation vector
AWGN	Additive white Gaussian noise
CAD	Computer-aided design
CEM	Computational electromagnetics
CNN	Convolutional neural network
CROSR	Classification-reconstruction learning for open-set recognition
CS	Compressive sensing
DFT	Discrete Fourier transform
EM	Electromagnetic
FCNN	Fully-connected neural network
FDTD	Finite difference time domain
FEM	Finite element method
FFT	Fast Fourier transform
FMCW	Frequency modulated continuous waveform
GMM	Gaussian mixture model
GO	Geometric optics
GPU	Graphics processing unit
GTD	Geometric theory of diffraction
HRR	High range resolution
HRRP	High range resolution profile
ISAR	Inverse synthetic aperture radar
KKC	Known known classes
KNN	K-nearest neighbours
KUC	Known unknown classes
LDA	Linear discriminant analysis
LFM	Linear frequency modulated
LOS	Line-of-sight
LR	Logistic regression
MLFMM	Multi-level fast multipole method

MoM	Method of moments
MPO	Modified physical optics
MRPO	Multiple reflection physical optics
MSE	Mean squared error
MUSIC	Multiple signal classification
NCTR	Non-cooperative target recognition
OSR	Open-set recognition
PCA	Principal component analysis
PEC	Perfect electrical conductor
PO	Physical optics
PTD	Physical theory of diffraction
RCS	Radar cross-section
ReLU	Rectified linear unit
RWG	Rao-Wilton-Glisson
SA	Sparse auto-encoder
SAR	Synthetic aperture radar
SBR	Shooting and bouncing rays
SF	Stepped-frequency
SNR	Signal-to-noise ratio
SVM	Support vector machine
UTD	Universal theory of diffraction
UUC	Unknown known classes
WGN	White Gaussian noise
ZNCC	Zero-mean normalised cross-correlation

Chapter 1

Introduction

1.1. Background

In recent years, the use of small multi-rotor drones has become increasingly popular for military, industrial and civil applications [1]. Consumer drones, which are widely available to the general public, can carry a variety of payloads, such as cameras, delivery packages or potentially more dangerous items, such as explosive devices. This raises several security and privacy concerns. Considering this, the ability to detect and classify drones has become a topic of interest.

As a sensor capable of operating under all weather conditions, day and night, radar is widely used for detecting, classifying and tracking targets. While radar traditionally focuses on large targets, such as aeroplanes, ships and large terrestrial vehicles, it can also be leveraged for use with small targets, such as consumer drones. However, some considerations should be taken when considering small targets. One of the primary considerations is that traditional large targets typically have a large radar cross section (RCS), while small targets, by virtue of their size, tend to have a smaller RCS. As a result, the radar returns from a tiny target are typically smaller in magnitude and therefore have a significantly lower signal-to-noise ratio (SNR). One means of increasing the SNR is to use radar imaging techniques, such as Inverse Synthetic Aperture Radar (ISAR). ISAR improves the SNR through the coherent integration of multiple radar pulses [2]. Another advantage of using ISAR for target recognition is that the images produced are highly interpretable, reflecting the physical structure of the target (with some similarity to optical images [3]). ISAR images can be presented to a human operator for classification; however, this requires significant costs and time to train and employ a human operator for these tasks. Additionally, human operators may be limited in their ability to perform this task with the required accuracy and speed. The time requirement is particularly noticeable at low SNRs, where the target structure is challenging to make out through visual inspection. Another limitation of using human operators identified by [4] is the time required for classification, typically several seconds.

Automatic systems for target recognition can be used to address some of the shortcomings of manual classification. Automated systems need only be implemented once and are capable of forming accurate predictions in minimal time (a couple of microseconds). Furthermore, modern image classification techniques have been shown to surpass human accuracy [5]. Numerous techniques for automating this process may be employed, ranging from specialised techniques developed for radar targets to general image recognition techniques.

The vast majority of classification algorithms aim to recognise new observations as one of several known classes. This uses the underlying assumption that all new observations belong to the closed set of classes used in training. While this is a reasonable assumption in certain controlled environments, this is not necessarily valid for an automatic target recognition system. In this case, it is not viable to train a model on every possible target that could be observed, and the system does not have control over the targets that present themselves during operation. It is therefore important to consider how an automatic target recognition system handles unknown targets that do not belong to any known classes.

1.2. Problem statement

Recognising targets using ISAR imaging poses challenges beyond those typically seen in general image recognition problems. One such challenge is that ISAR images of a target display high variation, particularly with changes in imaging angle. This variation includes both affine transformations (such as rotation and translation) as well as distortions resulting from the complex interaction between targets and the electromagnetic field and self-occlusion. Additionally, while ISAR imaging improves the SNR of the radar returns, there is often a significant noise component remaining in the resultant image. Very low SNRs from small targets with weak EM interactions present challenges for classification systems, which need to effectively differentiate between the noise and the underlying signal in order to form a prediction. When considering targets with small radar cross-sections, this is a notable issue. The final major challenge identified is that the targets observed by a radar system operating in the field are not guaranteed to belong to any known classes, or the measurement conditions may not be sufficient to accurately classify an observation. The common assumption that all targets belong to a closed set of known classes is therefore not generally valid. As such, it is necessary for a target recognition system to be capable of recognising when a given observation does not belong to any known class.

1.3. Project aims

The primary aim of this work is to investigate automatic target recognition methods capable of distinguishing between several small complex targets in real-time. To be of practical use in a real-world application, the system should meet the following requirements:

1. The system is insensitive to changes in the target's orientation.
2. The system is capable of operating at low SNRs, while maintaining high accuracy.
3. The system is not limited to a closed set of known targets and can handle observations of unknown targets.

The process for working towards this objective is broken down into the following steps:

1. Generate a dataset of suitable images that can be used for training and testing.
2. Develop an understanding of the methods and techniques used and gain an insight into how well these methods work through a comparative study of several machine learning approaches to target recognition.
3. Identify techniques that perform well on closed-set recognition tasks under the first two conditions (significant changes in imaging angle and low SNRs).
4. Extend promising closed-set recognition techniques to handle open-set recognition.

1.4. Scope

This work focuses on the recognition of small complex targets in a 'blue-sky' scenario. That is, only aerial targets surrounded by empty space are considered. This eliminates scenarios where clutter or multipath need to be considered. Furthermore, this study only considers cases where there is only one target present in any given observation. The tasks of detecting the presence of a target and segmenting the data to separate individual targets are recommended for future research. To simplify the acquisition of sufficient training data, this work makes use of computational electromagnetic (CEM) software to simulate the backscattering from small complex targets. As a further simplification, the targets used are modelled as perfect electrical conductor (PEC) meshes in free space. While full-wave CEM solvers were used to calculate the backscattered electric field as accurately as possible (as opposed to asymptotic solvers), the simulation results will differ from those obtained through real-world measurements. These simulations do not consider the deflections of the target structure during operation or moving parts on the target. Additionally, it is assumed that the motion of the target is known, and does not need to be estimated for the purposes of focusing or applying motion compensation techniques in the ISAR imaging process. Although the open-set classification problem is considered, it is only possible to perform testing on a limited number of small complex targets (which are used to represent both known and unknown targets).

1.5. Publications

Some of the findings of this study have been presented in the following publications:

- C. D. Stewart-Burger, D. J. Ludick and M. Potgieter, "A Comparison of Various Machine Learning Algorithms on ISAR Image Classification of Complex Targets with Varying Levels of Gaussian Noise," 2021 International Conference on Electromagnetics in Advanced Applications (ICEAA), 2021, pp. 228-228, doi: 10.1109/ICEAA52647.2021.9539533.

- C. D. Stewart-Burger, D. J. Ludick and M. Potgieter, "Open-set Classification of Small Complex Targets with ISAR Imaging," 2022 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (AP-S/URSI), 2022, pp. 1168-1169, doi: 10.1109/AP-S/USNC-URSI47032.2022.9887003.

- C. D. Stewart-Burger, D. J. Ludick and M. Potgieter, "Application of Capsule Networks to Open-set Target Recognition of ISAR Images of Small Complex Targets," 2022 International Conference on Electromagnetics in Advanced Applications (ICEAA), 2022, pp. 149-149, doi: 10.1109/ICEAA49419.2022.9899926.

1.6. Thesis outline

Following this introduction, Chapter 2 provides context for this work. This includes an overview of various radar signatures, image formation techniques, and classification algorithms that are applied to target recognition problems. This chapter discusses several similar and related studies from other researchers. The methods used to simulate the EM interactions of the targets, and the generation of the dataset of ISAR images used for training and testing are detailed in Chapter 3. Chapter 4 discusses the implementation of three widely used traditional machine learning algorithms. This discussion includes the theory and implementation details, how these models are trained on the dataset and a comparison of the performance of these algorithms. Several investigations into approaches aimed at improving the performance of these classifiers (through pre-processing or adaptions to the training process) are also discussed. A similar discussion on various deep learning techniques follows in Chapter 5. This chapter additionally includes sections on how these deep learning methods are adapted to handle open-set recognition problems. The final chapter concludes the thesis with a summary of the findings as well as a discussion on improvements and recommendations for future work.

Chapter 2

Literature Review

2.1. Radar signatures used in target recognition

Techniques used for radar target recognition are generally based on measurements of a target's dynamic characteristics, physical structure or a combination of these. Those techniques which exploit the target's physical shape are typically based on the platform's range-amplitude signature. In contrast, techniques reliant on the target dynamics make use of the platform's frequency-amplitude signature (based on Doppler shifts) [6]. This section introduces some of the radar signatures frequently used for classification purposes. While frequency-amplitude and range-amplitude data may be used separately (see Sections 2.1.1 and 2.1.2, respectively), imaging techniques which combine this data are of particular interest.

2.1.1. Doppler signatures

The earliest forms of primitive radar target recognition (still frequently used in modern battlefield radar [7]) involve converting radar returns to an audio Doppler signature and playing this audio to human operators [4]. These operators are trained to classify targets after listening to these tones. It requires significant time and effort to learn and perform manual target recognition, and this method relies heavily on the target dynamics (particularly size and velocity). The use of audio tones is only effective on slow-moving targets (with Doppler shifts in the audible range) and requires a couple of seconds of listening time per target. Reference [8] discusses the automation of the above task by using speech recognition techniques to replace the work performed by human operators. Targets with moving parts, such as propellers, rotors or jet engine fans, that move relative to the main body of the target add extra components to the Doppler signature. This effect, known as micro-Doppler, can be exploited by target recognition systems [9]. As Doppler-based techniques rely on target motion, these techniques are ineffective in recognising stationary targets.

2.1.2. High range resolution profiles (HRRP)

Information about the physical structure of a target can be deduced by generating high range resolution profiles (HRRP), which resolve the backscattering from the target into multiple range bins spanning the length of the target. High range resolution (HRR) techniques obtain this resolution by using a wide bandwidth pulse and then performing pulse compression. Range compression is usually done by applying a fast Fourier transform (FFT) to the backscattered data from a linear frequency modulated (LFM) chirp sampled at evenly spaced frequencies, or a frequency stepped waveform. Other methods, such as compressed sensing (CS), can also be employed [10]. Li et al. [11], and Zyweck et al. [12] demonstrate the use of HRRPs in target recognition systems aimed at classifying aircraft.

2.1.3. Radar imaging

Radar imaging goes a step further than HRR processing to produce a high-resolution two-dimensional image of a target. Geometrically, radar images represent the projection of the scattering centres of targets of interest onto a two-dimensional plane. Real aperture imaging, synthetic aperture radar (SAR) imaging and inverse synthetic aperture radar (ISAR) imaging are three different methods used to form radar images [3].

2.1.3.1. Real aperture imaging

While most radar imaging techniques require relative motion between the radar and the target, real aperture imaging uses large (hundreds of metres) antenna arrays to obtain a very narrow beam width. Real aperture radar can be challenging in practice due to the difficulties in setting up and calibrating a very large antenna array [3]. The advantage, however, is that this imaging technique is generally instantaneous. In contrast, motion-based methods require multiple pulses over time as the motion occurs, which are then subject to further signal processing.

2.1.3.2. Synthetic aperture radar (SAR) and inverse synthetic aperture radar (ISAR)

SAR overcomes the need for a large physical antenna by using multiple pulses from a small antenna that moves through space over time to create a synthetically large aperture. Examples of SAR images of two targets from the MSTAR dataset [13] are provided in Figure 2.1.

With ISAR, the emulated antenna baseline is formed through the target's movement rather than the movement of the antenna [6]. ISAR is generally more difficult than SAR in practice, as ISAR is typically used with non-cooperative targets whose motion cannot be controlled by the radar platform. The images formed by these techniques often closely resemble the physical shape of the target; however certain effects such as self-shadowing

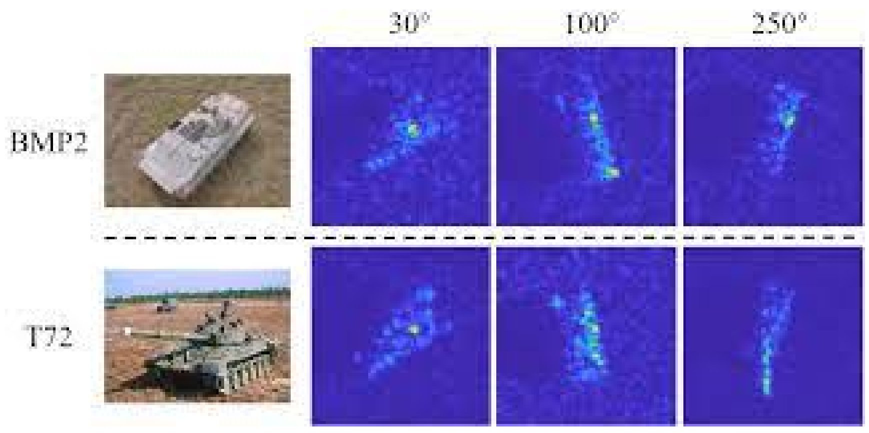


Figure 2.1: SAR images of two targets from the MSTAR dataset. The leftmost image in each row is a photograph of the targets, while the next three are SAR images from different imaging angles. [14]

and micro-Doppler can cause distortions and artefacts that fall outside the physical bounds of the target platform (sometimes referred to as *ghost scattering*). This resemblance to the physical geometry makes radar images highly intuitive and well suited to non-cooperative target recognition (NCTR) [3]. Figure 2.2 provides examples of ISAR images from [15], where the resemblance to the target can be seen despite some distortion. The processing

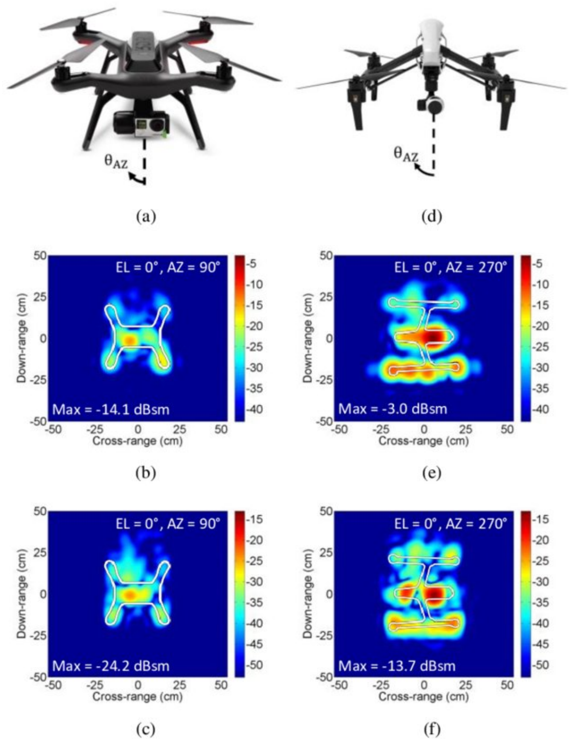


Figure 2.2: ISAR images (b), (c), (e) and (f) of drone targets (a) and (d), imaged from various aspect angles. [15] ©2016 IEEE.

required for SAR and ISAR image formation applies coherent integration, which improves the signal-to-noise ratio (SNR) of the radar returns. This makes SAR/ISAR particularly useful in situations with a low SNR. Compared to HRRP, ISAR has been shown to be more beneficial to the performance of target recognition systems [16]. Section 2.3 further discusses the formation of ISAR images.

2.1.3.3. Three-dimensional inverse synthetic aperture radar (3D-ISAR)

ISAR processing can be extended to three dimensions in cases where the target exhibits motion in both azimuth and elevation (relative to the radar). Seybold et al. [17] first extended ISAR processing to three dimensions. The concept was demonstrated by rotating targets on a turntable to obtain azimuth motion while the radar antenna was raised to change the relative elevation. While 3D-ISAR reveals a lot of information about the target geometry, 3D-ISAR remains challenging outside of an experimental setting, where the target's motion cannot be controlled. While 3D-ISAR remains an active area of research, the practical difficulties of using these techniques with non-cooperative targets limit its usefulness in NCTR.

2.2. Simulating high-resolution radar

The study of target classification using high-resolution radar requires enormous amounts of data — particularly when data-intensive methods like machine learning are used. Obtaining data from a high-resolution radar can be an expensive and challenging exercise. Access to both high-resolution radars and targets of interest is limited. This problem can be partially overcome through simulation. Simulations for ISAR imaging can be performed at several different modelling complexities. Studies such as [18] and [19] use relatively simple modelling techniques, where the target is modelled by several point-scatters in three-dimensional space. Figure 2.3 shows an example of a target represented by numerous point-scatterers and the resulting ISAR image formed in [18]. Such models eliminate the need for computationally expensive CEM methods but are generally only applicable to specific problems and do not provide accurate solutions for different target types and scenarios [20].

Because simplistic scattering models are generally not sufficiently accurate for simulation, the rest of this section focuses on methods that more accurately calculate the backscattering based on the target geometry. Various CEM methods capable of calculating the backscatter from a geometric representation of the target can be used for such simulations. These CEM methods can be broadly split into full-wave (exact) and asymptotic (approximate) methods, which are discussed in the following sections.

2.2.1. Full-wave methods

Full-wave methods are used to solve Maxwell's equations [21]. Solutions can be found using either the integral form of the equations, with techniques such as the method of

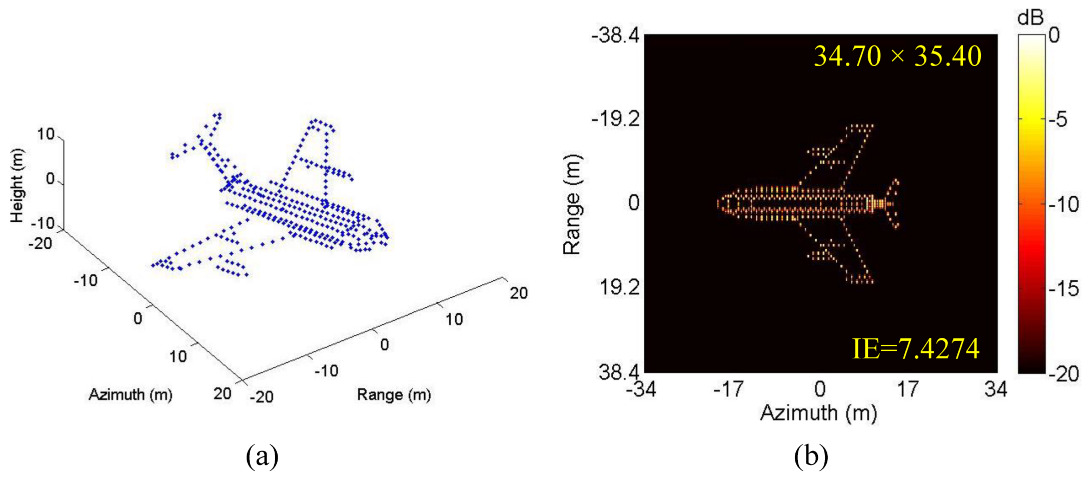


Figure 2.3: Three-dimensional point scatter model (a) used in [18] with the resulting ISAR image (b).

moments (MoM), or the differential form, using methods such as the finite element method (FEM) or the finite difference time domain method (FDTD).

2.2.1.1. Method of moments (MoM)

MoM is a well know frequency-domain method used to solve electromagnetic boundary or volume integral equations [22]. To apply this method to high-resolution radar simulation, the target geometry is broken down into a mesh of small triangular patches that can be described by Rao-Wilton-Glisson (RWG) basis functions. The moment equation,

$$\mathbf{ZI} = \mathbf{V}, \tag{2.1}$$

where $\mathbf{Z}$ is the impedance matrix and $\mathbf{V}$ is a voltage excitation vector, is set up and solved to calculate a vector of RWG expansion coefficients, $\mathbf{I}$ [23]. To obtain an accurate solution, the geometry should be discretized into a mesh with the size of each mesh element on the order of $\lambda/10$. Problems discretized with N mesh elements require an impedance matrix of size N^2 to be stored in memory, and the runtime for solving (2.1) with a direct linear solver scales with N^3 [24]. As a result, the memory and runtime requirements for solving problems with electrically large geometries (i.e. structures that are large relative to the wavelength of the excitation) become prohibitively high. Solving (2.1) can be accelerated using methods such as the multi-level fast multi-pole method (MLFMM). However, even with such techniques, the runtimes for problems with a very large number of elements (N) render this method impractical.

2.2.2. Asymptotic methods

Numerous methods that approximate the solution to Maxwell's equations have been proposed that trade accuracy for computational efficiency [25]. Asymptotic methods thus overcome the limitations of full-wave methods when applied to electrically large problems. Significantly, these methods also increase in accuracy as the electrical size of the target increases. Asymptotic methods can be further divided into ray-based methods and current-based methods. The former neglects the wave-based properties of electromagnetic radiation entirely, while the latter makes smaller approximations and serves as an intermediate between ray-based and full-wave methods.

2.2.2.1. Geometric optics (GO)

GO is a ray-based method that uses rays to trace specular reflections from the target geometry. While being a very fast method, GO suffers from many inaccuracies. Flat or singularly curved surfaces are known to sometimes produce infinite scattering results under GO [26]. While conventional GO neglects diffraction and other edge effects, extensions such as the geometric theory of diffraction (GTD) or the uniform theory of diffraction (UTD) are sometimes used in conjunction with GO to account for these effects.

2.2.2.2. Shooting and bouncing rays (SBR)

Similarly to GO, SBR also traces the specular reflections of many rays or ray tubes launched at the target. However, this method differs from GO in that an equivalent source is placed at each interaction point between the rays and the target geometry. The scattering from the target can be calculated by integrating over these equivalent point sources. SBR is used in [27] to simulate ISAR images of targets with moving parts.

2.2.2.3. Physical optics (PO)

The PO approximation assumes that the surface current is limited to the regions of the target with line-of-sight visibility to the source, with the surface current over any shadowed regions being equal to zero. Being a current-based method, PO relates the incident magnetic field to the induced surface current over the illuminated regions. While PO is typically more accurate than GO or SBR, this approximation still neglects certain scattering mechanisms, such as surface waves and edge diffraction. Modified PO (MPO) and the physical theory of diffraction (PTD) have been proposed as extensions to conventional PO that account for edge diffraction phenomena. For complex geometries with concave regions, the PO approximation performs poorly. In such instances, PO can be applied recursively to model multiple reflections. This is referred to as multiple reflection PO (MRPO). Wang et al. [28] make use of PO to simulate ISAR imaging of satellites.

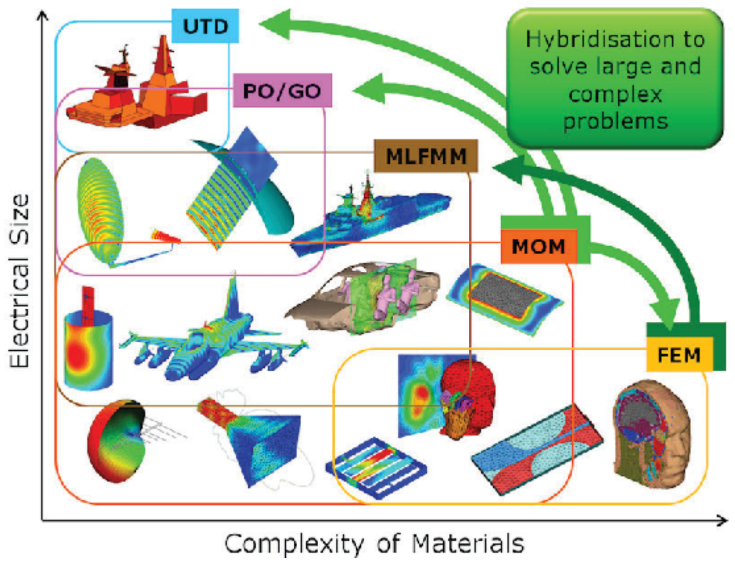


Figure 2.4: Various CEM methods and their applications according to [29]. © 2013 IEEE.

Each of the aforementioned CEM methods trades accuracy for computational efficiency. Figure 2.4 illustrates each of these methods and their appropriate use cases, with coarser asymptotic methods becoming increasingly appropriate with increasing electrical size.

2.3. ISAR image formation

Several methods for forming ISAR images exist, ranging from relatively straightforward conventional approaches to significantly more involved approaches proposed to either improve on certain qualities of the image formed (such as super high-resolution techniques) or to form ISAR images under specific scenarios (including sparse sampling). While a small degree of rotational motion is required for ISAR imaging, any excess motion (additional rotation or translation) can result in a degradation of the image quality in standard ISAR image formation techniques. For this reason, a number of motion compensation algorithms which eliminate the effects of the target's motion can be applied. This section gives a brief overview of some of the ISAR image formation and motion compensation techniques commonly used in the literature.

2.3.1. Conventional methods

Conventional methods include Fourier-based and time-frequency-based methods, both of which are discussed in [30]. The simplest technique for ISAR image formation is a Fourier-based method, which is valid under the conditions of a small angle ($\Omega \leq$ approx. $6°$ [31]) and narrow bandwidth relative to the centre frequency ($\beta < \frac{f_c}{10}$ [31]). This method involves taking a two-dimensional Fourier transform according to the following k-space formulation [15]:

$$Image(r_d, r_c) = \int\int E_s(f,\phi)e^{jk_x r_d}e^{jk_y r_d}dk_x dk_y \tag{2.2}$$

where

$$k_x = \frac{4\pi f}{c}\cos\phi \approx \frac{4\pi f}{c}, \qquad k_y = \frac{4\pi f}{c}\sin\phi \approx \frac{4\pi f_c}{c}\phi \tag{2.3}$$

r_d and r_c are the down-range and cross-range respectively, f is the frequency, ϕ is the angle of rotation, c is the speed of light and E_s is the backscattered electric field as a function of f and ϕ. This method is used in [15] to form ISAR images of small consumer drones. Blomerus et al. [32] use a time-frequency method to form ISAR images of small targets.

2.3.2. Other methods

In instances where the small angle and narrow bandwidth approximations are invalid, other methods or additional motion compensation algorithms are required. One such algorithm, viz. back-projection, is demonstrated in [33] to produce high-resolution ISAR images using frequency-modulated continuous waveform (FMCW) radar. Zang et al. make use of another method based on compressive sensing (CS) to form high-resolution ISAR images from limited observations. Other methods aimed at enhancing the resolution of the images formed include the multiple signal classification (MUSIC) [34] algorithm which is demonstrated by Odendaal et al. [35] and methods that make use of deep learning, such as [36].

2.3.3. Motion compensation

Some motion compensation methods, such as polar reformatting and phase gradient, can be used to apply conventional Fourier-based ISAR processing techniques in situations where either the small angle or narrow bandwidth assumptions are not valid [32]. Reference [37] demonstrates the use of polar reformatting to apply Fourier-based ISAR processing to form images of fighter aircraft using both simulated and measured radar cross-section data (RCS) data. Other motion compensation algorithms aimed at eliminating the effects of both rotation and translation of the target are discussed in [30].

2.4. Classification techniques

Target classification can be performed in several ways, such as through manual inspection by trained radar operators or using algorithmic methods. The majority of modern target recognition systems employ some kind of machine learning approach to the problem — often in combination with a human operator (i.e. a man-in-the-loop system). That is, they use training data comprised of numerous examples of possible observations to form models capable of recognising new observations. Machine learning-based target recognition can be broadly split into two categories, namely traditional machine learning and deep learning, which are reviewed in the following section.

2.4.1. Traditional machine learning

Traditional machine learning typically relies on a few (often hand-crafted) features in the form of low-dimensional feature vectors that characterise the data to form models. The performance of these techniques is generally linked to the choice of features used. This creates a significant designer overhead in instances where the features are hand-crafted. The requirement to intelligently pick hand-crafted features can be avoided by using some form of dimensionality reduction, such as Principle Component Analysis (PCA) [38] or Linear Discriminant Analysis (LDA) [38] on a high-dimension feature space, such as the pixel intensities in an image. Both PCA and LDA are, however, limited to linear combinations of the original features. More sophisticated methods to extract complex features include Fourier transforms and polar mapping [39] and wavelet coefficients [40]. Open literature reports on the application of numerous different machine learning algorithms to recognise targets from ISAR images.

2.4.1.1. Other studies using traditional machine learning

Botha [41] discusses the use of a nearest-neighbour classifier to classify ISAR images of military aircraft. The study compares the use of extracted features (viz. geometrical moments, invariant features based on moments, shape features and quantized energy strips) to the use of ISAR data (in the form of pixel intensities) as inputs to the nearest neighbour classifier. Similarly, Park et al. [39] also use a nearest-neighbour classifier, but go a step further by preprocessing the ISAR images to achieve rotational and translational invariance in an attempt to make the recognition system less sensitive to changes in imaging angle. Using a similar technique aimed at extracting invariant features, Cexus et al. [42] compare several machine learning techniques, including Naïve Bayes, Support Vector Machine (SVM), K-nearest neighbour (KNN) and neural network classifiers.

2.4.2. Deep learning

Deep learning is a subset of machine learning that makes use of multiple layers of repeating units to form artificial neural networks (ANNs) [43]. As the name suggests, ANNs draw inspiration from the brain. The basic structure of an ANN is the artificial neuron (discussed further in Section 5.1), which was proposed to mimic the function of living neurons. In contrast to traditional machine learning approaches, where there is a focus on extracting a small number of highly discriminative features, in the deep learning paradigm, there is a tendency to use a very high dimensional feature space as an input. The rationale here is

that while feature extraction can decrease the complexity of the classification problem, the resulting loss in information causes a decrease in performance. Neural networks are able to handle the computational complexity associated with processing high-dimensional inputs since these models are highly parallelisable, and can thus make use of GPU acceleration. The first few layers of a deep learning model perform complex feature extraction processes, while classification happens in the final layers.
The most popular deep learning models used in conjunction with ISAR images are Fully Connected Neural Networks (FCNN) [41, 42] and Convolutional Neural Networks (CNNs) [32, 44–48]. Other deep learning methods have also been used in the literature. For example, in [49], a Sparse Autoencoder (SA) network is used to perform feature extraction on ISAR images, after which a Softmax classifier was used for classification based on the extracted features. In another recent paper, Zhou et al. [50] report on the use of a Capsule Network [51] for ISAR image recognition. In Section 5.4, Capsule Networks will be explored in more detail, where a method will be proposed to classify targets that are not included in the training set. While some papers, such as [42] and [41], experiment with preprocessing steps to perform feature extraction, most models in literature use the raw ISAR images as inputs to the neural networks.

2.5. Open-set recognition (OSR)

Most classification systems are limited to a fixed number of predefined classes, making the assumption that all observations belong to a closed set. However, this is a poor assumption for most automatic target recognition systems. A radar system operating in the field has little guarantee that it will only observe targets belonging to certain classes. Moreover, in many circumstances, designers of an automatic target recognition system may not have examples of all possible classes available for training. As such, there is a necessity to develop systems that can reliably determine when an observation does not belong to any known classes. Advanced systems may use some form of unsupervised online learning to form new classes from previously unknown classes, while simpler systems may simply flag unknown targets as such. Regardless, the first step is to determine whether or not an unlabelled observation can be classified according to the known classes.
In OSR, classes can be categorised according to three basic categories according to [52]:

1. *Known known classes* (KKCs) - Labelled classes with examples available for training.

2. *Known unknown classes* (KUCs) - Labelled examples available for training/validation that do not form part of the known classes. This can include miscellaneous examples that are not part of any distinct class.

3. *Unknown known classsses* (UUCs) - Classes for which no training examples are available.

Modifications for various well-known machine learning algorithms have been proposed to extend their capabilities to handle unknown classes. A comprehensive summary of these methods and modifications can be found in [53], which discusses methods for both traditional machine learning and deep learning models. The methods discussed in [53] are split into two broad categories:

1. OSR for discriminative models, in which a closed-set model is typically modified after training to identify unknown observations based on some threshold.
2. OSR for generative models, which includes methods that generate synthetic examples of UUCs to be used as training examples.

Notable modifications include OpenMax [54], which is proposed to replace softmax [55] (frequently used in the final layers of deep learning networks) to handle observations of unknown classes. A novel approach to OSR called Classification-Reconstruction learning for Open-Set Recognition (CROSR) [56] proposes the use of models trained to both classify and reconstruct observations from latent representations. In this approach, the accuracy of the reconstruction is used as a metric to detect unknown classes.

2.5.1. OSR for automatic target recognition

Open-set recognition is frequently overlooked, and there appears to be very little research into OSR of ISAR images available in open literature. To the best of the author's knowledge, the only paper that applies OSR to ISAR images to be found in open literature is [57], in which the authors used a CNN with an OpenMax [55] layer to classify ISAR images of ships of both known and unknown classes. Zhao *et al.* [57] report that using an OpenMax layer resulted in an 11.8 % gain in precision when compared to the use of a softmax layer in one experiment. A few recent papers, however, investigate OSR applied to SAR images [58–60].

Proposing an edge exemplar method which extracts the (closed) boundary of each known class, [58] demonstrates a classifier with an accuracy of 96.04 % when tested on the MSTAR dataset [13] where three of the targets are considered known classes and the remaining seven are considered unknown classes. Unfortunately, no indication is given regarding the impact on the known class recognition rate when enforcing closed decision boundaries for the known classes using this method.

The open-set classification problem is decomposed into two tasks by [60]: classification and anomaly detection. For this task, a network inspired by a conditional generative adversarial network (CGAN) is proposed. After training a generator and a discriminator on the known classes, the discriminator learns the distribution of the known class data and outputs both a class prediction and a score which indicates the confidence that the input belongs to the classes observed in training. Using the score as an anomaly detector, it is possible to differentiate known classes from unknown classes. This method reportedly outperforms that proposed in [58], achieving an accuracy of 98.8 % when using the same dataset.

2.6. General image recognition

While there are notable differences between ISAR images and optical images [42], the similarities should not be overlooked. Many of the challenges in ATR and general image recognition are shared, and general image recognition is a well-researched field in which constant progress is being made. For many years, state-of-the-art image recognition models were CNNs [61–63]. Recently, transformer networks have begun to outperform CNNs [64, 65] and show promise for ATR.

One of the significant challenges associated with ISAR image classification is the significant within-class variation seen in ISAR images of a target imaged from different angles. Some of these challenges have been addressed by general image classification techniques. While CNNs are insensitive to translations in images through translational invariance, they are sensitive to other deformations and affine transformations. Some studies have proposed image recognition methods that apply preprocessing techniques, such as Radon and Fourier-Mellin transforms, to input images to achieve invariance with respect to other transformations, such as scale, rotation and blur [66, 67]. These techniques could benefit ATR systems, in which such transformations are prevalent. In 2017, Sabour et al. [68] proposed a model called Capsule Networks (CapsNet), which achieved state-of-the-art classification performance on the MNIST dataset [69] (images of handwritten digits). Importantly, the proposed CapsNet architecture extracts equivariant features, which potentially allow it to recognise observations of an object with different instantiation parameters (such as translation, rotation or distortion) as equivalent to one another. This appears well suited to the significant variation found in ISAR images over changes in imaging angle. Capsule Networks have been shown effective when applied to SAR image classification [70–74].

2.7. Existing works on ISAR image recognition

There is little existing research on the recognition of ISAR images of small targets under a metre in length. Studies on small targets such as consumer drones seem to focus on other radar signatures, such as micro-Doppler [75] rather than ISAR imaging as used in this work. Existing research into various state-of-the-art techniques for the recognition of both small and medium targets is discussed below.

2.7.1. Small targets

Reference [76], investigates the use of an eight-layer CNN (five convolutional layers, and three fully connected layers) to classify ISAR images of three targets ranging from 30 cm to 60 cm in size. Measurements for the ISAR images were taken in an anechoic chamber at Ka-band with bandwidths of both 4 GHz and 8 GHz. The authors report a classification accuracy of 96.00 % and 87.78 % for bandwidths of 4 GHz and 8 GHz, respectively. Their findings confirm that images formed with narrow bandwidths (resulting in lower range resolution) significantly impacted the level of structural detail captured in the ISAR images formed. This in turn impacts the classifier's ability to differentiate between targets.

2.7.2. Medium-sized targets

There is significantly more research available on ISAR image recognition focusing on targets with dimensions of a couple of metres (i.e. in the 1–5 m range). Yang *et al.* [44, 77, 78] investigate the recognition of ISAR images of satellite targets. All three of these studies use the same set of ISAR images generated through simulation using mesh targets of five different satellites. It is noted that with satellite targets, it is generally only necessary to consider a small range of elevation angles from which the satellite can be imaged [50]. This significantly decreases the variation within each class.

In [77], the ISAR images are pre-processed to extract low-dimensional features before classification is done. By using a low-dimensional feature space for classification, the computational efficiency of the target recognition process is reduced. This is more relevant as the electrical size of the targets (and the size of the ISAR image formed) increases. Using a support vector machine (SVM) classifier, a classification accuracy of between 97.4 % and 99.8 % (depending on the training/testing data split) is reported across the five targets.

An ensemble of pre-trained CNN models (trained on optical images) is used with the same dataset of space targets in [78]. To overcome some of the limitations of deep CNNs (particularly overfitting and sensitivity to the imaging conditions), the dataset was augmented by applying various transforms such as rotation, contrast adjustment and scaling to the images in the dataset. The use of pre-trained models makes it possible to make use of large CNN models which typically require an enormous amount of training data with only a small dataset of ISAR images required to perform transfer learning. Individually, the classification accuracies obtained using the pre-trained models varied from 69.43–94.43 % depending on the model and the fine-tuning process. Used together as an ensemble, however, the reported classification accuracy is 97.36 %. Notably, a CNN model trained on the ISAR data from scratch reportedly achieved a classification accuracy of 79.25 % — in the same region as the transferred pre-trained models. The takeaways from this study are that the use of models pre-trained on optical images can be effectively utilised to recognise ISAR images (with a varying degree of success depending on the transfer learning is done), and that a stacking ensemble yields better results than any model used independently.

Rather than focusing on improving the classification accuracy further, [44] focuses on improving the computational efficiency by proposing an algorithm to search for the optimal neural network architecture within a specific search space. This enables the researchers to find a model which maintains a high classification accuracy while being a significantly smaller, lightweight model compared to those previously investigated.

Zhou *et al.* [50] propose a novel attention-augmented deformation robust ISAR image recognition network. This is demonstrated on a different dataset containing ISAR images of four satellites generated from simulated data (using a physical optics based solver). Amongst other state-of-the-art neural network architectures, the network proposed by [50] is compared to a capsule network. In the results, the reported accuracy of the capsule network is 88 %, while the proposed network achieves a classification accuracy of 91 % across the four satellites in the practical test set. The paper finds that the proposed network is more robust with respect to the deformations found in ISAR images than other networks.

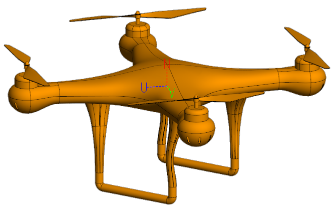
Chapter 3

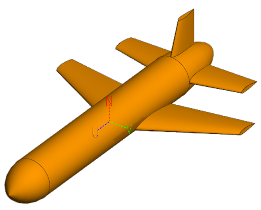
Dataset Generation

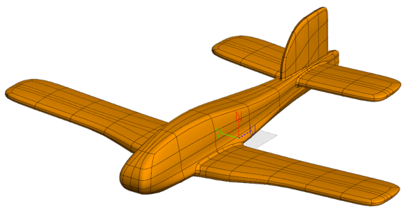
Training supervised machine learning models requires a considerable number of labelled training examples. Obtaining a sufficient number of ISAR images of several complex targets through field measurements is challenging and expensive, particularly if one intends to obtain measurements from numerous aspect angles. Synthesising these measurements through simulations makes it possible to generate a large amount of data that can be used for training and testing. Another advantage of simulation is that data can be generated for targets to which the designers of a target recognition system do not have physical access or only have restricted access. This is an important consideration, particularly for military applications, where there is an interest in non-cooperative enemy platforms (distinct from cooperative, friendly targets which respond to interrogation with a predefined coded signal to identify themselves). For these reasons, the decision was made to use simulated data to generate a dataset of synthesised ISAR images in this study. The electromagnetic simulations were carried out using FEKO [79], after which the scattering data were processed using a MATLAB script to form images. The dataset includes ISAR images of various small complex targets, viz., a multi-rotor drone, a missile, a model fixed-wing aircraft and a flying-saucer type UFO. In addition to the multiple aspect angles for each target, Gaussian noise was added to the ISAR images at various signal-to-noise ratios (SNR). Multiple targets and SNRs were added to the dataset for a more realistic radar environment classification investigation.

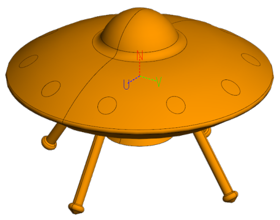
3.1. Choice of target models

Renderings of the five CAD models used to represent the selected targets are shown in Figure 3.1. All the selected targets are between 0.5 m and 1 m in length along their maximum dimension. The multirotor drone (Figure 3.1a) and the generic cruise missile (Figure 3.1b) targets were selected as typical targets of interest for a radar system, and represent the 'known' classes. These targets are chosen for their significantly different geometries, which should make the differentiation between these targets a relatively simple problem. The remote-controlled model fixed-wing aircraft (Figure 3.1c) and the flying-saucer type UFO (Figure 3.1d) are selected as examples of other aerial targets. Specifically,


(a) Multirotor drone.


(b) Generic cruise missile.


(c) Model fixed-wing aircraft.


(d) UFO (flying saucer).

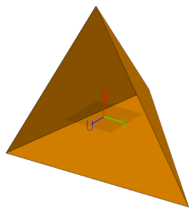

(e) Trihedral corner reflector.

Figure 3.1: CAD models of the selected targets.

these two targets are chosen as examples of targets that designers of ATR systems are less likely to consider, making them an appropriate choice to represent ‘unknown’ targets. Importantly, these targets display structural similarity to the first two targets. The model fixed-wing aircraft and the generic cruise missile both have an elongated body/fuselage, three tail fins and two larger fins/wings attached to the body. While the multirotor drone and the UFO targets display less visual similarity, the UFO’s body and four legs resemble the drone’s body and four arms. This resemblance is more apparent in the ISAR images produced, as discussed in Section 3.8. The similarity between the targets in each of these pairs presents a more challenging classification problem.

The intentional choice of targets displaying varying degrees of similarity makes it possible to differentiate between classification systems with reasonable performance, capable of differentiating between dissimilar targets, and excellent classification systems, capable of distinguishing between targets with a similar structure.

The fifth target, a trihedral corner reflector (Figure 3.1e), is somewhat of an outlier, being a simple geometrical structure rather than a complex target. Despite the simple shape, a trihedral displays significant variation in its scattering return as the aspect angle changes. Specular scattering dominates the returns for imaging angles where illumination occurs at right angles to the trihedral faces, while imaging angles looking into the concave region are dominated by multiple reflections. Section 5.2 discusses how this simple, well-known reference structure is used to represent arbitrary structure for open-set recognition.

All of the CAD models used in this study are based on models publicly available for download and use [80]. The models were improved using CADFEKO to ensure they could be converted into a mesh appropriate for the computational electromagnetic (CEM) methods used in this study. For simplicity, the models are approximated as perfect electrical conductor (PEC) structures.

3.2. Dataset overview

The generated dataset contains nearly 1.2 million 27 px × 21 px ISAR images. A breakdown of the number of ISAR images in the dataset is given in Table 3.1.

Table 3.1: Breakdown of the number of ISAR images in the dataset

No. of unique elevation angles	*19*
No. of unique azimuth angles	*242*
No. of unique viewing angles	4598
No. of SNRs	*17*
No. of noise samples per SNR	*3*
No. of ISAR images per target	234498
No. of target models	5
Total no. of ISAR images	**1172490**

For each of the five chosen targets, images were generated over a range of aspect angles from -45 to 45 degrees elevation and 360 degree coverage in azimuth. Through the addition of additive white Gaussian noise (AWGN) to the simulated data, the ISAR images are repeated at 17 SNRs ranging from -24 dB to 24 dB at 3 dB intervals. The remainder of this chapter gives a theoretical background on ISAR imaging and the techniques used to simulate and generate these images.

3.3. ISAR theory

3.3.1. Electromagnetic scattering

The electromagnetic pulse transmitted by a radar induces surface currents on targets that interact with the radiated field. These surface currents, in turn, radiate an electromagnetic field known as the scattered field. A portion of the scattered field is propagated in the direction of the radar receiver, where this excitation is sensed. The scattered field is a function of the target geometry, motion and material composition (amongst other things, such as frequency, aspect angle etc.), and this information is thus encoded into the scattered field [3]. Target classification systems rely on these phenomena and utilise the characteristics of the received signal to extract information about the target platform to form predictions. Because the scattered field originates from surface currents distributed across the target, it is possible to differentiate between the scattering contributions from different parts of the target, given sufficient resolution. The term *high-resolution radar* is generally used to describe systems capable of resolving individual scattering centres on a single target.

3.3.2. Imaging plane

ISAR is a technique used to achieve high resolution in two dimensions. The scattering contributions are projected onto a two-dimensional plane separated into multiple down-range and cross-range bins (also termed range of cross-range cells). The projection plane is shown in Figure 3.2. The down-range dimension (y) is aligned with the radar line-of-sight (LOS) (r_a). The second dimension, cross-range (x), is formed perpendicular to both the down-range axis (y) and the axis of rotation of the target (ω_0).

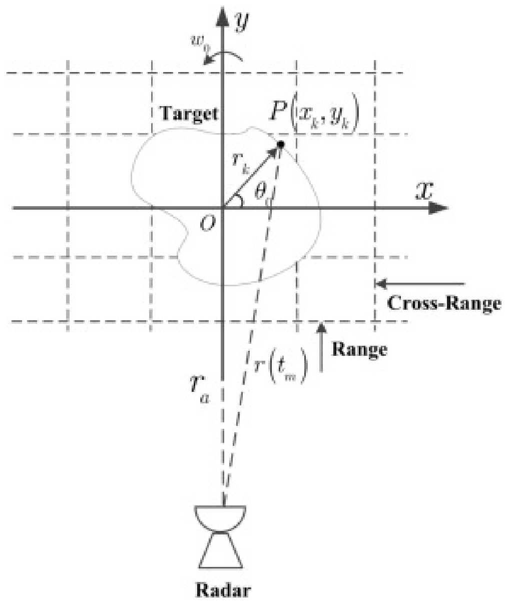


Figure 3.2: Projection plane used in ISAR imaging, with the cross-range and down-range axes depicted by x and y, respectively [81]. (Reproduced with permission from Elsevier.)

3.3.3. Down-range resolution

For a single-frequency pulsed radar with constant amplitude A, the waveform can be expressed as

$$s_T(t) = A \sin(2\pi f_c t) \operatorname{rect}\left(t - \frac{T}{2}\right) \tag{3.1}$$

where f_c is the carrier frequency and $rect(t)$ is the rectangular window function, given as

$$rect(t) = \begin{cases} 1, & \text{for} - \frac{T}{2} \leq t \leq \frac{T}{2} \\ 0, & \text{elsewhere.} \end{cases} \tag{3.2}$$

In the case of a monotonic pulsed waveform such as this, the range resolution (i.e. the minimum distance at which two scattering centres can be differentiated) is inversely proportional to the pulse duration T. The relationship between the pulse duration T and the range resolution ΔR can be expressed as

$$\Delta R = \frac{cT}{2} \tag{3.3}$$

where c is the speed of light. Figure 3.3 shows that for scattering centres with a down-range separation less than this (i.e. the distance required for a round trip between the scattering centres, the echos begin to overlap.

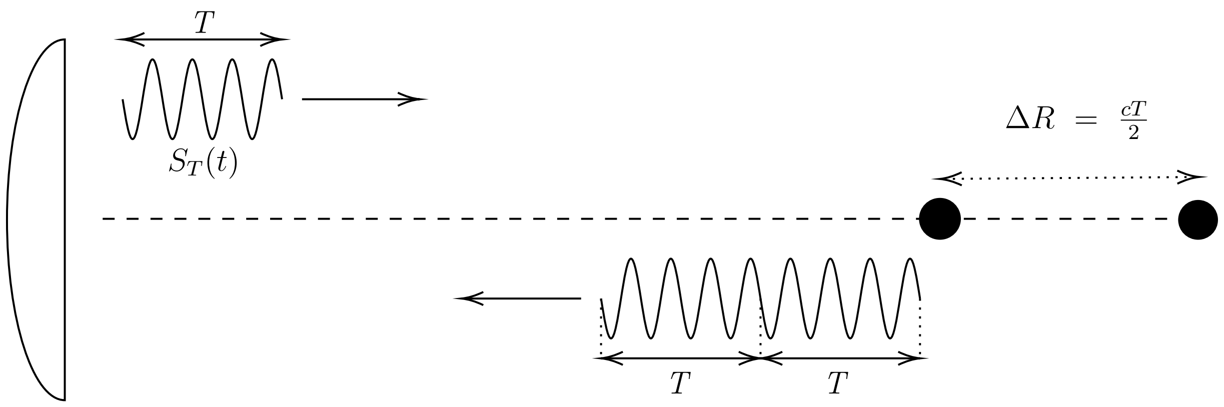


Figure 3.3: Range resolution of a monotone pulsed radar is limited by the spacing (in time) between echoes from points separated in range. The resolution corresponds to the spacing which reduces the gaps between echos of the returned pulses to zero.

Given that the energy in the transmitted pulse is proportional to the pulse duration, we find that when using a monotone pulsed radar, there is a trade-off between the energy in the received echo (and therefore SNR) and the range resolution as the pulse duration is varied. This trade-off is often unacceptable, particularly when considering small targets with a small RCS. Using a frequency-modulated waveform and applying pulse compression makes it possible to use a longer pulse duration (and therefore maintain a reasonable SNR) without sacrificing range resolution. Both linear frequency modulated (LFM) and stepped frequency (SF) waveforms are possible candidates for this approach. In both of these cases, it can be shown that the range resolution ΔR_d is inversely proportional to the bandwidth β of the transmitted signal [32],

$$\Delta R_d = \frac{c}{2\beta}\,. \tag{3.4}$$

3.3.4. Cross-range resolution

Scattering centres contained within the same range bin of a rotating target can be separated by the Doppler shift induced by this motion. Figure 3.4 illustrates that for a target with a rotational velocity of ω (rad/s) about a fixed axis perpendicular to the radar line-of-sight, the radial velocity of a scattering centre situated at a distance r_c from the centre of rotation

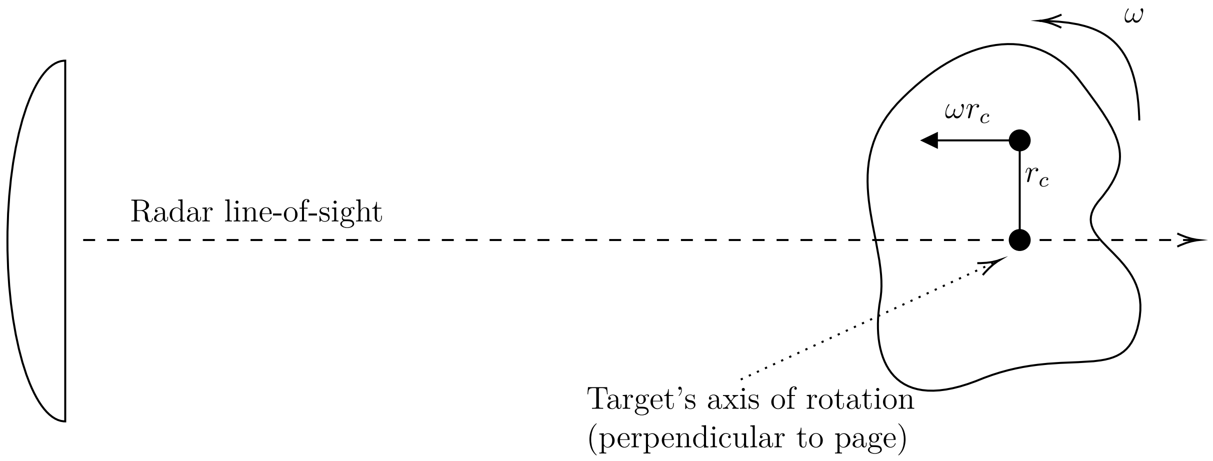


Figure 3.4: Radial velocity of a scattering point induced by a target's rotation. The velocity vector of an arbitrary point on the target is depicted by the solid arrow labelled ωr_c where r_c is the radial distance to the centre of rotation.

in the cross-range direction is ωr_c. This radial velocity induces an instantaneous Doppler shift,

$$\begin{aligned} f_D &= \frac{2\omega r_c f_c}{c} \\ &= \frac{2\omega r_c}{\lambda} \end{aligned} \tag{3.5}$$

where f_c is the centre frequency of the radar. Using (3.5), it can be found that two scattering centres in the same range bin, separated by a distance δr_c in the cross-range direction, exhibit a frequency difference

$$\delta f_D = \frac{2\omega \delta r_c}{\lambda}. \tag{3.6}$$

Rearranging (3.6), we find

$$\delta r_c = \frac{\lambda \delta f_D}{2\omega}. \tag{3.7}$$

From this, it is apparent that the cross-range resolution is proportional to the Doppler resolution. Let the assumption be made that the angular velocity, ω remains constant over the coherent integration time T_N. For an SF radar that transmits a pulse train, $s_T(t)$, with N pulses over a duration of T_N per burst, the received signal from two scatterers in the same range bin with different Doppler shifts, f_{D1} and f_{D2}, will be the sum of these two frequency-shifted echos, $s_{R1}(t)$ and $s_{R2}(t)$. Figure 3.5 shows the spectra of the components of the received signal with the minimum frequency difference required to resolve the individual components according to the Rayleigh criterion. This is met when the peak of one of the components coincides with the minimum (magnitude) of the other. For signals with a pulse duration of T_N, these minima occur $\frac{1}{T_N}$ Hz on either side of the

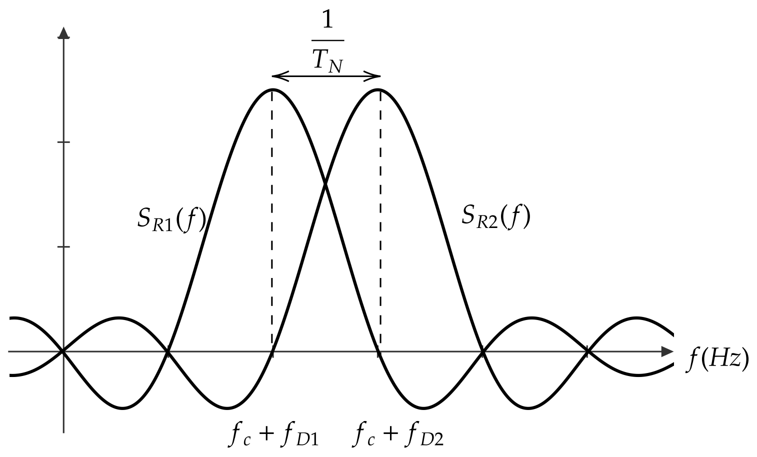


Figure 3.5: Fourier transforms of two received echoes at the Rayleigh criterion.

peak. The Doppler resolution can therefore be expressed as

$$\Delta f_D = \frac{1}{T_N}\,. \tag{3.8}$$

The cross-range resolution, ΔR_c can therefore be expressed by setting δf_D in (3.7) to the Doppler resolution as expressed in (3.8),

$$\begin{aligned}\Delta R_c &= \frac{\lambda \Delta f_D}{2\omega} \\ &= \frac{\lambda}{2\omega T_N} \\ &= \frac{\lambda}{2\Omega}\end{aligned} \tag{3.9}$$

where $\omega T_N = \Omega$ is the angle through which the target rotates during the observation period.

3.3.5. Unambigious range

Another factor to consider is the extent of the region imaged in each dimension. This range is easily expressed as the product of the resolution in each dimension and by the number of samples used. The unambiguous down-range extent U_d is therefore given by

$$\begin{aligned}U_d &= N\Delta R_d \\ &= \frac{Nc}{2\beta}\end{aligned} \tag{3.10}$$

for an SF radar with N steps or an LFM radar sampled at N equally spaced intervals. In this case, the bandwidth can be expressed as $\beta = N\Delta f$, where Δf is the frequency step size. From this, (3.10) simplifies to

$$U_d = \frac{c}{2\Delta f}\,. \tag{3.11}$$

When the extent of the target in the down-range dimension is greater than the unambiguous range U_d, scattering points that lie outside of this are not imaged correctly. Depending on the type of radar used, these scattering points will either be missed or, in the case of an SF radar, will result in aliasing. This effect, known as *fold-over*, is discussed with examples in [30].

For the unambiguous range extent in the cross-range dimension, the same logic applied to the calculation of the down-range extent can be used. For an ISAR image generated from M range profiles (from the same number of chirps/bursts over the observation time), the cross-range resolution U_c can be expressed (using (3.9)) as

$$\begin{aligned} U_c &= M\Delta R_c \\ &= \frac{M\lambda}{2\Omega}\,. \end{aligned} \tag{3.12}$$

If the target rotates through an angle Ω with a constant angular velocity over M chirps/burst, the angle of rotation between bursts (ie. the angle step-size) can be expressed as $\Delta\phi = \frac{\Omega}{M}$. Using this, (3.12) can be written as

$$U_c = \frac{\lambda}{2\Delta\phi}\,. \tag{3.13}$$

3.4. ISAR image formation

Once the far-field backscattered field, $E_s(f, \phi)$, has been calculated over a range of frequencies and incident angles and arranged according to (3.18), ISAR images can be formed. The ISAR image $f(x, y)$ is related to the scattered field data according to

$$f(x,y) = \frac{1}{BW_k\Omega}\int_{\phi_1}^{\phi_2}\int_{k_1}^{k_2} E_s(k,\phi)e^{2j(k\cos\phi x + k\sin\phi y)}dkd\phi \tag{3.14}$$

where $k = \frac{2\pi f}{c}$ is the wave-number and $BW_k = k_2 - k_1$ is the wave-number bandwidth [37]. In situations where the frequency bandwidth is small relative to the centre frequency f_c and the imaging angle is small, the approximations

$$\begin{aligned} k &\approx k_c \\ &= \frac{2\pi f_c}{c} \end{aligned} \tag{3.15}$$

and

$$\begin{aligned} \cos\phi &\approx 1 \\ \sin\phi &\approx \phi \end{aligned} \tag{3.16}$$

can be made, respectively. Under these conditions, [31] shows that (3.14) can be approximated by the 2D inverse Fourier transform of the backscattered field. The simulations in this work, however, do not respect these conditions and cannot make direct use of this approximation. To form ISAR images under conditions where a large bandwidth (relative to f_c) and large imaging angle are used, a technique called polar reformatting is used.

3.4.1. Polar reformatting

Sampling the backscattered electric field uniformly with respect to the frequency and azimuth angle results in a non-uniform sampling grid in the spatial-frequency domains (i.e. the k_x-k_y plane) as illustrated in Figure 3.6 [31]. Noting that $k_x = k\cos\phi x$ and

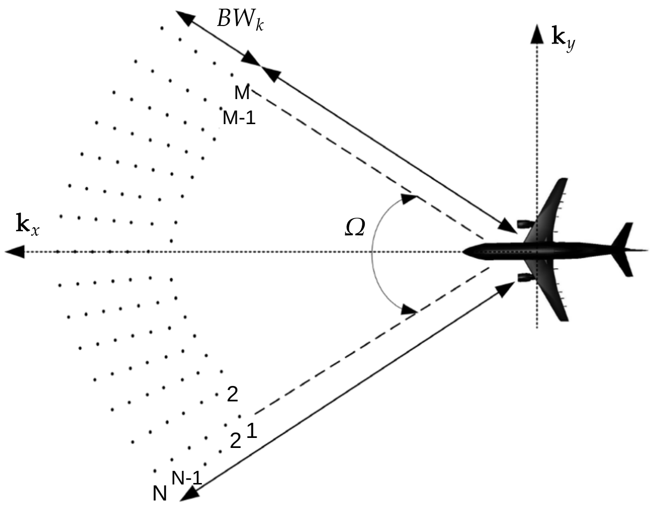


Figure 3.6: Sampling points in the spatial-frequency domain [31]. (Reproduced with permission from John Wiley and Sons.)

$k_y = k\sin\phi y$, (3.14) can be written as

$$f(x,y) = \frac{1}{BW_k\Omega}\int_{\phi_1}^{\phi_2}\int_{k_1}^{k_2} E_s(k,\phi)e^{2j(k_x+k_y)}dkd\phi\,. \tag{3.17}$$

From this, it is clear that there is a Fourier relationship between k_x and x and between k_y and y. The use of a discrete Fourier transform (DFT), however, requires uniform sampling of the data in this (k_x-k_y) domain. This can be achieved by interpolating the data sampled in a polar format to a uniform Cartesian grid, as shown in Figure 3.7 (i.e. polar reformatting). In this study, polar reformatting is done using bilinear interpolation.

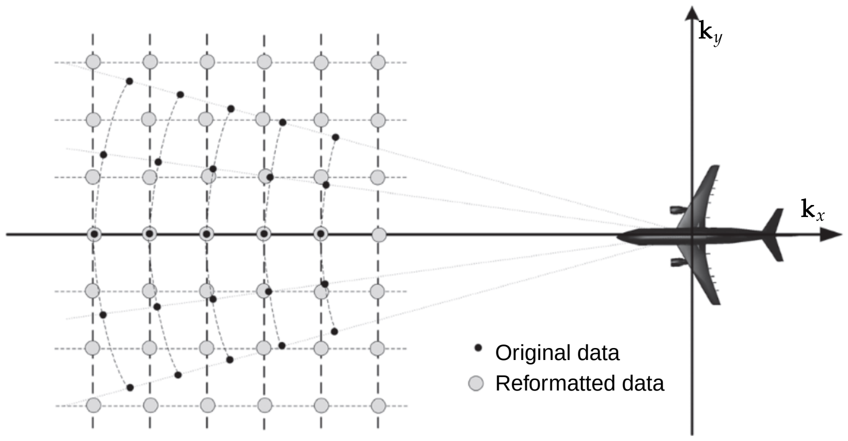


Figure 3.7: Interpolation of the electric field data from a polar format to a uniform Cartesian grid [31]. (Reused with permission from John Wiley and Sons.)

While the interpolation in polar reformatting is known to introduce unavoidable numerical noise, [54] notes that this method is significantly less computationally complex than numerical integration of (3.14). Before using a 2D FFT on the reformatted data to form an image, the data is windowed using a 2D Hamming window with a sidelobe level of $-42.5\,\text{dB}$. The use of this windowing function reduces the sidelobe levels of the point-spread function, resulting in a clearer image with better-defined peaks at a slight cost to the resolution (as a result of the greater bandwidth of the Hamming window compared to a rectangular window).

3.5. Simulation

In this study, the measurement of each target with a single polarised (VV) SF radar operating at C-band (with a centre frequency $f_c = 5.55\,\text{GHz}$) was simulated. While higher frequencies could potentially be used to obtain a higher resolution, using a C-band with a limited bandwidth radar emphasises the challenges associated with classifying small targets. At the chosen centre frequency, the electrical sizes of the targets are around 15λ-18λ. This is small enough that full-wave CEM simulations remain a viable option, while asymptotic methods may suffer from a loss in accuracy at smaller scattering centres located on the targets. Simulations of the EM interactions with the PEC target models were therefore carried out using the MoM solver in FEKO [79]. Using this solver, the backscattered electric field was calculated for each target, illuminated by a vertically polarised plane wave over a range of frequencies and incident angles. The simulation parameters, such as the frequencies and azimuth angle steps used, are discussed later in this section. The incident angles used for these measurements range from -45 to 45 degrees elevation and 360 degree azimuth coverage of the target (with azimuth steps calculated in Section 3.5.2) at each elevation, as illustrated in Figure 3.8. This angular coverage ensures that the dataset represents a significant variety of aspect angles from which ISAR images are formed. Once

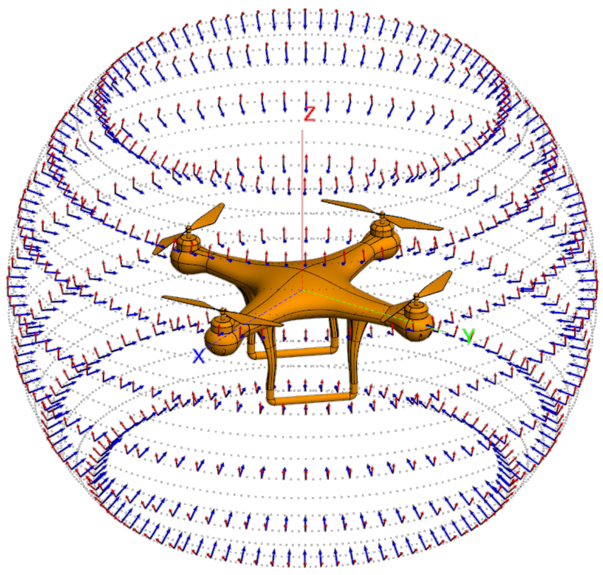


Figure 3.8: FEKO configuration to measure the RCS of a target from many incident angles.

the backscattering has been calculated for each target, these data can be organised into matrices that correspond to measurements taken for a synthetic aperture. Specifically, the measurements obtained by an SF radar sensing a target with a turntable rotation can be synthesised by organising the data into a matrix such as

$$E_s(f,\phi) = \begin{bmatrix} E_s(f_1,\phi_1) & E_s(f_1,\phi_2) & \cdots & E_s(f_1,\phi_m) & \cdots & E_s(f_1,\phi_M) \\ E_s(f_2,\phi_1) & . & . & . & . & . \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ E_s(f_n,\phi_1) & . & . & E_s(f_n,\phi_m) & . & . \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ E_s(f_N,\phi_1) & . & . & . & . & E_s(f_N,\phi_M) \end{bmatrix} \tag{3.18}$$

where $E_s(f_n,\phi_m)$ is the far-field backscattered electric field from the target for frequency f_n and azimuth angle ϕ_m at a single elevation. Since this study considers a single polarised radar, only the vertically polarised component of the scattered electric field is recorded. ISAR processing can then be used on these data to form an image of the target.

3.5.1. Image parameters

To obtain the necessary information about a target to perform tasks such as classification, it is important that the resolution of the ISAR images is fine enough to discern discriminating features. A finer resolution results in more information about the structure of the target being imaged. It is also important that the imaging extents are sufficient to image the full target.

For many applications, it is a sensible choice to attempt to use the same resolution for both the down-range and cross-range. Doing so avoids perceived distortions in the resulting image due to different scales on each dimension. In light of this, a resolution of 5 cm was chosen for both dimensions, which results in 10 samples on the smallest target.

Considering that the selected targets have a maximum size of 1 m along the longest dimension, an unambiguous range in each dimension of 1 m may seem to be sufficient. In practice, however, it is noted that smearing in the down-range direction (possibly due to delayed scattering due to complex scattering mechanisms such as multiple reflections or the broadening of specular returns due to windowing) causes the image of the target to extend beyond the unambiguous range. The undesirable aliasing resulting from this is avoided by adding an additional 25 % to the unambiguous range in the down-range dimension. The desired imaging parameters designed for are shown in Table 3.2.

Table 3.2: Image parameters.

Image parameter	**Value**
Down-range resolution (ΔR_d)	5 cm
Cross-range resolution (ΔR_c)	5 cm
Down-range unambiguous range (U_d)	1.25 m
Cross-range unambiguous range (U_c)	1.0 m

3.5.2. Simulation parameters

From Section 3.3, it can be seen that the down-range resolution, down-range unambiguous range, cross-range resolution and unambiguous cross-range are determined by the bandwidth, frequency step-size, imaging angle and imaging angle step-size, respectively. These simulation parameters are therefore carefully chosen in consideration of the desired image parameters. Table 3.3 shows the simulation parameters calculated using the relationships discussed in Section 3.3. Note that the step sizes for both the frequency and angle increments are rounded down such that the total number of steps to get to the bandwidth or imaging angle, respectively, is an integer value.

Table 3.3: Calculated simulation parameters.

Simulation parameter	**Value**
Bandwidth (β)	3.00 GHz
Frequency step-size (Δf)	115.4 MHz
Number of frequency steps (N_f)	27
Imaging angle (Ω)	30.9°
Angle step-size ($\Delta\phi$)	1.5°
Number of angle steps (N_a)	21

3.6. Simulating different noise-levels

In order to study the impact of SNR on the classification algorithms investigated, this work required the dataset to contain examples of ISAR images generated from data sampled at various noise levels. To achieve this, white Gaussian noise was added to the sampled backscattered electric field data, $E(f, \phi)$, before any ISAR processing was performed. Images were generated at SNRs ranging from -24 dB to 24 dB with 3 dB increments while adjusting the power of the noise relative to that of the sampled electric field.

3.6.1. Defining SNR

SNR is conventionally defined as the ratio of the mean power of the signal to the mean power of the noise [82]. In the case of a signal, $x[n]$, with additive white Gaussian noise with variance σ^2, the SNR is conventionally defined as

$$SNR = \frac{\sum_{n=0}^{N-1} |x[n]|^2}{N\sigma^2} \,. \tag{3.19}$$

From this definition, it is apparent that any energy-conserving transform, such as the Fourier transform, does not affect the SNR defined in (3.21). Once an ISAR image is formed, it is clear that in this domain, the energy of the signal is limited to certain localised regions (i.e. the scattering centres), while the noise energy is evenly distributed over the entire domain (i.e. white). In this case, the conventional definition, which averages the signal power over the entire domain, is not appropriate. This work, therefore, adopts a different definition of SNR, proposed by [82], which only considers regions where the energy is above the -3 dB point relative to the maximum (similar to the definition of SNR used in communications where the signal is only considered in its bandwidth). For this definition, we consider a signal $x[n]$ and let

$$\mathcal{B} = \{n : |x[n]|^2 \geq 0.5 \times max(|x[n]|^2)\} \,. \tag{3.20}$$

Reference [82] then defines the SNR as

$$\text{SNR} \triangleq \frac{\sum_{n\in\mathcal{B}} |x[n]|^2}{|\mathcal{B}|\sigma^2} \tag{3.21}$$

where $|\mathcal{B}|$ is the cardinality of $\mathcal{B}$. This definition provides a more meaningful measure of SNR and is used in this study to determine the required variance σ^2 of the white Gaussian noise necessary to achieve the desired SNRs. It is also noted that augmenting the (training) dataset through the addition of noise is also a widely used technique in machine learning that has been shown to reduce overfitting [83]. To make use of this, three independent noise samples were added to each synthetic aperture for each SNR. Other processing steps frequently used in ISAR image generation are normalisation and upsampling.

3.7. Normalisation and upsampling

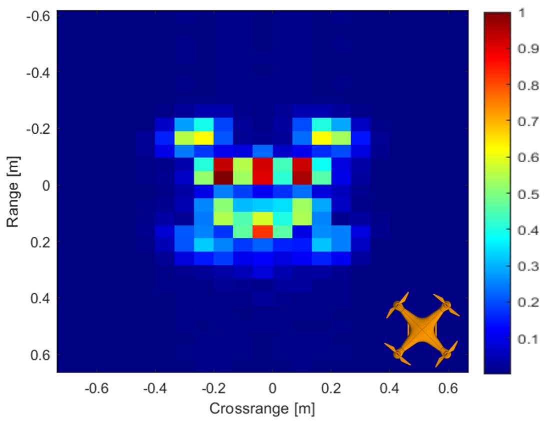
Normalisation can ensure that the images are independent of factors affecting the magnitude, such as range, antenna gain, etc. This is desirable for classification tasks, where these factors offer little to no information regarding the identity of the target. In this work, each ISAR image is normalised to the peak absolute value in the final image.

The perceived resolution of the output image can be increased by zero-padding the sampled aperture before applying the 2D Fourier transform. This technique is frequently used in the formation of ISAR images, as it results in visually appealing images with more pronounced scattering centres. While this technique superficially improves the image quality and may make the data more interpretable (to human operators), this is merely a method of interpolating the data (often termed DFT interpolation) and does not actually improve the resolution or add information. Because no information is added, zero-padding is not used for the most part in this study. In Section 4.4.5, a small experiment is carried out with images which have been upsampled using this technique to investigate its impact on machine learning based classification algorithms.

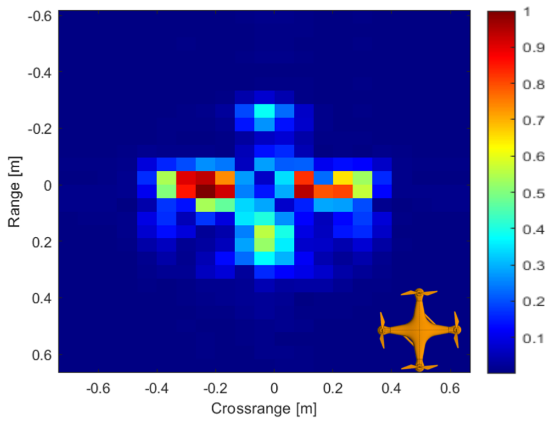
3.8. Visualising the dataset

Each ISAR image is stored as a 2D matrix of complex values. Figures 3.9 to 3.13 display normalised ISAR images of each of the target models, imaged from various aspect angles. The absolute values of each pixel are used to plot these images. The CAD models superimposed on the bottom right of each image indicate the approximate orientation of the target. In all of these figures, the negative range direction points towards the radar. The visual similarity between Figure 3.9b and Figure 3.12a provides a good demonstration of how ISAR images of two targets with dissimilar geometries can result in very similar ISAR images. It is quite conceivable that a human operator or machine learning model (particularly one which has been trained on only one of these targets) may misclassify these images. This highlights the necessity for a robust open-set classification algorithm capable of discriminating between similar images such as these.

Figure 3.14 provides examples of ISAR images generated at various SNRs. Here the target and aspect angle have been kept constant.

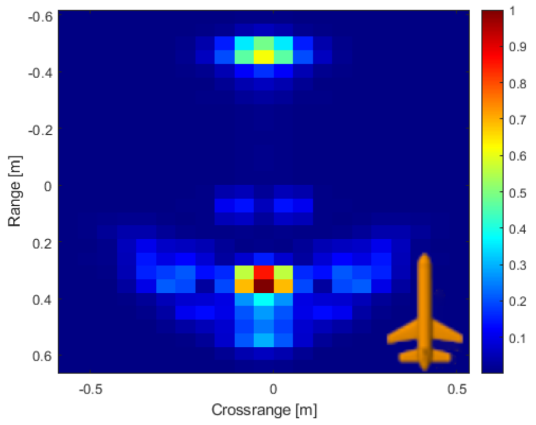


(a) 0° elevation, 0° azimuth

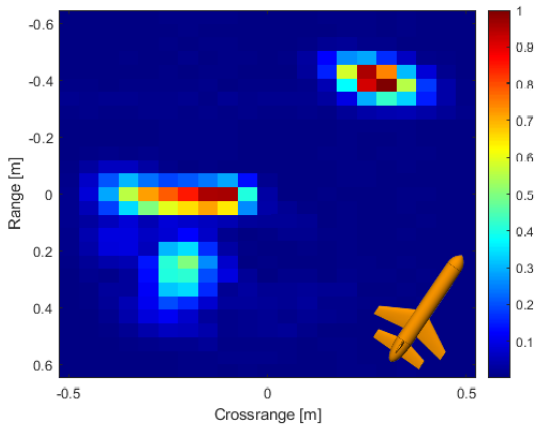


(b) 0° elevation, 45° azimuth

Figure 3.9: ISAR images of the multirotor drone centred at different azimuth angles.

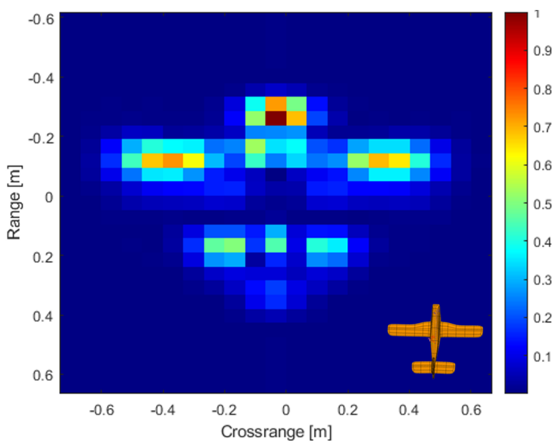


(a) 0° elevation, 0° azimuth

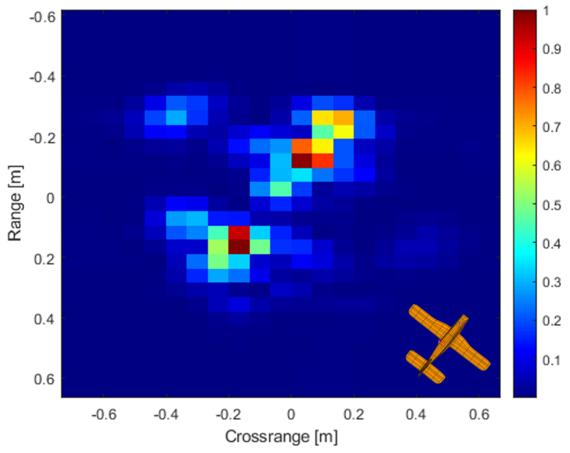


(b) 0° elevation, 45° azimuth

Figure 3.10: ISAR images of the generic cruise missile centred at different azimuth angles.

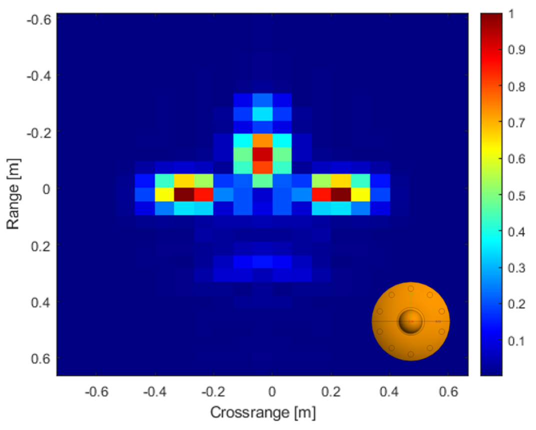


(a) 0° elevation, 0° azimuth

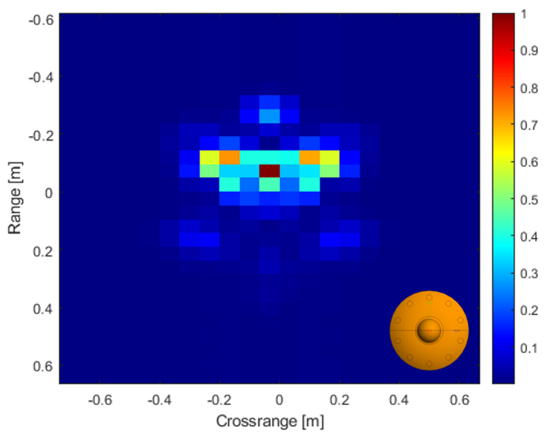


(b) 0° elevation, 45° azimuth

Figure 3.11: ISAR images of the model fixed-wing aircraft centred at different azimuth angles.

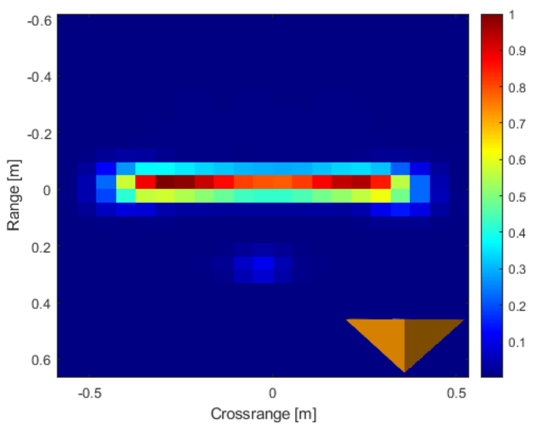


(a) 0° elevation, 0° azimuth

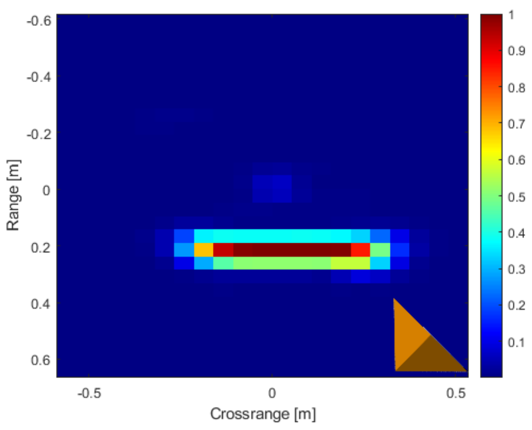


(b) 0° elevation, 45° azimuth

Figure 3.12: ISAR images of the flying-saucer type UFO centred at different azimuth angles.

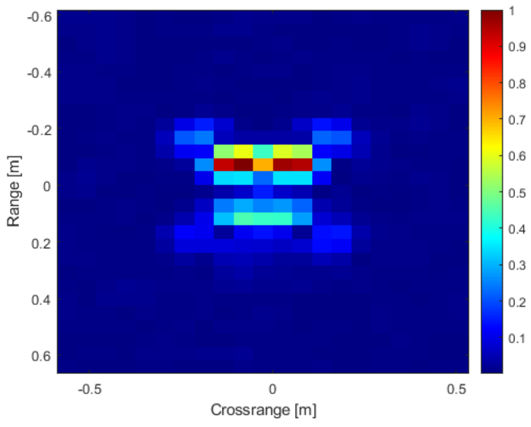


(a) 0° elevation, 0° azimuth

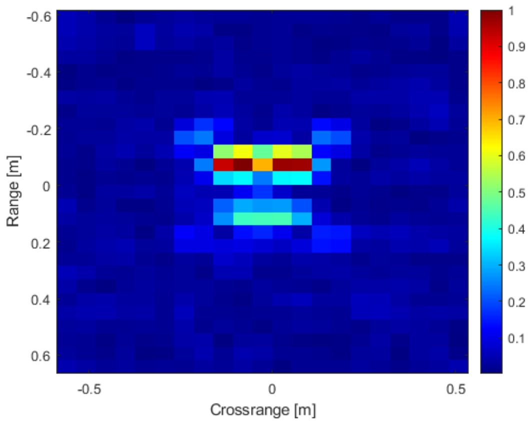


(b) 0° elevation, 45° azimuth

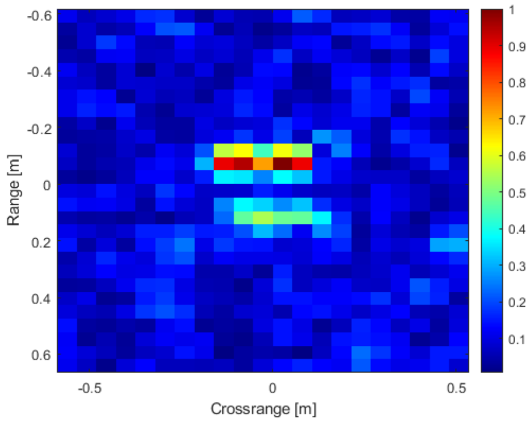
Figure 3.13: ISAR images of the trihedral corner reflector centred at different azimuth angles.



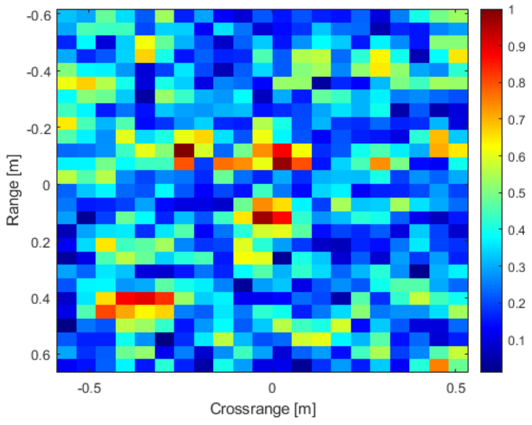
(a) SNR = 24 dB



(b) SNR = 12 dB



(c) SNR = 0 dB



(d) SNR = −12 dB

Figure 3.14: ISAR images of the multirotor drone with added noise at various SNRs.

3.9. Conclusion

After first outlining the theory behind ISAR imaging, this chapter details the process taken to generate a dataset of ISAR images of several small complex targets. Efforts were made to make the simulated data a better approximation of real-world measured data through the addition of white Gaussian noise. In the following chapters, this dataset is used to train and test machine learning classifiers to perform automatic target recognition.

Chapter 4

Traditional Machine Learning Approaches

The applications of three traditional machine learning algorithms (viz. logistic regression, k-nearest neighbours and support vector machine) are discussed in this chapter. All of these techniques are well-known machine learning approaches that have been applied to a broad range of problems in other studies. Each of the chosen algorithms has a slightly different approach to the classification task and therefore comes with its own set of pros and cons. These differences are discussed in detail in the relevant sections for each technique. It was specifically decided to investigate both linear and non-linear classifiers to gauge the degree to which a (simple) linear classifier is applicable. While none of these approaches is a state-of-the-art image recognition technique, there are a number of reasons to investigate their performance. The first motivation behind the use of these algorithms is to generate a baseline for the dataset with which the performance of more advanced (deep-learning) machine-learning approaches, discussed in Chapter 5, may be compared. Secondly, through this investigation, insight into the difficulty and nature of the classification task at hand can be gained. The use of a logistic regression classifier, specifically, gives insight as to whether the classes are linearly separable in the feature space (or to what extent this approximation works). Additionally, the impact of variability in the training- and test sets (with regards to both the imaging angle and SNR) is investigated here. This is more easily achieved with traditional machine learning algorithms than with deep learning models, as there are fewer hyper-parameters to tweak during these investigations.

4.1. Logistic regression (LR)

Logistic regression (LR) is a linear classification algorithm that calculates a hyper-plane that best separates two labelled classes. Being a probabilistic model, LR estimates the probability that an observation belongs to each class, rather than merely assigning a predicted class. The probability of a class prediction is related to the distance of the observation from the hyperplane with the logistic function. Being a linear classifier, LR cannot create complex decision boundaries. While LR is limited in its ability to model complex datasets, its simplicity means that LR has very fast prediction times. This is relevant for real-time classification, which is essential when considering the classification of targets that pose a potential threat. Being a probabilistic discriminative model, LR directly estimates $P(C = k|\mathbf{x})$, the probability that an observation $\mathbf{x}$ belongs to class k (rather than modelling the joint distribution $p(\mathbf{x}, y)$, as is the case with a probabilistic generative model). This has the advantage of more efficiently utilising the training data to distinguish between classes while discarding properties of the data which are common between classes and offer no discriminative value. Although the LR classifier is a binary classifier, a generalised extension, known as softmax regression or multinomial logistic regression, can be used for multi-class classification [84]. It is worth noting that many deep-learning models, such as that used in [49], implement a Softmax classifier as the final layer of the model. As a result, the logistic regression classifier gives a good baseline with which the performance of other models (such as the model to be discussed in Section 5.2) can be compared to.

Despite the fact that the ISAR dataset considered in this study is unlikely to be linearly separable, the use of LR helps gauge whether a linear approximation can still be used. If a linear classifier still provides reasonable performance at a reduced computational cost, it is worth consideration. Section 4.1 provides a theoretical background of LR and how the model parameters can be found, followed by details on how this algorithm was implemented.

4.1.1. Theory

Logistic regression gets its name from the logistic sigmoid function,

$$\sigma(a) \triangleq \frac{1}{1 + \exp(-a)}, \tag{4.1}$$

which maps the input a to a value between 0 and 1, as illustrated in Figure 4.1. Specifically, considering a binary classification problem with classes $\mathcal{C}_1$ and $\mathcal{C}_2$, the logistic sigmoid function is used to relate the posterior probability to the observation $\mathbf{x}$ according to

$$P(C = \mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0), \tag{4.2}$$

where $\mathbf{w}$ and w_0 are the trainable model parameters. This expression can be simplified by redefining $\mathbf{w}$ to include the bias term and prepending a 1 to the observation feature vector as shown:

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_0 \\ \vdots \\ x_d \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}. \tag{4.3}$$

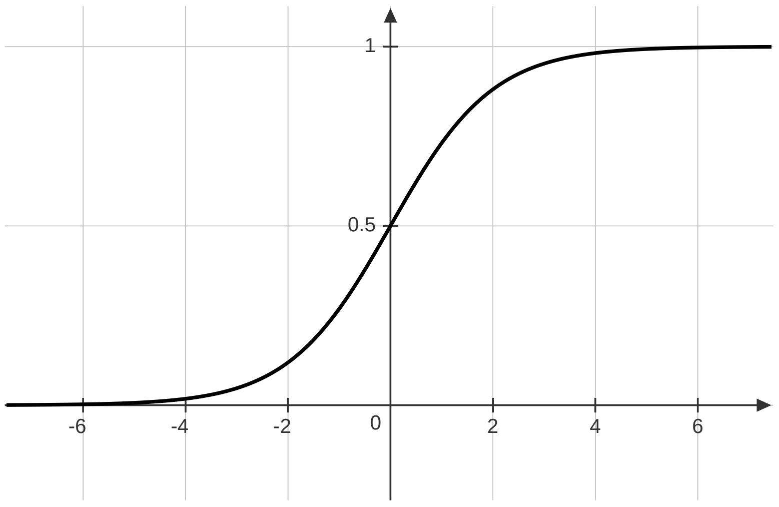


Figure 4.1: The logistic sigmoid function, $\sigma(a)$.

With this new definition, the posterior for $\mathcal{C}_1$ becomes

$$P(C = \mathcal{C}_1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}). \tag{4.4}$$

The decision boundary, i.e. at $P(C = \mathcal{C}_1|\mathbf{x}, \mathbf{w}) = P(C = \mathcal{C}_2|\mathbf{x}, \mathbf{x}) = 0.5$, is given by

$$\mathbf{w}^T\mathbf{x} = 0. \tag{4.5}$$

4.1.1.1. Training

Let us consider a training dataset $\mathcal{D} : \{\mathbf{x}_n, y_n\}$, where y_n denotes the class membership of each observation $\mathbf{x}_n$ (let $y_n = 0$ and $y_n = 1$ correspond to $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively). Fitting a logistic regression model to the training data involves finding values for $\mathbf{w}$ that maximise the likelihood of the data conditioned on $\mathbf{w}$, $p(\mathcal{D}|\mathbf{w})$.
For convenience sake, the observations and class labels in the training data can be separated as $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ and $\mathbf{y} = \{y_1, y_2, ..., y_n\}$. Assuming that y_n is only dependent on $\mathbf{x}_n$, the likelihood to be maximised can be written as,

$$\begin{aligned} p(\mathcal{D}|\mathbf{w}) = p(X, \mathbf{y}|\mathbf{w}) &= P(\mathbf{y}|X, \mathbf{w})p(X|\mathbf{w}) \\ &= p(X) \prod_{n=1}^{N} p(y_n|X, \mathbf{w}) \\ &= p(X) \prod_{n=1}^{N} P(y_n|\mathbf{x}_n, \mathbf{w}), \end{aligned} \tag{4.6}$$

noting that X is independent of $\mathbf{w}$. Given that y_n follows a Bernoulli distribution, we can express the final term in (4.6) as

$$\begin{aligned} P(y_n|\mathbf{x}_n, \mathbf{w}) &= P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})^{y_n}(1 - P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w}))^{1-y_n} \\ &= \sigma(\mathbf{w}^T\mathbf{x}_n)^{y_n}(1 - \sigma(\mathbf{w}^T\mathbf{x}_n))^{1-y_n}, \end{aligned} \tag{4.7}$$

where (4.4) has been used. Substituting (4.7) into (4.6), the likelihood becomes

$$p(\mathcal{D}|\mathbf{w}) = p(X)\prod_{n=1}^{N}\sigma(\mathbf{w}^T\mathbf{x}_n)^{y_n}(1 - \sigma(\mathbf{w}^T\mathbf{x}_n))^{1-y_n}. \tag{4.8}$$

While maximising (4.8) will theoretically yield values that correctly fit the model parameters $\mathbf{w}$ to the data $\mathcal{D}$, this expression can lead to issues with numerical stability in practice (resulting from the product of many small terms). It is, therefore, common practice to rather minimise the negative log-likelihood,

$$E(\mathbf{w}) \triangleq -\ln p(\mathcal{D}|\mathbf{w}) = -\sum_{n=1}^{N}\{y_n \ln\sigma(\mathbf{w}^T\mathbf{x}_n) + (1-y_n)\ln(1 - \sigma(\mathbf{w}^T\mathbf{x}_n))\} - \ln p(X). \tag{4.9}$$

It can be seen that this expression can be minimised by making $\mathbf{w}$ arbitrarily large, without changing the position boundary (noting that (4.5) is unaffected by scalar multiples of $\mathbf{w}$). It is, therefore, necessary to restrict the value of $\mathbf{w}$ either through constrained optimisation (eg. imposing $\mathbf{w}^T\mathbf{w} = 1$) or through the addition of penalty terms. The benefit of using the latter approach is that it can be used as a form of regularisation, as discussed below.

4.1.1.2. Regularisation

A sufficiently complex model can be forced to tightly fit the training data in a way that does not generalise well to observations outside the training set. This problem, known as overfitting, can be addressed through regularisation, which reduces how 'tightly' the decision boundary is formed around data points that lie close to the boundary. Regularisation can be applied to LR by adding a regularisation term $\frac{1}{2\lambda}\mathbf{w}^T\mathbf{w}$ to the negative log-likelihood, $E(\mathbf{w})$. This gives a new cost function

$$J(\mathbf{w}) = E(\mathbf{w}) + \frac{1}{2\lambda}\mathbf{w}^T\mathbf{w} \quad = -\sum_{n=1}^{N}\{y_n \ln\sigma(\mathbf{w}^T\mathbf{x}_n) + (1 - y_n)\ln(1 - \sigma(\mathbf{w}^T\mathbf{x}_n))\} - \ln p(X) \tag{4.10}$$

to be minimised. Note that the regularisation term added here introduces an additional parameter, λ. Hyperparameters such as this are optimised through the use of a validation set, as discussed in Section 4.4.

4.1.2. Implementation

Using the described theory, an LR classifier was implemented in MATLAB. Rather than implementing an optimisation algorithm from scratch, the BFGS (Broyden, Fletcher, Goldfarb, and Shanno [85–88]) Quasi-Newton method implemented in the Optimisation Toolbox was employed to minimise (4.10). This method requires an expression for the gradient, which is calculated as

$$\boldsymbol{\nabla} E(\mathbf{w}) = \sum_{n=1}^{N} (\sigma(\mathbf{w}^T \mathbf{x}_n) - y_n)\mathbf{x}_n. \tag{4.11}$$

Once the model parameters $\mathbf{w}$ have been set, predictions for new observations $\mathbf{x}_{new}$ are then calculated as,

$$y_{new} = \sigma(\mathbf{w}^T \mathbf{x}_{new}) \geq 0.5. \tag{4.12}$$

4.2. Support vector machine (SVM)

The support vector machine (SVM) classifier has a lot in common with LR, in that it is also a linear classifier which separates two classes with a hyper-plane. The first major difference between LR and SVM is how the optimal hyperplane is defined. While the optimal decision boundary according to LR is defined through a statistical approach, SVM defines the optimal decision boundary based on the geometrical properties of the data. More precisely, LR maximises the likelihood of the training data, while SVM maximises the margin between the decision boundary and the nearest observations (support vectors). Due to this property, SVM is termed a *large margin* classifier. A more in-depth explanation of SVM is provided below, with a formal definition of the optimal hyperplane (decision boundary) and details on the training procedure. This is followed by a discussion on how the *kernel* trick can be used to form a non-linear decision boundary in the original feature space by mapping the data to a new, higher-dimensional, feature space. The section then concludes with the implementation details of the SVM used in this study.

SVM is widely used in general image classification [89], and has also been applied to the classification of ISAR images in previous studies [90, 91]. The ability to create non-linear decision boundaries in the input feature space makes SVM an appropriate technique for the classification of the ISAR images used in this study.

4.2.1. Theory

Consider once again the training data $\mathcal{D} = \{\mathbf{x}_n, y_n\}$. For this section, it is helpful to assign the class labels such that $y_i = 1$ indicates class $\mathcal{C}_1$ (also referred to as the positive class) and $y_i = -1$ indicates class $\mathcal{C}_2$ (or the negative class). Any hyperplane in the feature space can be described by the set of points in x which satisfy the equation

$$\mathbf{w}^T\mathbf{x} - b = 0, \tag{4.13}$$

where $\mathbf{w}$ is a normal vector to the hyperplane and $\frac{b}{||\mathbf{w}||}$ gives the offset (in the direction of $\mathbf{w}$) of the hyperplane from the origin. Note that $\mathbf{w}$ need not necessarily be a unit vector in this definition. We will begin by formulating SVM for linearly separable data, before extending the concept to include data that is not linearly separable.

4.2.1.1. Formulation for linearly separable data

If we assume that the data is linearly separable, it follows that there exist (at least) two parallel hyperplanes that separate the two classes. These hyperplanes, shown in Figure 4.2, can be described by

$$\mathbf{w}^T\mathbf{x} - b = 1 \tag{4.14}$$

and

$$\mathbf{w}^T\mathbf{x} - b = -1 \tag{4.15}$$

where any point on or above (4.14) belongs to $\mathcal{C}_1$ and any point on or below (4.15) belongs to $\mathcal{C}_2$. The hyperplane that lies mid-way between these two planes is therefore described by

$$\mathbf{w}^T\mathbf{x} - b = 0, \tag{4.16}$$

The region between the bounding hyperplanes, known as the margin, can be calculated as

$$d = \frac{2}{||\mathbf{w}||}. \tag{4.17}$$

For SVM, the objective is to find two such planes that maximise the margin d. The hyperplane that lies mid-way between two hyperplanes with maximum separation is referred to as the *maximum-margin* hyperplane. From (4.17) it is apparent that in order to maximise the margin d, $||\mathbf{w}||$ should be minimised. This is done under the constraint

$$y_i\left(\mathbf{w}^T\mathbf{x} - b\right) \geq 1 \tag{4.18}$$

to ensure that the observations for each class remain on the correct side of the margin. Under this condition, this is known as a hard-margin SVM. It should be noted that the decision boundary found by SVM is dependent only on the observations that lie close to the decision boundary (on the hyperplanes defining the margin), and is completely independent of the rest of the data. The data points lying on the margin are known as *support vectors*.

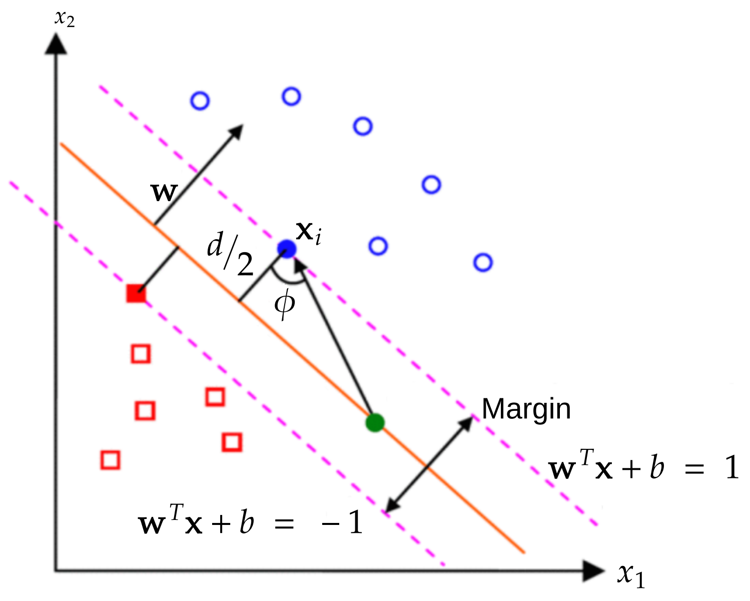

x_2
$\mathbf{w}$
$d/2$
$\mathbf{x}_i$
ϕ
Margin
$\mathbf{w}^T\mathbf{x}+b = 1$
$\mathbf{w}^T\mathbf{x}+b = -1$
x_1

Figure 4.2: Linearly separable classes (represented by the blue circles and red squares) separated by parallel hyperplanes (dashed lines). [92] ©2019 IEEE.

4.2.1.2. Extension to data that is not linearly separable

Previously, we made the assumption that the training data $\mathcal{D}$ is linearly separable. It is obvious, however, that this is not the general case. To extend the use of SVM to data that is not linearly separable, the *hinge loss* function

$$\max\left(0, 1 - y_i\left(\mathbf{w}^T\mathbf{x} - b\right)\right) \tag{4.19}$$

is employed. This function reduces to 0 in situations where (4.18) is satisfied but, for all observations on the wrong side of the margin, increases linearly with respect to the distance from the margin. It is therefore possible to create a 'soft' margin by relaxing the constraints in (4.18) and penalising choices of $\mathbf{w}$ with the *hinge loss* (4.19). The optimisation goal thus becomes minimising

$$\|\mathbf{w}\|^2 + C\left[\frac{1}{n}\sum_{i=1}^{N}\max\left(0, 1 - y_i\left(\mathbf{w}^T\mathbf{x} - b\right)\right)\right] \tag{4.20}$$

where the hyperparameter C can be adjusted to control the trade-off between the size of the margin and the extent to which the condition (4.18) is relaxed. Large values of C cause the SVM to behave similarly to the hard-margin SVM, while allowing it to work on data which are not linearly separable. Techniques used to approach the task of minimising (4.20) are discussed below.

4.2.1.3. Computation

Minimising (4.20)) can be approached using numerous optimisation techniques. Because the expression in (4.20) is a convex function in $\mathbf{w}$ and $\mathbf{b}$, gradient descent methods such as sub-gradient descent can be used [93]. The classic approach [94], however, which involves reducing (4.20) to a quadratic programming problem, is outlined here. To start off, let us introduce a variable $\mathbf{Z} = (\zeta_1, ..., \zeta_n)$ where $\zeta_i = \max\left(0, 1 - y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right)\right)$. With this, the optimisation problem can be rewritten to minimise

$$\|\mathbf{w}\|^2 + C\left[\frac{1}{n}\sum_{i=1}^{N}\zeta_i\right] \tag{4.21}$$

subject to

$$y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right) \geq 1 - \zeta_i \tag{4.22}$$

$$\zeta_i \geq 0,\ \forall i\,. \tag{4.23}$$

From this, we can construct a Lagrangian using Lagrangian multipliers $\mathbf{\Lambda} = (\alpha_1, ..., \alpha_n)$ and $\mathbf{R} = (r_1, ..., r_n)$ to enforce constraints (4.22) and (4.23), respectively,

$$L(\mathbf{w}, \mathbf{Z}, b, \mathbf{\Lambda}, \mathbf{R}) = \mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{N}\zeta_i - \sum_{i=1}^{N}\alpha_i\left[y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 + \zeta_i\right] - \sum_{i=1}^{N}r_i\zeta_i \tag{4.24}$$

The saddle point of $L(\mathbf{w}, \mathbf{Z}, b, \mathbf{\Lambda}, \mathbf{R})$ can be found where

$$\frac{\partial L}{\partial W} = 2\mathbf{w} - \frac{C}{n}\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i = 0 \tag{4.25}$$

$$\frac{\partial L}{\partial b} = \frac{C}{n}\sum_{i=1}^{N}\alpha_i y_i = 0 \tag{4.26}$$

and

$$\frac{\partial L}{\partial \zeta_i} = \frac{C}{n} - \alpha_i - r_i = 0\,. \tag{4.27}$$

Rearranging, we find that at this point

$$\mathbf{w} = \frac{C}{2n}\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i\,, \tag{4.28}$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0\,, \tag{4.29}$$

and

$$r_i = \frac{C}{n} - \alpha_i \,. \tag{4.30}$$

The solution to the original optimisation problem can therefore be found by substituting these expressions for $\mathbf{w}$, r_i and $\sum_{i=1}^{N} \alpha_i y_i$ into the Lagrangian (4.24) and maximising with respect to α_i:

$$\text{maximise } f(\alpha_1, ...\alpha_n) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i \alpha_i \left(\mathbf{x}_i^T \mathbf{x}_j \right) y_i \alpha_i \tag{4.31}$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0, \tag{4.32}$$

$$0 \leq \alpha_i \leq \frac{C}{n} \,. \tag{4.33}$$

This maximisation problem with linear constraints can be solved in α_i with quadratic programming algorithms or, alternatively, methods such as coordinate descent [95]. Notably, it can be shown that $\alpha = 0$ for any $\mathbf{x}_i$ on the correct side of the margin and $0 < \alpha_i \leq {}^{C}\!/_{n}$ when $\mathbf{x}_i$ lies on the margin boundary (i.e. $\mathbf{x}_i$ is a *support vector*). It has already been shown that $\mathbf{w}$ can be expressed as a linear combination of support vectors in (4.28). The hyperplane offset value, b, can be recovered by substituting the coordinates of a support vector $\mathbf{x}_i$ (choosing an index i such that $0 < \alpha_i \leq {}^{C}\!/_{n}$) into

$$\begin{aligned} y_i \left(\mathbf{w}^T \mathbf{x}_i - b \right) &= 1 \\ b &= \mathbf{w}^T \mathbf{x}_i - y_i \end{aligned} \tag{4.34}$$

noting that $y_i^{-1} = y_i$ since $y_i = \pm 1$. The fact that the maximisation problem obtained in (4.31) only requires the dot product of two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is noted here. The importance of this property is highlighted below, in the discussion of how the kernel trick is applied.

4.2.1.4. Non-linear kernels

In general, it is unreasonable to expect that ISAR images of different classes are linearly separable. This study, therefore, needs to consider non-linear classification algorithms. While an algorithm that separates two classes with a hyperplane is strictly a linear classifier, there is no reason why classification should be performed in the original feature space. It is possible that data which are not linearly separable in the original feature space $\mathcal{X}$ can be transformed to a new, often higher dimensional, feature space $\mathcal{V}$ in which the two classes are linearly separable via a non-linear mapping function $\varphi : \mathcal{X} \to \mathcal{V}$ such that

$$\mathbf{v}_i = \varphi\left(\mathbf{x}_i\right). \tag{4.35}$$

Figure 4.3 demonstrates this concept with an example, where the mapping $\mathbf{v} = (v_1, v_2, v_3) = (x_1, x_2, x_1^2 + x_2^2)$ is used to transform two class data from a feature space where the classes are linearly inseparable to one where the classes are linearly separable.

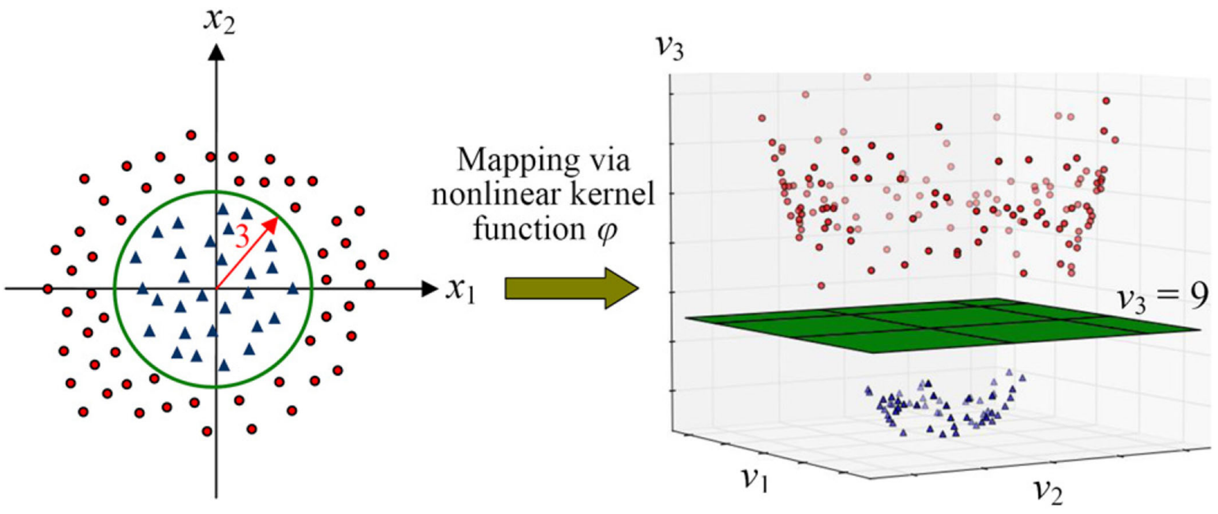


Figure 4.3: An example from [92] of how a non-linear mapping can be used to convert a linearly inseparable problem to one which is linearly separable in a higher dimensional feature space. ©2019 IEEE.

A direct approach to using the above concept would be to transform the entire dataset into the new feature space first, before using an SVM on the transformed dataset. Computing the coordinates in the new feature space for every data point is, however, computationally expensive. It is worth recognising that for the optimisation problem in (4.31), the transformed feature space $\mathcal{V}$ is only needed for computation of the dot product $\mathbf{x}_i \cdot \mathbf{x}_j$, and we are at no point directly interested in the full representation of the data in this space. The *kernel trick* [96] can therefore be utilised. A kernel function which maps the inner product of any two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ (in $\mathcal{X}$) in another feature space $\mathcal{V}$,

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left\langle \varphi\left(\mathbf{x}_i\right), \varphi\left(\mathbf{x}_j\right)\right\rangle_{\mathcal{V}} \tag{4.36}$$

can be used to perform the necessary calculations in $\mathcal{V}$ without the overhead requirement of transforming the entire dataset to the new feature space. In fact, an explicit definition of $\varphi(\cdot)$ is not necessary and it is even possible to use kernels that correspond to mappings to infinite dimensional feature spaces [96, p. 825]. Any function satisfying Mercer's condition [97] may be used as a kernel function, however, popular choices of kernel functions include [92]:

- Polynomial kernel: $k\left(\mathbf{x}_i \mathbf{x}_j\right) = \left(\mathbf{x}^T \mathbf{x}_j + a\right)^b$

- Gaussian radial basis function: $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$
- Hyperbolic tangent: $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \tanh\Big(a\mathbf{x}_i^T\mathbf{x}_j + b\Big)$

where a and b are scalar hyperparameters to be considered. Kernel functions can be used by replacing every inner product in the above linear SVM formulation with the desired kernel function. The optimisation task (4.31) therefore becomes

$$\text{maximise } f(\alpha_1, ...\alpha_n) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i\alpha_i\left(\varphi\left(\mathbf{x}_i\right)\cdot\varphi\left(\mathbf{x}_j\right)\right)y_i\alpha_i \tag{4.37}$$

$$= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i\alpha_i\left(k\left(\mathbf{x}_i, \mathbf{x}_j\right)\right)y_i\alpha_i \tag{4.38}$$

subject to

$$\sum_{i=1}^{N}\alpha_i y_i = 0, \tag{4.39}$$

$$0 \leq \alpha_i \leq \frac{C}{n}. \tag{4.40}$$

While it is known that $\mathbf{w}$ in the transformed space can be described by

$$\mathbf{w} = \frac{C}{2n}\sum_{i=1}^{N}\alpha_i y_i\varphi\left(\mathbf{x}_i\right), \tag{4.41}$$

finding $\mathbf{w}$ with this definition is problematic in the absence of an explicit mapping function $\varphi(\cdot)$ (as may be the case for certain choices of kernel function). Class predictions can, however, be computed without first explicitly finding $\mathbf{w}$, as is discussed below.

4.2.1.5. Prediction

Once the optimal hyperplane has been found, forming class predictions for a new observation $\mathbf{x}_{new}$ is as straightforward as determining which side of the decision boundary the unlabelled data lie. This can be written

$$y_{new} = sign\Big(\mathbf{w}^T\varphi\left(\mathbf{x}_{new}\right) - b\Big). \tag{4.42}$$

For implicit definitions of $\varphi(\cdot)$, however, we require an expression that does require $\mathbf{w}$ or the mapping $\varphi(\cdot)$. The offset parameter b can be calculated by finding some index i such that $0 < \alpha_i \leq {}^{C}/_{n}$, where it is known that $\varphi\left(\mathbf{x}_i\right)$ lies on the margin boundary, and solving

$$\begin{aligned} b &= \mathbf{w}^T \varphi(\mathbf{x}_i) - y_i \\ &= \left[\sum_{j=1}^{N} \alpha_j y_j \varphi(\mathbf{x}_j) \cdot \varphi(\mathbf{x}_i)\right] - y_i \\ &= \left[\sum_{j=1}^{N} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i)\right] - y_i \end{aligned} \tag{4.43}$$

The class prediction for a new observation $\mathbf{x}_{new}$ can then be calculated with

$$\begin{aligned} y_{new} &= sign\left(\mathbf{w}^T \mathbf{x}_{new} - b\right) \\ &= sign\left(\left[\sum_{j=1}^{N} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i)\right] - b\right). \end{aligned} \tag{4.44}$$

It is noted that SVM is not probabilistic, and is therefore only capable of forming discriminative class predictions and does not output a probability $P(y = 1|\mathbf{x})$. Platt scaling [98] can, however, be used for probabilistic classification using SVM.

4.2.2. Implementation

This study makes use of the SVM implementation used in the MATLAB *Statistics and Machine Learning Toolbox*. This implementation is straightforward to use, where the `mdl = fitcscvm(...)` function creates a model from the data and hyperparameters passed to it. The model can be queried with the `y_new = predict(x, mdl)` function to generate predictions (where `x` is an unlabelled ISAR image to be classified). Section 4.4 further explains how the dataset of ISAR images was used to train and test this algorithm.

4.3. K-nearest neighbours (KNN)

The third traditional machine learning classifier investigated, k-nearest neighbours, is a conceptually simple technique. In contrast to LR and SVM, the KNN algorithm is not a statistical model. The rationale behind the KNN algorithm is the assumption that observations of the same class are grouped together in the feature space. KNN, therefore, infers the class membership of new (unlabelled) observations by considering the distances to the k nearest labelled observations. Because inference is done using the labelled data directly rather than using a model of the data, KNN does not involve any trainable parameters that need to be optimised. Hence, there is no training step required. This property can be beneficial in that new labelled data can be added at any time to update, which allows it to be updated in real-time. The ability to be updated in real-time could be extremely advantageous for a target recognition system operating in a dynamic environment where the types of targets observed change over time. The major disadvantage, however, is that this comes at the cost of speed and computational efficiency. The run-time of the KNN algorithm scales poorly with both the size of the labelled dataset, as well as the dimensionality of the data. For target recognition purposes, faster prediction times are desirable, as this gives more time for the user of the system to react (particularly when the target is identified as a threat). Other studies have previously used KNN to recognise targets with ISAR imaging [90, 91]. The KNN algorithm is formalised below.

4.3.1. Theory

Consider a set of labelled data $\mathcal{D} = \mathbf{x}_n, y_n$ and a unlabelled data-point $\mathbf{x}_{new}$. The first step of the KNN algorithm involves calculating the distance, d_n from each labelled observation to the unlabelled observation. A variety of distance metrics may be employed (eg. Euclidian distance, Manhattan distance or Minkowski distance), however, this work specifically makes use of the Euclidean distance,

$$d_n = ||\mathbf{x}_{new} - \mathbf{x}_n|| \tag{4.45}$$

Using this metric, the k nearest labelled data points are selected to form a set of nearest neighbours $\mathcal{N}$, as shown in Figure 4.4. The unlabelled observation $\mathbf{x}_{new}$ is finally assigned to the modal class in the set $\mathcal{N}$.

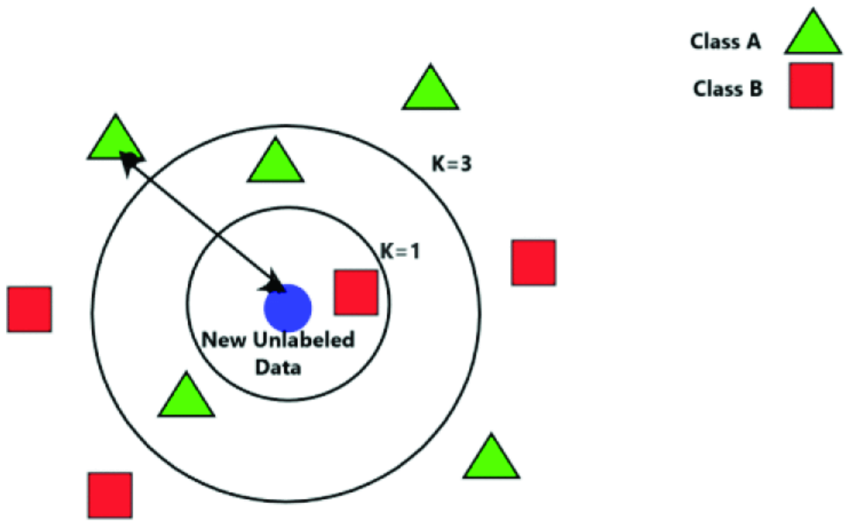


Figure 4.4: An unlabelled observation shown in relation to its nearest neighbours. Circled regions indicate different sets $\mathcal{N}$ for various values of k. Red and blue dots represent labelled observations and the respective colour of the highlighted regions indicates to which class a new observation in that region will be assigned. [99] ©2019 IEEE.

The number of neighbours to be considered, k, is a hyper-parameter that should be optimised appropriately (discussed in Section 4.4). It should be noted that, in general, increasing k reduces the effect of noise on the classification error, while smaller values of

k result in more complex decision boundaries [100]. Figure 4.5 shows how increasing k has a smoothing effect on the non-linear decision boundary. For a two-class problem, k is generally chosen as an odd number to avoid tied votes.

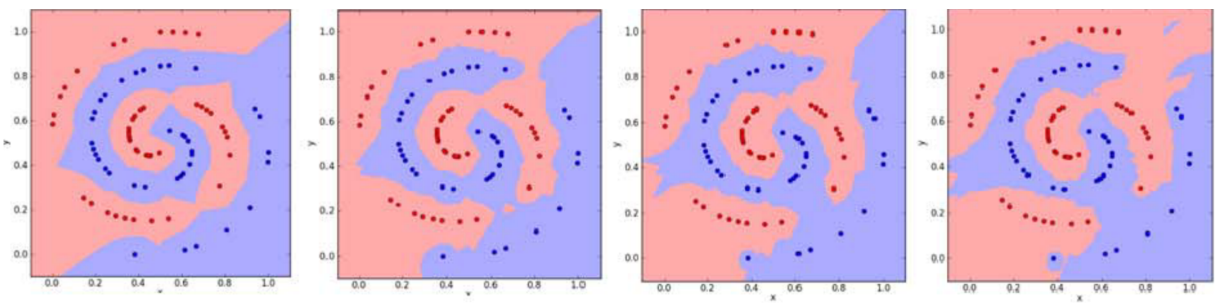


Figure 4.5: KNN decision boundary with (from left to right) $k = 1$, $k = 3$, $k = 5$ and $k = 7$ for an arbitrary dataset from [101]. ©2012 IEEE.

4.3.2. Implementation

Pseudocode for the KNN algorithm is given in Algorithm 4.1 which accepts a list of known observations `X_ref` with corresponding labels `y_ref` and an unlabelled observation `x_new`. The corresponding MATLAB implementation can be found in Appendix C.

Algorithm 4.1: Pseudo code for the KNN implementation used in this study.

```
for $x \in X\_ref$ do
    $d \leftarrow x - x\_new$
end for
$d\_sorted \leftarrow sort(d)$                        ▷ Sort in ascending order.
$y\_pred \leftarrow mode(d\_sorted[1, .., k])$
return $y\_pred$
```

4.4. Experiments

The three machine learning algorithms being investigated were compared under various training and testing conditions. Specifically, the conditions were adjusted to examine the effect of the variation between the test set and the training set (i.e. varying the imaging angles used), the effects of adjusting the SNR of the training data and the impact of upsampling the ISAR images. In this section, both the closed-set classification accuracy and the prediction times are assessed under these various conditions. Section 4.4.1 outlines the aspects of the investigation and training approach common across all the experiments. The sections following this discuss the specific details of each experiment and report on the results.

4.4.1. General approach

All of the experiments in this chapter implement binary classifiers trained on two of the five classes from the dataset. In all instances, these two classes are the multirotor drone and missile classes. Each ISAR image is represented by a matrix of complex values. The classification algorithms used here, however, require the input to be a real-valued feature vector for each observation. In open literature, there is a constant debate around using raw radar data or using some preprocessing and feature extraction. In this study the raw pixel of the ISAR images are used for the following reasons:

1. The use of handcrafted features is tedious and challenging, and it is, therefore, desirable to avoid this to reduce design overhead.

2. The results from the traditional machine learning classifiers are meant to form a baseline with which the deep learning techniques can be compared. For a fair comparison, we wish to apply the same pre-processing steps (deep learning techniques are designed to work on the raw data).

3. The resolution of the images is low enough (27px × 21px) that the use of the raw pixel data is still a viable option.

To transform the complex-valued matrices into real-valued feature vectors, the matrices were flattened and the complex values were split into their real and imaginary components (resulting in two features per complex value). This results in a vector with $27 \times 21 \times 2 = 1134$ features for each observation (i.e. a single ISAR image), which can be used as inputs for the classification algorithms.

4.4.2. Training at a single elevation

The first experiment's objective was twofold:

1. To get a performance comparison between the classification algorithms used on a simplified problem.

2. To evaluate each classifier's ability to generalise to observations from imaging angles outside of what they are trained on.

For this purpose, a subset of the dataset, containing ISAR images generated from a single elevation angle $\theta = 0°$ (illustrated in Figure 4.6) was used. Since only one elevation is considered, this subset displays significantly less variance than the full dataset, simplifying the classification problem. Of the images generated at this elevation, 70% of the images (corresponding to 70% of the azimuth angles used) were used for training, while the remainder was set aside for testing. Note that for this experiment, only ISAR images

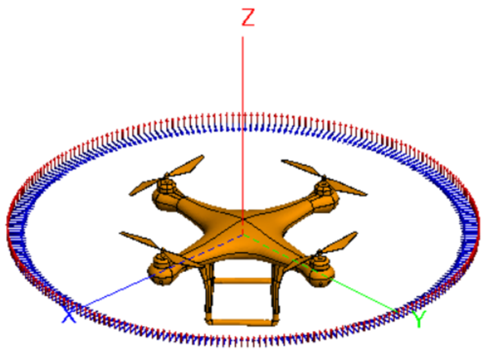


Figure 4.6: Imaging angles at a single elevation ($\theta = 0$).

without added noise were used for training. The LR, SVM and KNN classifiers were then trained using this reduced training set.

4.4.2.1. Hyperparameter tuning

A small validation set comprised of ISAR images with no added noise was used to tune the hyperparameters associated with each algorithm. The optimal hyperparameters λ and k for the LR and KNN classifiers, respectively, were determined empirically by plotting the accuracy obtained with the validation set against the hyperparameter value. The MATLAB implementation of SVM conveniently automates the optimisation of the numerous SVM hyperparameters using a grid-search algorithm. Table 4.1 summarises the hyperparameter choices used for testing.

Table 4.1: Hyperparameter values.

LR:	λ	0.001
KNN:	k	1
SVM:	C (`BoxConstraint` in MATLAB implementation)	930.66
	kernel function	Gaussian
	kernel scale	2.75

4.4.2.2. Results: Testing at a single elevation

Once trained, the models were tested on the ISAR images generated from the same elevation angle, that were previously set aside (excluded from training). Figure 4.7 plots the test accuracy against the SNR for each of the classifiers. From these results, it can be seen that all three classifiers achieve reasonable classification accuracy at high SNRs. The KNN classifier achieves a classification accuracy of 100 % on this test set for SNRs as low as −3 dB, after which the classification accuracy begins to decline. The SVM classifier has a slightly poorer accuracy than the KNN classifier, maintaining a 98-99 % classification accuracy for SNRs above 6 dB. It can be observed, however, that the SVM classifier is more susceptible to image quality degradation as a result of noise as the classification accuracy declines below an SNR of 6 dB. Of the three algorithms, LR performs notably

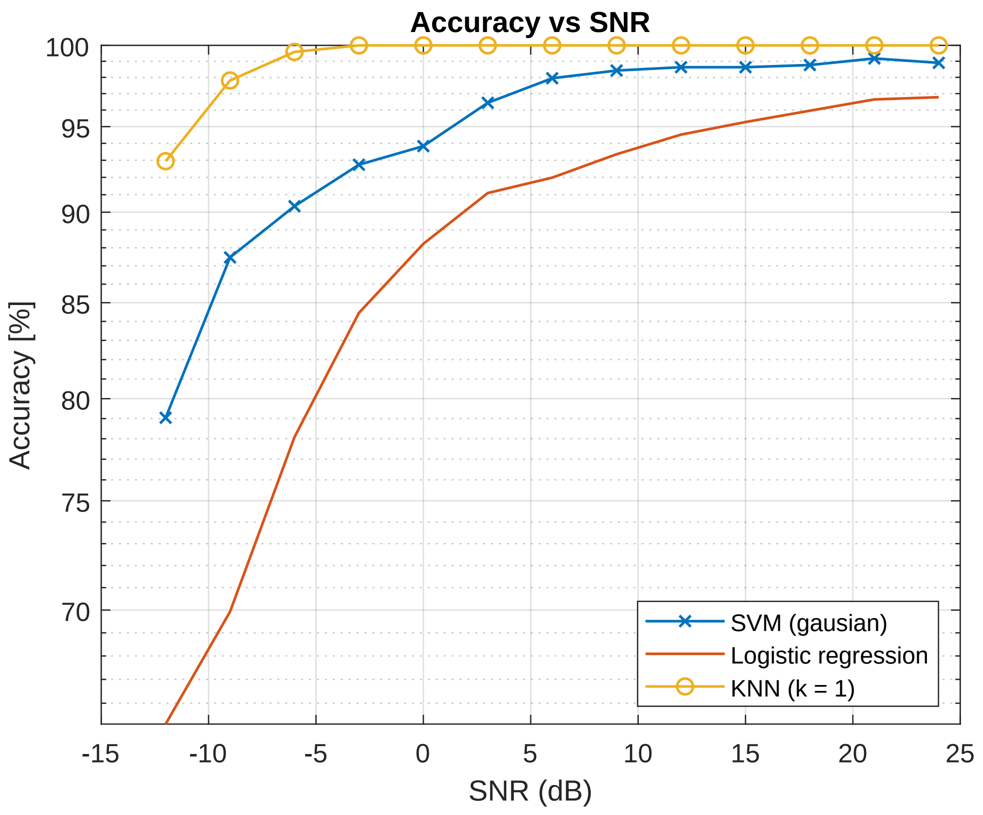


Figure 4.7: Accuracy vs SNR for the LR, KNN and SVM classifiers trained and tested on a single elevation. All classifiers achieve reasonable accuracy for high SNRs, which then drops off as the SNR decreases. The KNN classifier performs significantly better than the other two methods.

worse (with regard to accuracy). This poor performance can be attributed to the fact that the data is not linearly separable in the input feature space. The LR classifier lacks the complexity to properly fit the training data.

4.4.2.3. Results: Testing at multiple elevations

In many situations, designers of a target recognition system have limited amounts of training data. It is, therefore, relevant to investigate each classifier's ability to handle observations of targets imaged from aspect angles significantly different to those used in training. Each classifier, trained at a single elevation (0°), was tested on a set of images generated for multiple elevation angles to gauge the classifiers' ability to generalise to novel observations. The multiple-elevation test set contained ISAR images of the targets imaged at elevation angles from −45° to 45°. To provide some indication of the variation between the training and test sets, Figure 4.8 shows an example of an ISAR image of the drone target imaged at an elevation of 0° (used in the training set), while Figure 4.9 shows ISAR images of the same target at the same azimuth angle but at different elevations (used in the test set).

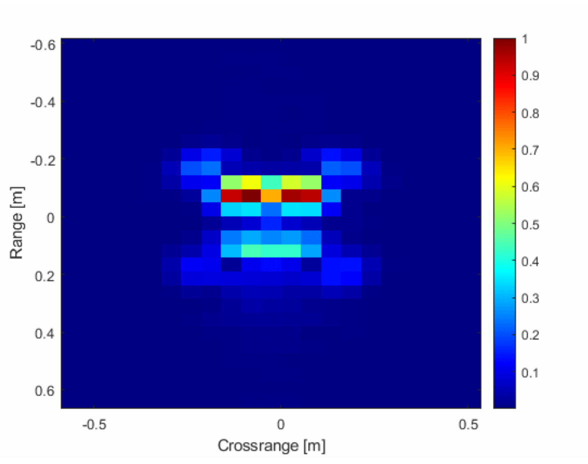
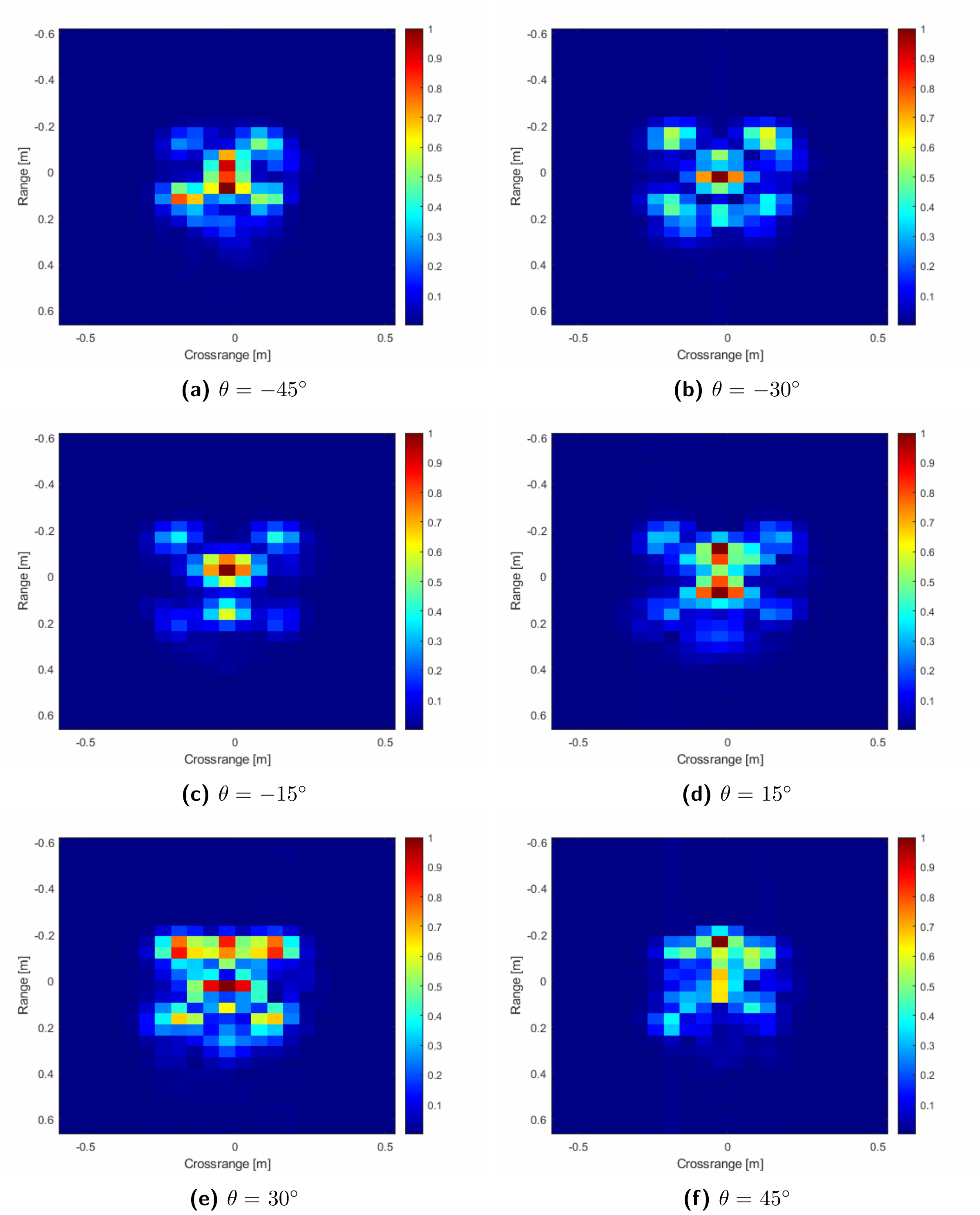


Figure 4.8: An ISAR image of the drone target imaged at an elevation angle $\theta = 0°$ and azimuth angle $\phi = 0°$.

The accuracy obtained on this test set is plotted against SNR in Figure 4.10. Both the LR and SVM classifiers are observed to generalise poorly to ISAR images generated from novel aspect angles outside of the training data's range, achieving a maximum classification accuracy of 63 % and 78 %, respectively. On the other hand, the KNN classifier continues to perfectly classify the test set for SNRs above 18 dB and maintains a classification accuracy >99 % for SNRs above 3 dB, despite the significant variation between the training and test sets. The most observable difference in the KNN algorithm's performance between the single- and multiple-elevation test sets is that when tasked with classifying data outside the training domain (with regard to aspect angle), the KNN classifier is slightly more sensitive to noise.



(a) $\theta = -45°$

(b) $\theta = -30°$

(c) $\theta = -15°$

(d) $\theta = 15°$

(e) $\theta = 30°$

(f) $\theta = 45°$

Figure 4.9: ISAR images of the drone target imaged at multiple elevation angles θ (given in each sub-figure) and azimuth angle $\phi = 0°$

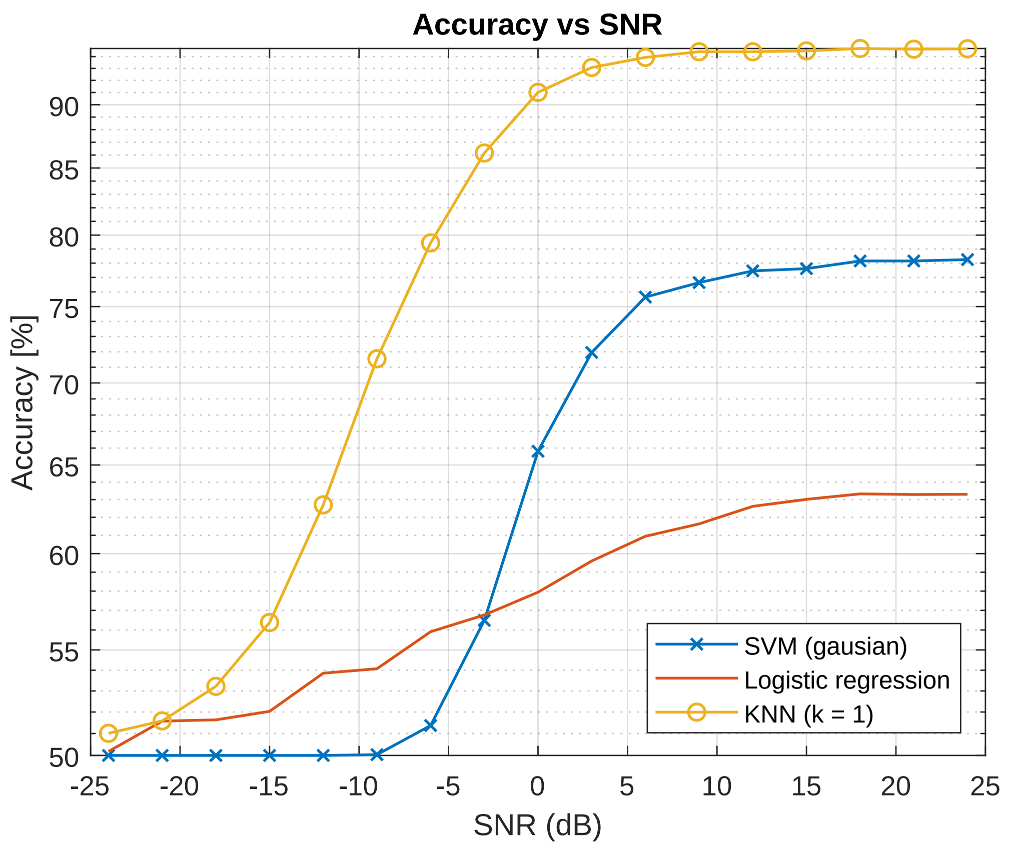


Figure 4.10: Accuracy vs SNR for the LR, KNN and SVM classifiers trained on a single elevation and tested on multiple elevations. Here it can be seen that the KNN handles the out-of-distribution observations significantly better than the other two classifiers which are unable to effectively classify ISAR images from novel imaging angles outside of those observed in training.

4.4.3. Training at multiple elevations

Following the experiment where the classifiers were trained with data from a single elevation, the classifiers were retrained using data from multiple elevation angles (45° above the horizontal to 45° below the horizontal) and all azimuth angles, as shown for the drone target in Figure 4.11. Again, the training data only included ISAR images with no added noise, with a 70:30 split between the training and test set.

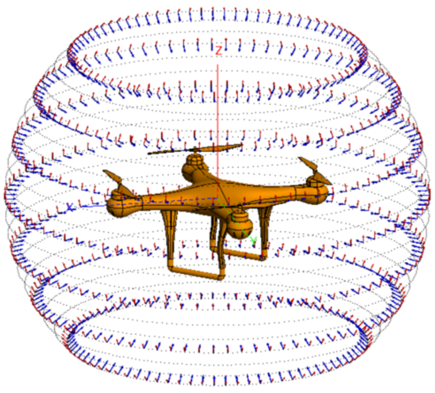


Figure 4.11: Imaging angles for multiple elevations ($-45 \leq \theta \leq 45$).

4.4.3.1. Results

The new models were tested on the full test set, containing ISAR images of the two target models imaged at multiple elevation and azimuth angles (i.e. the same test set as used in Section 4.4.2) above. The results are plotted in Figure 4.12. Comparing the results obtained with the new models in Figure 4.12 to the results from the old model trained on a single elevation (Figure 4.10), it can be seen that the SVM and KNN classifiers benefit significantly from the increased training set. The range of SNRs over which the KNN classifier is capable of perfect classification extends down to 0 dB, and the accuracy only begins to drop significantly from −9 dB. The increased performance of the KNN classifier can be attributed to the increase in labelled observations to which new observations are compared.

While the SVM classifier trained on a single elevation achieved a maximum classification accuracy of 78 %, training with multiple elevation angles enabled it to achieve a classification accuracy between 98 and 99 % for high and moderate SNRs (>0 dB). Interestingly, in this SNR range, the SVM classifier shows better performance when trained and tested on multiple elevations (Figure 4.12) than when trained and tested on a single elevation (Figure 4.7), despite the fact that the latter classification problem is a subset of the former. This is an interesting consequence of the use of the kernel trick. Increasing the number of observations in the training set effectively increases the rank (number of dimensions used) of the data in the mapped feature space. As a result, increasing the number of

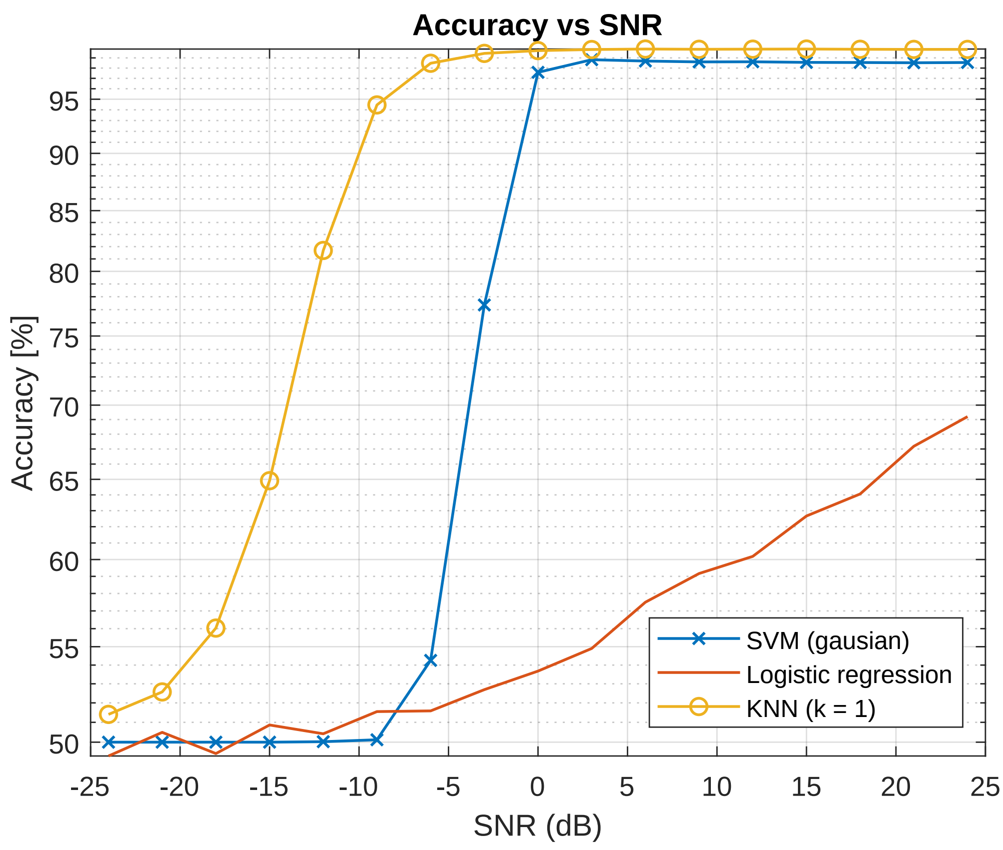


Figure 4.12: Accuracy vs SNR for the LR, KNN and SVM classifiers trained and tested at multiple elevations. While the KNN and SVM algorithms manage to fit the data well, the logistic regression classifier is unable to fit the data when ISAR images from multiple elevation angles are used.

training observations increases the SVM's ability to map the data to a feature space in which the classes are linearly separable. It is worth noting, that the SVM classifier was able to perfectly fit the training data (which is only possible for data which is linearly separable in the mapped feature space). For SNRs below 0 dB, the SVM classifier displays high sensitivity to noise.

In contrast, the performance of the LR classifier only shows a marginal increase when trained with multiple elevations. This is unsurprising given that the data is not linearly separable in the input feature space. In this instance, once the optimal hyperplane has been found, an increase in training data does not contribute anything to the model. The data remains linearly inseparable, and the LR classifier lacks the complexity to form a non-linear decision boundary.

4.4.3.2. Runtimes

For a target recognition system, while classification accuracy is important, this is not the only performance metric to consider. For a system performing real-time classification, the prediction speed is also of interest. Note that the training time is less important since this is a once-off cost that doesn't affect the normal operation of a target recognition system. Table 4.2 reports the prediction times for each of the classifiers and captures the impact of the training set size on the prediction speed. All computations were performed on an Intel i7-7700HQ CPU.

Table 4.2: Prediction times for KNN, SVM and LR classifiers.

Classifier	**Average prediction time for a single observation [ms].**	
	Single elevation (340 training observations)	**19 elevations (6460 training observations)**
LR	0.0365	0.0376
SVM	0.0382	1.3000
KNN	2.5000	48.8000

From these results, it is observed that the KNN classifier has significantly longer prediction times than the other two classifiers. Additionally, this prediction time increases linearly with the size of the labelled dataset. Despite the KNN classifier's excellent classification accuracy, in the case of large datasets the prohibitively long prediction times limit its suitability to real-time automatic target recognition systems. The LR and SVM classifiers are shown to form predictions significantly faster. In the case of the LR classifier, this time is constant with respect to the size of the training set. While the SVM classifier prediction time remains low in these examples, it should be noted that the prediction speed scales poorly with the number of *support vectors* (and hence the size of the training set).

4.4.4. Training at other SNRs

In the previous experiments, it is noted that the training sets only contained ISAR images with no added noise. In an attempt to make the classifiers less sensitive to noise, the effect of including examples of noisy images in the training data was investigated. The models were retrained on multiple training sets comprised of ISAR images of the two targets at all recorded aspect angles (azimuth and elevation), with each training set using images with a different SNR (in the range -24-24 dB). For each combination of aspect angle and SNR, three noise samples were used.

4.4.4.1. Results

The classifiers were tested on the same test set used previously (i.e. the 30 % split not used in training). Figures 4.13 and 4.14 depict the results for the LR and SVM classifiers, respectively. The KNN classifier was not included in this investigation due to its lengthy prediction times (we have also already seen good performance from this classifier for SNRs as low as -6 dB).

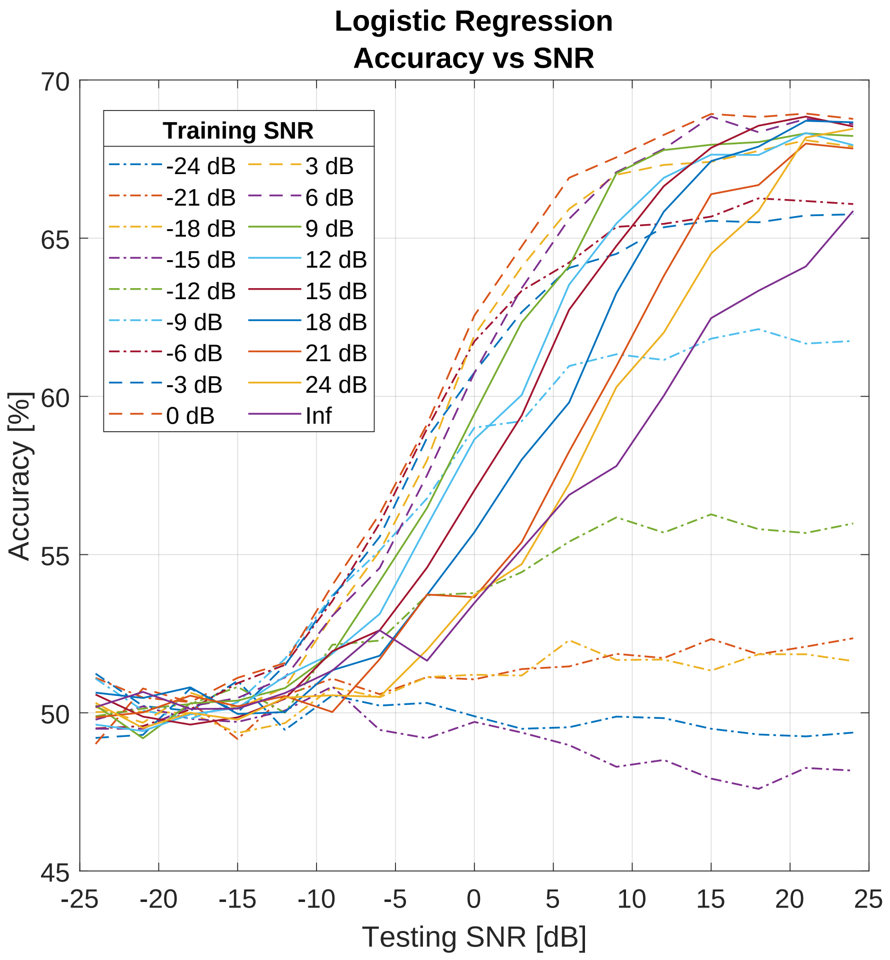


Figure 4.13: Accuracy vs SNR for the LR classifier trained at various SNRs. Adding a small amount of noise to the training data improves the noise sensitivity of the classifier, while too much noise degrades performance.

Looking at Figure 4.13, it can be seen that while training the LR classifier at lower SNRs (in the range 24–0 dB) does not have a significant effect on the accuracy when tested on the test set with an SNR of 24 dB. When tested at lower SNRs, however, it is observed that the addition of noise to the training data improves the classifier's sensitivity to noise. The general trend is that for training SNRs between 0 and 24 dB, as the SNR of the training set decreases, the classifier's ability to accurately classify targets at lower SNRs increases. Training with SNRs below 0 dB, however, has a detrimental effect and reduces the classification accuracy overall. At training SNRs below -18 dB, the quality of the training data decreases to the point where the LR algorithm cannot separate the classes and the classification accuracy drops to approximately 50 % (i.e. as good as a random guess).

For the SVM classifier, Figure 4.14 shows that for training SNRs between 24 and 0 dB, the addition of noise results in a slight decrease in the maximum accuracy obtained when the test set SNR is above 3 dB. As the training SNR approaches 0 dB, however, the SVM

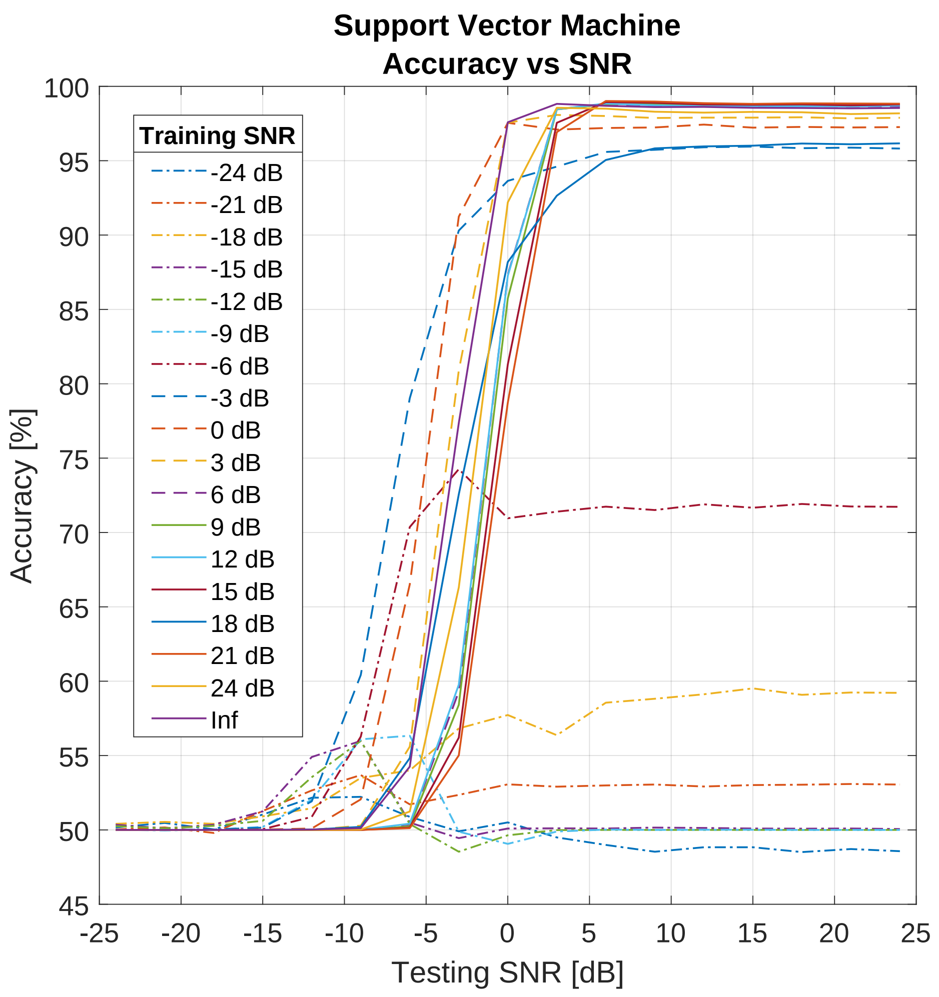


Figure 4.14: Accuracy vs SNR for the SVM classifier trained at various SNRs. Adding a small amount of noise to the training data improves the noise sensitivity of the classifier despite decreasing the performance at very high SNRs. The addition of too much noise degrades performance overall.

classifier remains accurate over a larger range of testing SNRs. For training SNRs below -3 dB, the accuracy of the SVM classifier drops significantly as the training SNR decreases. Despite these trends, the SVM classifier trained without any added noise in the training data appears to result in the best overall performance.

4.4.5. Up-sampling

For ISAR images presented to human operators, up-sampling is used often to provide a smoother interpolated image in which it is easier to see the detail of the target. Examples of this are shown in Figure 4.15. While higher-resolution images are easier for a human to interpret, up-sampling does not add any more information to the image. Additionally, using a higher image resolution is computationally expensive when it comes to training a machine learning algorithm (ML) on thousands of such ISAR images. This poses the question of whether up-sampling is beneficial to machine learning algorithms.

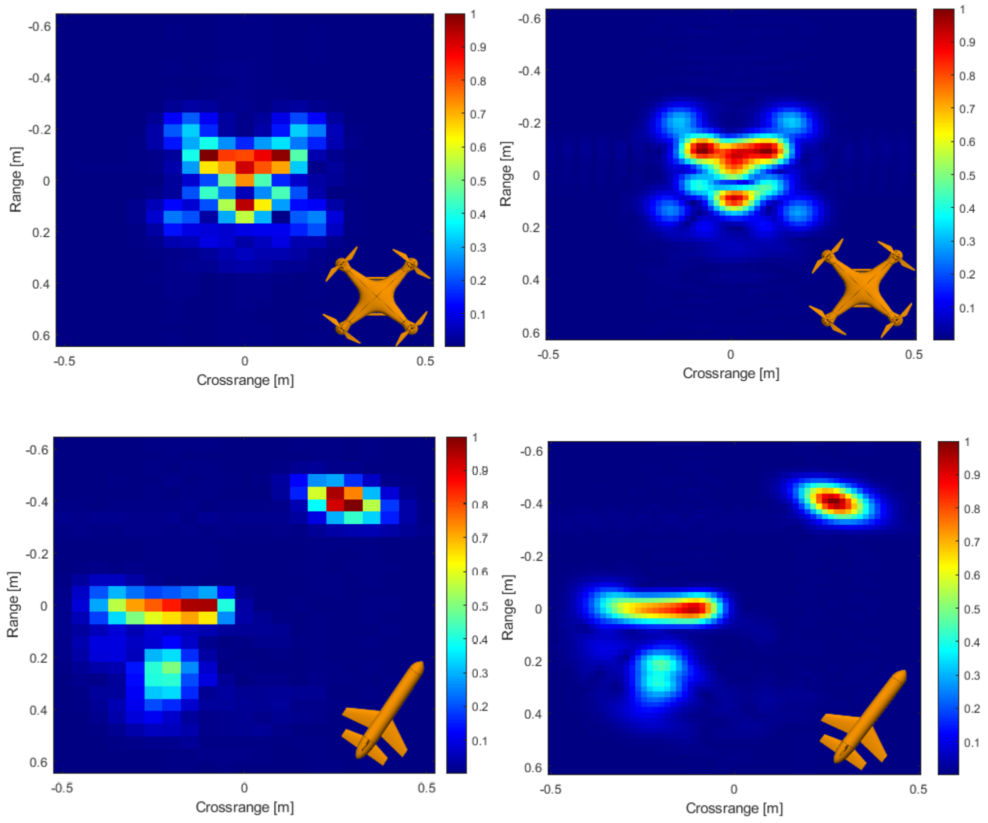


Figure 4.15: Images (a) and (c) show example synthetic ISAR images generated from the drone and missile models respectively. Images (b) and (d) show the same images up-sampled with a 64-point FFT.

This experiment investigates the effects of using up-sampling (through FFT interpolation) as a pre-processing step. This is done by training and testing the classifiers on ISAR images that have been up-sampled with a 64-point FFT. These results can be compared to those obtained in Section 4.4.4, which uses the same testing and training data, but without the up-sampling. To examine the effects, both the classification accuracy and prediction times are considered. Again, the KNN classifier was excluded from this investigation due to its prohibitively slow prediction times.

4.4.5.1. Results

The results obtained using the up-sampled ISAR images with the LR classifier are shown in Fig. 4.16. The same general trends for performance versus training SNR (as with the previous results in Figure 4.13) can be seen, where training with a moderate SNR (0 dB to 6 dB) yields the best performance. It is interesting to note that the use of up-sampled ISAR images has a negative effect on the accuracy of the LR classifier at moderate to high SNRs, however, it can be noted that the use of up-sampled ISAR images improves the performance of classifiers trained at low training SNRs (dash-dotted lines).

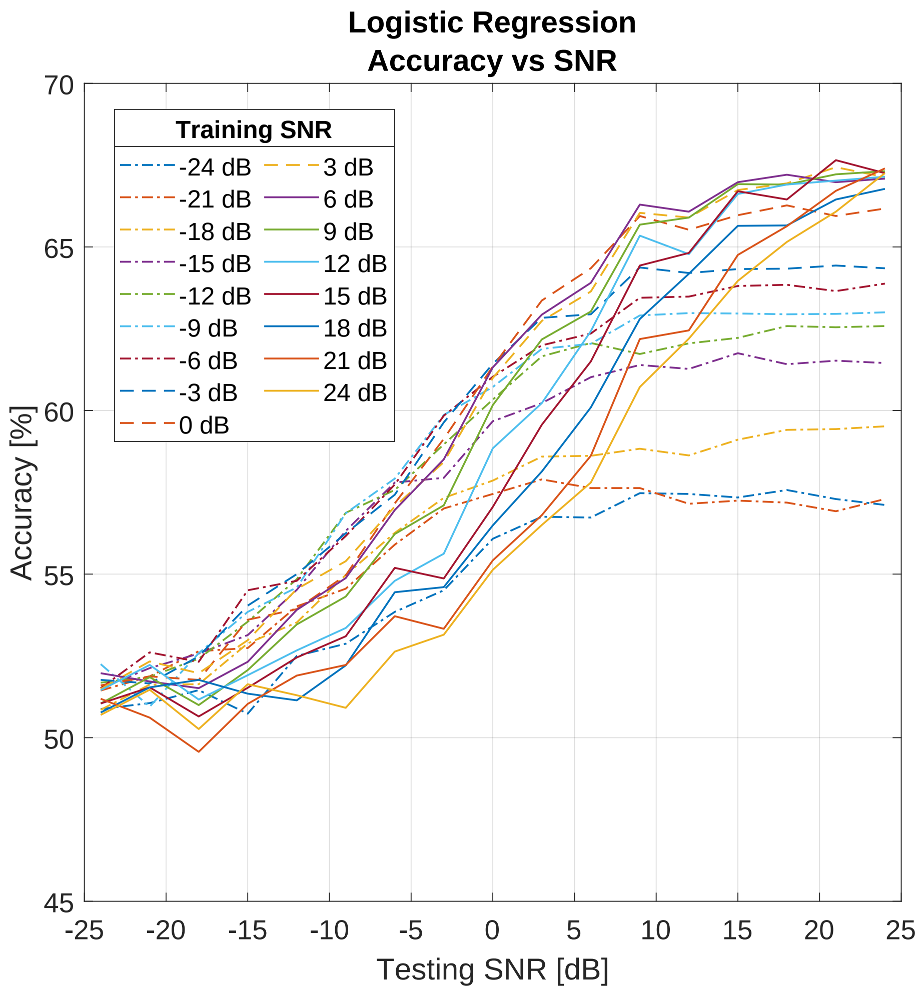


Figure 4.16: Accuracy vs SNR for the LR classifier trained at various SNRs on up-sampled ISAR images. This has a negative impact compared to the use of data which has not been up-sampled.

As seen in Fig. 4.17, the maximum accuracy of the SVM trained with up-sampled ISAR images is less than 70 %. This shows a significantly degraded performance when compared to images obtained without using up-sampling. One possible explanation is that the SVM hyper-parameters (such as the number of support vectors) were optimised with respect to both accuracy and training/prediction time. Owing to the fact that training with the up-sampled ISAR images is significantly more computationally expensive, accuracy was forfeited in favour of viable training and prediction times.

4.4.5.2. Prediction times

Table 4.3 shows the average prediction times for a single observation for the LR and SVM classifiers. In each case, the prediction time increases when the ISAR images are up-sampled, as this introduces more input features. The LR classifier has significantly faster prediction times (below 10 µs), that scale reasonably when up-sampled ISAR images

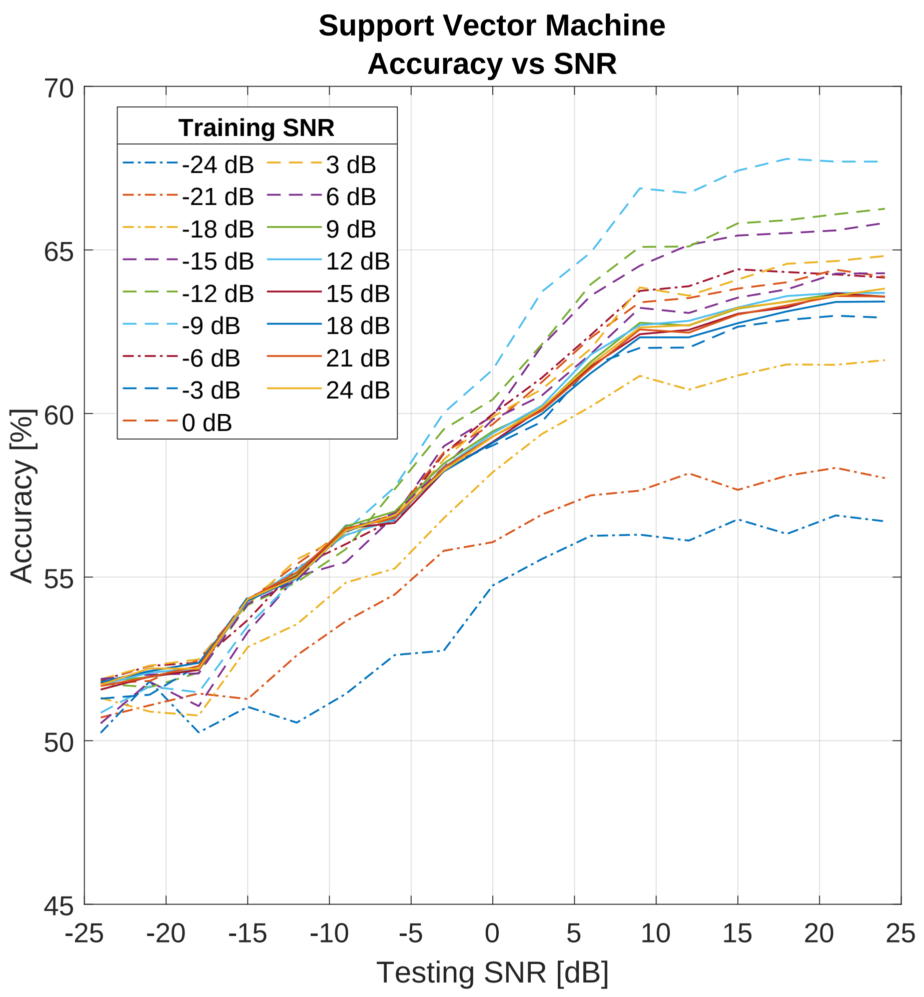


Figure 4.17: Accuracy vs SNR for the SVM classifier trained at various SNRs on up-sampled ISAR images. This has a significant negative impact compared to the use of data which has not been up-sampled.

are used. The SVM classifier is, however, a few orders of magnitude slower. Moreover, the SVM prediction times scale poorly with the number of input features. This is a severe limitation for the SVM classifier, as it cannot form predictions in real time under these conditions.

Table 4.3: Average prediction times per observation for the LR and SVM classifiers.

Algorithm	**Average prediction time [ms]**	
	No up-sampling	**With up-sampling**
LR	0.0056	0.0606
SVM	11.535	3363.5

4.4.6. Conclusion

Three traditional machine learning approaches were tested on a binary classification problem, with the goal of automatically discerning between two classes given an ISAR image of the target in question. Despite having two significantly different classes, the raw pixel data presents a linearly inseparable problem. As a result, it was observed that the LR classifier, which forms a linear decision boundary, was incapable of forming accurate class predictions given a dataset of images generated from significantly different aspect angles.

The KNN and SVM classifiers, which are capable of forming more complex non-linear decision boundaries both achieved a much higher classification accuracy. While the best SVM model maintained a classification accuracy of >97 % when classifying observations with an SNR ≥0 dB, the KNN was able to perfectly classify the test set in the same SNR range. Moreover, the KNN classifier continued to form class predictions with an accuracy above 95 % for SNRs as low as −6 dB. It was noted that the inclusion of noisy training examples (with a high to moderate SNR in the range 0–24 dB) generally improves the classifiers' sensitivity to noise without significantly impacting the accuracy at high SNRs.

While the KNN classifier clearly performed best with regard to classification accuracy, the KNN algorithm was many magnitudes slower than the others even on this relatively small dataset. With larger datasets, KNN is too slow to be used for real-time target recognition.

Finally, it was found that the use of FFT interpolation to form smoother images does not benefit the selected traditional machine learning algorithms. Rather, the resulting increase in input dimensions has a detrimental effect on the prediction speed, particularly for the SVM classifier.

While the KNN and SVM classifiers perform reasonably well, the fact that these classifiers do not estimate the likelihood of their predictions is problematic when considering open-set recognition. Open-set recognition requires some measure of the confidence of the classifier in its prediction. The next chapter focuses on machine learning algorithms that are capable of not only forming accurate class predictions but also estimating their confidence in their predictions, with an emphasis on open-set classification.

Chapter 5

Deep Learning Approaches

Deep learning techniques, which make use of artificial neural networks (ANN), are commonly applied to image recognition problems. The basic unit of an ANN, an artificial neuron (discussed further in Section 5.1), mimics a biological neuron which outputs an impulse (value) related through some non-linear mechanism to its input. The output of one neuron can be directed toward the input or another, creating a more complex non-linear function.

Taking inspiration from the structure of the brain, ANNs are comprised of multiple layers of artificial neurons. These models have a number of applications, including image recognition tasks. While deep learning techniques typically take longer to train and require more training data than traditional machine learning techniques, their modular structure can be used to build large networks capable of fitting more complex data. In addition to this, ANNs can have relatively fast prediction times (particularly when accelerated using modern GPUs).

This chapter investigates the application of two different types of ANN to ISAR image recognition. The first network is a fully-connected feed-forward network (FCNN) (Section 5.2), which, being arguably the simplest type of neural network, demonstrates the effectiveness of deep learning approaches to ISAR image classification [102]. The second network, a capsule network (CapsNet) [51, 68], was chosen to specifically address some of the challenges associated with the classification of ISAR images (in particular the challenges arising from the variation resulting from imaging targets imaged from different aspect angles). Section 5.4 discusses, in-depth, how capsule networks are well-suited to classifying ISAR images with a significant intra-class variation.

In addition to the above, this chapter also considers how these networks can be adapted to perform open-set classification. The ability of classification systems to appropriately handle observations from classes not included in the training data is often overlooked. However, in the case of automatic target recognition systems, this could prove very useful where the types of targets observed cannot exclusively be determined a priori.

5.1. The artificial neuron

The structure of an artificial neuron, which accepts a finite number N inputs to produce a single output, is shown in Figure 5.1. Mathematically, an artificial neuron is modelled by applying some non-linear activation function $a(z)$ to the weighted sum of its inputs x_1–x_N and a bias term b [103]. The weights w_1–w_N and the bias term b (often represented as w_0 applied to a unit input) can be adjusted to influence how each input affects the output. This can be expressed as

$$y = a\left(\mathbf{wx}\right). \tag{5.1}$$

Typical choices for the activation function $a(z)$ include:

- The sigmoid function,
$$a(z) = \frac{1}{1+e^{-z}}, \tag{5.2}$$
- The hyperbolic tan function,
$$a(z) = \tanh(z) = \frac{2}{1+e^{-2z}} - 1, \tag{5.3}$$
- The rectified linear unit (ReLU),
$$a(z) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0. \end{cases} \tag{5.4}$$
- The softmax function,
$$a(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{5.5}$$

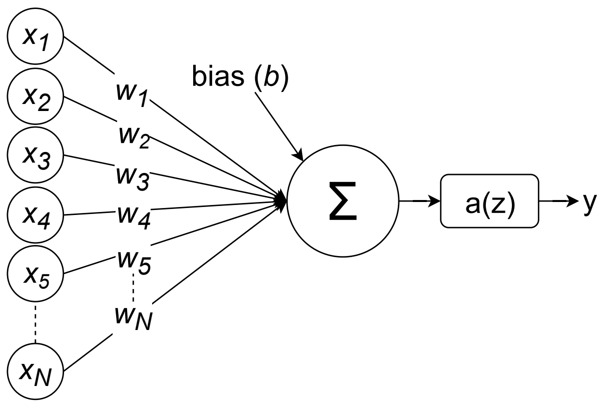


Figure 5.1: The structure of an artificial neuron. The output y is found by applying some non-linear function to the weighted sum of the inputs x_1 to x_N.

As discussed in the following sections, neural networks can be constructed by connecting multiple neurons to one another.

5.2. Fully-connected neural network (FCNN)

In the context of neural networks, two layers which are connected in such a way that the inputs to each neuron in one layer are connected to the outputs of every neuron in the previous layer are termed *fully-connected* layers [104]. A fully-connected neural network, also often referred to as a multi-layer perceptron (MLP), is therefore comprised of a series of fully connected layers. An example (and the architecture used in this study) is given in Figure 5.2. The layers of an ANN include a single input layer, a single output layer and one or more *hidden* layers between the input and output layers. The number of neurons in the input and output layers corresponds to the number of features and the number of classes, respectively. The number and size of the hidden layers can be adjusted and tuned through experimentation to find suitable values. In general, the more neurons per hidden layer, and the more layers in the network, the more complex the network becomes (enabling it to perform more challenging tasks). The benefits of a larger network are generally offset by the amount of time and training data required to train the model and the potential for overfitting as the model's complexity increases. A single hidden layer is sufficient to approximate any function that contains a continuous mapping from one finite space to another [105]. Furthermore, according to [105], a network with two hidden layers is theoretically capable of representing functions of any kind.

Once a model architecture has been decided, the training process is used to find the optimal weights and biases. In training, the weights are generally adjusted using the back-propagation algorithm [106]. While the back-propagation algorithm is not discussed in-depth here, it is an efficient implementation of the chain rule [107], and calculates the gradient of the loss function with respect to the weights one layer at a time, moving backwards through the network.

5.2.1. Architecture and implementation

In this study, a very basic FCNN with one hidden layer, given in Figure 5.2, is first used to perform binary classification. The size of the input layer matches the number of input features for a flattened $27\times\ 21\,\text{px}$ image with 2 channels (i.e. for the real and imaginary components of an ISAR image), and the output layer contains a neuron for each of the 2 classes. The number of neurons in the hidden layer was chosen as two-thirds of the size of the input layer, based on the guidelines given in [105]. For the hidden layer, the neurons use a ReLU activation function, while the output layers apply a softmax activation to ensure that the outputs are in the range $0 < y_k < 1$ and sum to 1 (over all the outputs).

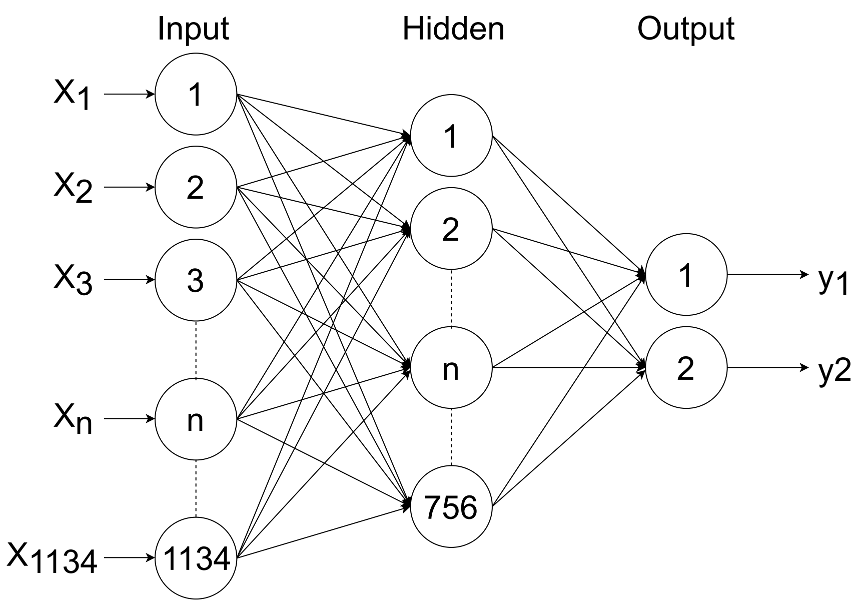


Figure 5.2: Diagram of the FCNN used in this work. The single hidden layer contains 756 neurones.

The network described above was implemented with Tensorflow 2.0 [108]. Listing 5.1 shows a snippet of the code used to define and train the network.

```python
import tensorflow as tf
model = tf.keras.models.Sequential([
tf.keras.layers.Flatten(input_shape=(21, 27, 2)),
tf.keras.layers.Dense(756, activation='relu'),
tf.keras.layers.Dense(2)
])

loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True
    )

model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=5, validation_data=(X_val, y_val))
```

Listing 5.1: Code snippet to define and train a FCNN with Tensorflow 2.0.

5.2.2. Training

The training set described in Section 4.4.3, comprised of ISAR images of the drone and missile targets imaged at various elevation and azimuth angles with no added noise, was used to train the FCNN. After training, the model was capable of perfectly classifying the training set (i.e. 100 % classification accuracy).

5.2.3. Results (closed-set)

The results obtained when testing on a (closed-set) test set comprised of the drone and missile ISAR images not used in training are presented below. Figure 5.3 shows the accuracy of the FCNN when tested on this test set at various SNRs, while Table 5.1 lists the average prediction times for a single observation for the FCNN classifier as well as the KNN and SVM classifiers from Chapter 4 (using the same training and test sets).

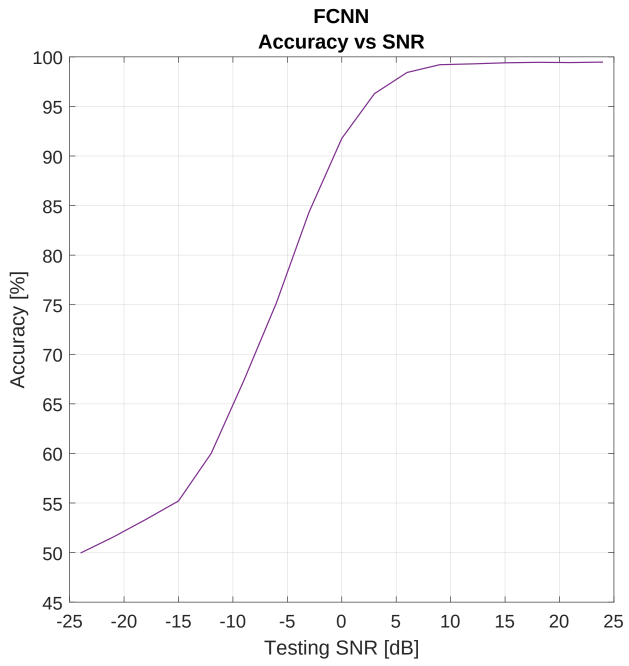


Figure 5.3: Accuracy vs SNR for the FCNN classifier tested on a closed-set.

Table 5.1: Prediction times for FCNN, KNN, SVM classifiers.

Classifier	Average prediction time [ms].
FCNN	0.0445
SVM	1.3000
KNN	48.8000

It can be seen that the accuracy of the FCNN classifier is comparable to the KNN and SVM classifiers discussed in Chapter 4. It maintains a classification accuracy of >99 % for testing SNRs above 9 dB, after which the accuracy decreases with the SNR of the test set. When it comes to the prediction times, however, the FCNN is considerably faster than the traditional machine learning algorithms. Table 5.1 shows that the FCNN forms predictions approximately two orders of magnitude faster than the SVM classifier and three orders of magnitude faster than the KNN classifier. Additionally, while we recall that the prediction times for the KNN and SVM algorithms scaled with the size of the training set, this is not the case with an FCNN. As mentioned previously, the training times (not reported here) are of less significance than the prediction times, since training can be performed separately, before the deployment of a system, and the training time does not affect normal operation. The other major advantage of the FCNN over the SVM and KNN algorithms is that the FCNN outputs scores for each class, which reflect the confidence in the model's prediction, rather than just the predicted class. This property is extremely useful in flagging observations which are potential misclassifications, and can potentially also be used for open-set recognition, as discussed in the following section.

5.3. Open-set adaptations to the FCNN

Although the trained FCNN is capable of classifying the test set comprised of ISAR images of the known classes (observed in training) with a high degree of accuracy, an automatic target recognition system should also be capable of handling observations that do not belong to any known classes. In this section, we discuss three common ways of adapting a trained FCNN to handle observations of unknown classes.

5.3.1. Softmax with a reject option

For closed-set classification, the output of each output neuron y_k can be interpreted as the probability that a given observation belongs to class k. Using this idea, it is possible to flag or reject class predictions where the probability of the prediction is below some threshold τ.

Given the nature of the softmax function, an FCNN using a softmax activation in the output layer makes the inherent assumption that any observation belongs to one of K known classes. As a result, the output of the K output neurons sum to unity,

$$\sum_{k=1}^{K} y_k = 1. \tag{5.6}$$

Due to this closed-set assumption, it is found that the probability of any observation belonging to none of the known classes is zero according to the model (i.e. there is a closed-set assumption). This method of applying some threshold to the output of the softmax layers is therefore not technically open-set recognition, since the closed-set assumption is still made [53]. One might hope that the estimated probability of the most-likely class according to the network is relatively low. This, however, is not necessarily the case, as is shown in Section 5.3.4.

This method is included here, however, to demonstrate its shortcomings and motivate the necessity for more rigorous approaches to open-set recognition. Additionally, this method is easy to implement, as it does not require any modifications to the trained network. The other open-set adaptations discussed below address some of these shortcomings.

5.3.1.1. Implementation

The only implementation detail to consider is the methodology used to determine the threshold value τ. Since the value of τ influences how sensitive the classifier is to false positives, adjusting this value trades the classifier's precision against its recall. Although the optimal balance between precision and recall may vary from system to system, in this study the optimal balance is considered to be where the harmonic mean of the precision and recall (i.e. F1-score) is maximised. The optimal value for τ is therefore estimated using a small (open) validation set, and plotting the F1-score against the threshold value τ. The threshold $\tau = 0.997$, corresponding to the maximum F1-score on the validation set, was used for testing the classifier.

5.3.2. Openmax

OpenMax [54] was proposed to address some of the problems of the softmax function when applied to open-set recognition. This algorithm replaces the softmax layer of a neural network and predicts the probability that an observation belongs to an unknown class using meta-recognition techniques and the Extreme Value Theory (EVT). This is achieved by first modelling the outputs of the final layer before the softmax function (termed the *activation vectors* AV) for each of the known classes. These models are in turn used to adjust the estimated probability that a given observation belongs to each class. The details of the *OpenMax* algorithm, and the implementation details follow below.

5.3.2.1. Algorithm

After training the FCNN, the activation vectors (AV), $\mathbf{v}_i$ are calculated for each training observation x_i. For each known class k, the *mean activation vector* (MAV) μ_k is calculated over all the correctly classified images in that class:

$$\boldsymbol{\mu}_k = \sum_i \left(\mathbf{S}_{i,k}\right), \tag{5.7}$$

where $\mathbf{S}_{i,k} = \mathbf{v}_i$ for each correctly classified observation x_i in class k. The distance from each AV to the MAV for the relevant class is then calculated. Finally, a Weibull distribution ρ_k is fitted to the η largest distances for each class using the LibMR [109] *FitHigh* function

$$\rho_k = (\tau_k, \kappa_k, \lambda_k) = \text{FitHigh}\left(\|\mathbf{S}_k - \boldsymbol{\mu}_k\|, \eta\right) \tag{5.8}$$

where $\tau_k, \kappa_k, \lambda_k$ are the shifting, shape and scale parameters of the Weibull distribution. Section 5.3.2.2 discusses how the value for the hyperparameter η is chosen. At prediction time, the class probability estimates are updated by using the Weibull cumulative density function to calculate the probability ω_k that the AV of a new observation belongs to each class according to the distributions calculated for that class. In practice, only the top α classes need to be considered, and it is not necessary to adjust the activations corresponding to the other classes. This results in

$$\omega_{s(i)}(\mathbf{x}) = 1 - \frac{\alpha - i}{\alpha} \exp\left(- \left(\frac{\|x - \tau_{s(i)}\|}{\lambda_{s(i)}} \right)^{\kappa_{s(i)}} \right) \tag{5.9}$$

where $s(i)$ is the class corresponding to the ith highest activation for $i = 1, \dots, \alpha$. The revised AV is then calculated by multiplying the original activation vector $\mathbf{v}(\mathbf{x})$ by the weightings $\boldsymbol{\omega}(\mathbf{x})$,

$$\hat{\mathbf{v}} = \mathbf{v}((\mathbf{x})) \cdot \boldsymbol{\omega}(\mathbf{x}) \tag{5.10}$$

The activation corresponding to the probability that the observation belongs to an unknown class is then defined as

$$\hat{v}_0 = \sum_i v_i(\mathbf{x})(1 - \omega_i(\mathbf{x})). \tag{5.11}$$

The final probabilities can finally be calculated from the revised AV as

$$P(y = k|\mathbf{x}) = \frac{e^{\hat{v}_k(\mathbf{x})}}{\sum_{i=0}^{K} e^{\hat{v}_i(\mathbf{x})}}. \tag{5.12}$$

Importantly, it is noted that, when using the *OpenMax* algorithm, the sum of the estimated class probabilities does not sum to unity over the known classes (i.e. the closed-set assumption is not made). Although this means that it is not strictly necessary to reject known class predictions based on their estimated probabilities, [54] notes that the algorithm does nonetheless benefit from applying some threshold ϵ.

5.3.2.2. Implementation

In this work, we used an adaptation of the implementation presented in [54] rewritten in Tensorflow 2.0 (based on [110]). The optimal values for the hyperparameters η, α and ϵ were found experimentally by optimising the F1-score of a validation set. The final choices for these values were 5, 1 and 0.996, respectively. For the meta-recognition functions, this implementation makes use of the libMR library [109].

5.3.3. Training with an explicit 'other' class

The third approach to using an FCNN to perform open-set classification is to include an additional class representing all unknown classes during training. This explicit definition of an 'other' requires numerous examples of *known unknown* observations to effectively train the model. Using this approach to open-set classification assumes that the 'other' class used in training is representative of all unknown targets. In practice, acquiring sufficient data and compiling such a dataset is extremely challenging, if possible at all. It is particularly challenging to verify that the *known unknown* examples adequately represent the *unknown unknown* classes. Despite these challenges, this method often produces reasonable results, even in situations where the 'other' class used in training is a fairly poor approximation for all unknown classes.

5.3.3.1. Implementation

In this work, rather than attempting the (impossible) task of compiling an exhaustive collection of arbitrary targets to represent all unknown targets, ISAR images formed from white Gaussian noise (WGN) were used. The rationale behind this is that images generated from white noise should be distributed relatively evenly throughout the feature space, thus encouraging the decision boundary for each of the known classes to exclude all regions in the feature space that do not contain examples of known classes. After some preliminary experimentation, it was noticed that the unstructured (noise-only) nature of the ISAR images used to represent the 'other' class had the undesirable effect of encouraging the classifier to classify any structured (containing some target) observations as known classes. This effect was particularly prevalent when attempting to classify observations at low SNRs. Combatting this required the addition of ISAR images containing some structure to the 'other' class. This was achieved through the inclusion of ISAR images of a trihedral corner reflector. A trihedral corner reflector is chosen for the following properties:

1. A trihedral corner reflector is a reasonable example of a simple target that is unlikely to be a target of interest to a target recognition system (hence being considered a *known unknown* class)

2. This target produces strong reflections (particularly from look-angles that look into the concave portion), thus resulting in ISAR images with clear structure.

3. The number of scattering centres and their location varies significantly as the imaging angle changes. (We want to increase within-class variation as much as possible for the 'other' class.)

4. Trihedral corner reflectors are commonly used for the calibration of radar systems, so it is expected that data on these targets is readily available for designers of an automatic target recognition system.

The results section reports on the use of this combined 'other' class containing both ISAR images generated from WGN and ISAR images of the trihedral corner reflector from Chapter 3. When testing, a threshold is also applied to the prediction scores (from the softmax output), combining the benefits of training with an open class with the threshold method mentioned previously.

5.3.4. Testing

After applying the three open-set adaptations to the FCNN, the modified networks were tested on an open-set test set. This test set contains ISAR images of the two known targets (specifically the 30 % split not seen in training) as well as an equal number of ISAR images of the fixed-wing aircraft and flying-saucer type UFO target which represent *unknown unknown* classes. For this investigation, only ISAR images with an SNR of 24 dB are considered — it will be shown that even at this (high) SNR, the open-set performance is less than desirable.

5.3.4.1. Metrics

Since the dataset is imbalanced (i.e. 25 % drone class, 25 % missile class and 50 % unknown classes), it is less meaningful to report the total accuracy of the classifier. For this reason, rather than simply reporting the accuracy, confusion matrices are shown for each test case. This gives significantly more insight into the performance of each classifier and indicates where the failure cases are. From the confusion matrices, both the precision,

$$\text{precision} = \frac{\text{Number of correct classifications for class } k}{\text{Total number of observations assigned to class } k}, \tag{5.13}$$

and recall,

$$\text{recall} = \frac{\text{Numer of correct classifications for class } k}{\text{Total number of observations of class } k}, \tag{5.14}$$

can be calculated. To compare the algorithms to one another, however, a single value metric is more useful. For this, the F1-score (i.e. the harmonic mean of the precision and recall) is calculated for each class, and the weighted average is reported for each algorithm.

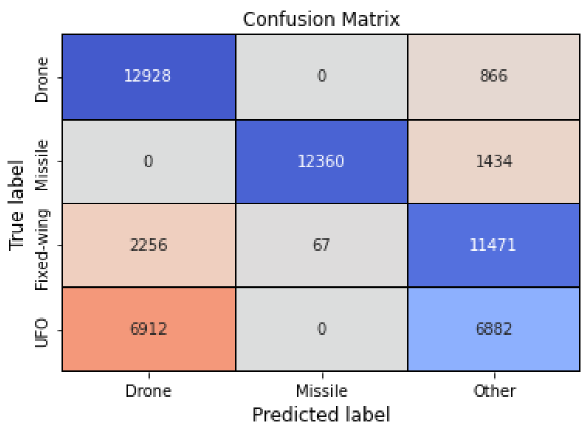
5.3.4.2. Results

The confusion matrices from testing each open-set adaptation to the FCNN are shown in Figure 5.4, while Table 5.2 shows the corresponding weighted-F1 scores. The adaptation

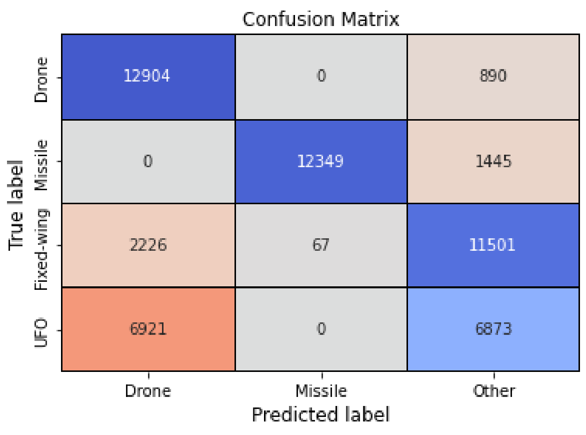
Table 5.2: Weighted-F1 scores for the FCNN open-set adaptations.

FCNN Adaptation	**Weighted-F1 score**
Softmax with a reject option	0.7981
OpenMax layer	0.7989
Training with an explicit ‘other’ class	0.9179

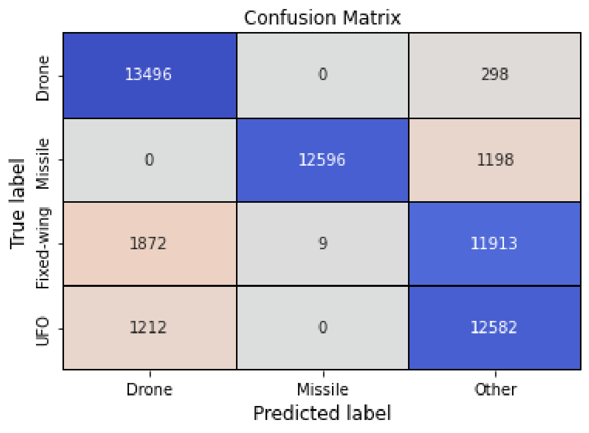
using a softmax layer with a reject option performs open-set classification fairly poorly with an F1-score of 0.7981 on the test set. Additionally, it can be seen that replacing the output layer with an *OpenMax* layer results in negligible improvement. From the confusion matrices in Figures 5.4a and 5.4b, it is noticed that the majority of the misclassifications are instances where targets from the unknown classes were falsely assigned to the drone



(a) Classification with a reject option (using the softmax layer).



(b) Replacing the final layer with an *OpenMax* layer.



(c) Training with an explicit ‘other’ class.

Figure 5.4: Confusion matrices for each of the three open-set adaptations of the FCNN classifier. (Tested on an open-set test set with a SNR of 24 dB for each test image.)

class. As a result, the classifiers have a low recall for the unknown classes and a low precision for the drone class. This is particularly noticeable for the UFO target, where the recall is $<50\,\%$. The poor performance of the classification with a reject option technique highlights the limitations of the softmax output layer. The lack of improvement when adapting the network to use an *OpenMax* layer can be attributed to the fact that the FCNN was trained only to discriminate between the known classes. As a result, the model only extracts and pays attention to discriminative features, and is not encouraged to retain other information about the observations that are useful for open-set recognition. The activation vectors used in the *OpenMax* algorithm are not particularly informative regarding whether a given observation belongs to a known or an unknown class.

Training the FCNN with an explicit ‘other’ class gave better results, however with an F1-score of 0.9179 on the test set, there is still room for improvement. Furthermore, it is noted that the performance of this technique is likely highly sensitive to the training and test sets used (specifically the relationship between the ‘other’ class used in training and the unknown targets used for testing). This is, therefore, not a particularly rigorous approach.

From these results, it is concluded that a rigorous approach to open-set classification which retains more information about the target than just that required to discriminate between the known classes is required. This partially motivates the use of a capsule network, as discussed in the following section.

5.4. Capsule networks

Capsule networks were proposed by Hinton et al. [51] to overcome some of the shortcomings of other neural networks, such as FCNNs and convolutional neural networks (CNN). Specifically, capsules seek to encode the instantiation parameters (referred to as the 'pose' in [51]) of a given observation. Furthermore, capsule networks recognise and estimate the instantiation parameters of the sub-elements of an observation (such as the rotor, wingtip, fins or fuselage of a target), and use this information to recognise larger entities through the (spacial) relationships between their constituent parts. Importantly, the instantiation parameters found are equivariant — meaning that they change in a deterministic manner for different instantiations (orientation, scale, translation etc.) of the same entity. This means that different observations of the same entity are recognised as such (i.e. the same entity, but with different instantiation parameters), even when there is significant variation in the observations [51]. Other neural networks, such as CNNs, which do not do this require enormous amounts of training data, since observations of the same entity under different conditions (such as rotation) are not inherently related to one another in the model — i.e. every instantiation has to be learned independently. The fact that capsule networks recognise the relationship between these observations of the same entity allows capsule networks to be trained on significantly less data [111]. This is an important consideration for automatic target recognition systems, as there is often a limited amount of data available for training.

Because the part-whole relationships and instantiation parameters are built into the network, capsule networks are, by design, highly effective at recognising objects from different viewing angles. This makes capsule networks highly suitable for the classification of ISAR images, which display significant variation with imaging angle. Reference [49] specifically notes that the image variation resulting from various radar imaging views poses a difficult challenge to recognition systems that use ISAR images. Capsule networks specifically address this challenge.

5.4.1. Basic theory

While the basic unit for most artificial neural networks is the artificial neuron, capsule networks make use of 'capsules'. Before explaining how capsule networks work, we first introduce these units.

5.4.1.1. Capsules

Capsules are small groups of neurons which function as a single unit. Rather than outputting a scalar value (as is the case with an artificial neuron), a capsule outputs an activation vector [1]. The length of the activation vector is used to represent the probability that a feature or entity is present, while the individual elements of the vector estimate the instantiation parameters (which could include the orientation, size, deformation and position) of that entity relative to an implicit canonical version [51]. While neurons use non-linear activation functions such as the logistic sigmoid or ReLU, capsules introduce a non-linearity through the *squashing* function [68],

$$\mathbf{v}_i = \frac{\|\mathbf{s}_i\|^2}{1+\|\mathbf{s}_i\|^2}\frac{\mathbf{s}_i}{\|\mathbf{s}_i\|}, \tag{5.15}$$

where $\mathbf{v}_i$ is the vector output of capsule i given an input $\mathbf{s}_i$. This is simply a non-linear scaling of the input vector, which shrinks vectors down such that their length is less than one.

5.4.1.2. Structure

Like in other neural networks, capsules are organised into multiple layers. Some earlier layers may be fully-connected or convolutional layers to perform basic feature extraction, but the final layers which handle the more complex entities should be capsule layers. The most common way of connecting capsule layers to one another is through routing-by-agreement [68], which is outlined below.

5.4.1.3. Dynamic routing algorithm

Through this algorithm, each capsule i, with output vector $\mathbf{u_i}$, in one layer forms a prediction of the output vector $\hat{\mathbf{u}}_{j|i}$ of each capsule j in the subsequent layer by means of some learned weight matrix W_{ij},

$$\hat{\mathbf{u}}_{j|i} = W_{ij}\mathbf{u_i}. \tag{5.16}$$

The final input vector $\mathbf{s}_j$ of each capsule j in the latter layer is calculated as the weighted sum of these predicted outputs from the previous layer,

$$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{i|j}, \tag{5.17}$$

[1]Despite sharing the same name, the activation vector referred to here is not the same as the activation referred to in Section 5.3.2, which is the activation of an entire layer.

where the weighting c_ij is the coupling coefficient between two capsules. The coupling coefficients are determined during the routing process from the log prior probabilities b_{ji} that capsule i should be routed to capsule j,

$$c_{ij} = \frac{e^{b_{ij}}}{\sum_k e^{b_{ik}}}. \tag{5.18}$$

The logits b_{ij} are initialised to 0, before being readjusted over r routing iterations. In each routing iteration, the agreement $a_{ij} = \mathbf{v}_j \cdot \hat{\mathbf{u}}_{j|i}$ between the current output $\mathbf{v}_j$ of capsule j and the prediction $\hat{\mathbf{u}}_{j|i}$ made by capsule i is added to the initial logit b_{ij}. This process is then repeated in the next routing iteration. In this way, the coupling coefficients between the capsules in one layer and the capsules in the next layer are increased where there is a strong agreement between the output and the prediction and are decreased where the agreement is weak. A summary of this algorithm can be found in [68, Procedure 1].

In the case of convolutional capsule layers, each capsule outputs a matrix of vectors to each type of capsule in the following layer. In this case, it is necessary to define a weighting vector for each element in the output matrix, as well as for each capsule type in the second later layer. Otherwise, the procedure remains the same.

5.4.1.4. Network output and training

The final layer of a capsule network contains a single capsule for each class. The estimated probability that a given observation belongs to each class is given by the length (L2-norm) of the output vector of the corresponding class capsule. For training, the network parameters are adjusted to minimise the margin loss L_k for each class capsule k,

$$L_k = T_k \max\left(0, 0.9 - \|\mathbf{v}_k\|\right)^2 + 0.5\left(1 - T_k\right) \max\left(0, \|\mathbf{v}_k\| - 0.1\right)^2, \tag{5.19}$$

where $T_k = 1$ for observations of class k. The second term in (5.19) is used to prevent the lengths of the output vectors in the class capsules from shrinking to 0 during the initial training period [68]. The total loss is then the sum of the margin loss for all class capsules. Note, however, that a capsule network predicts more than just the class probabilities since each class capsule outputs a vector. The network can be trained to encode the instantiation parameters of each observation with these vectors. This is done through reconstruction regularisation.

5.4.1.5. Reconstruction regularisation

Reconstruction regularisation works by routing the output of the class capsules to a decoder network trained to generate a reconstruction $\hat{X}$ of the original input observation X. Before sending the class capsule's output to the reconstruction network, this output is masked such that only the activity vector of the correct class is present and all other vectors are set to **0**. The mean squared error,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \hat{X}_i\right)^2, \tag{5.20}$$

is then added to the training losses. The reconstruction loss is scaled down by a factor of 0.0005 to ensure that the margin loss dominates the training loss. This ensures that the capsule network retains sufficient information to fully describe the input (through the latent representation given by the class capsules).

Although [68] uses reconstruction primarily as a regularisation technique to prevent overfitting, this study makes further use of the model's ability to reconstruct the inputs. Section 5.4.4 discusses how the reconstructions are used to perform open-set recognition.

5.4.2. Implementation

This study uses a similar capsule network architecture to that demonstrated in [68], which was designed to classify handwritten digits from the MNIST dataset [69]. Since the ISAR images used in this study are similar in size and complexity to the MNIST images, there is little need to adapt the architecture.

Originally, an attempt was made to support ISAR images with complex pixel values (represented by separate real and imaginary channels), but it was found that while the classification accuracy was reasonable, the model was unable to properly reconstruct the inputs. It is possible that the dimensionality of the output vectors (i.e. the latent representation of the observations) was insufficient to capture the phase information, or the reconstruction network lacked the necessary complexity. It was found, however, that using the absolute values of the ISAR images produced reasonable results, and it was possible to accurately reconstruct this data. The capsule network designed to work with the (normalised) absolute pixel intensities of the ISAR images, represented by a two-dimensional matrix of real values, is reported on below.

Figure 5.5 shows the network architecture, which has a single standard convolutional layer followed by two capsule layers. The standard convolutional layer contains 256, 9×9 filters, uses a stride of 1 and the ReLU activation function. This layer acts as a basic local feature detector. The first capsule layer, the *primary capsules*, uses 32 channels of 8-dimensional capsules. Each primary capsule contains 8, 9×9 convolutional filters and uses a stride length of 2. The total output of the primary capsule layer is therefore $8 \times 8 \times 32$, 8-dimensional vectors. This output is routed using routing-by-agreement to the final capsule layer, *class capsules*, which uses a 16-dimensional capsule for each class. The lower portion of Figure 5.5 shows the reconstruction network, which is comprised of three fully-connected layers of 512, 1024 and 1024 neurons, respectively. The two hidden layers in the reconstruction network use ReLU activation, while the final layer uses a sigmoid

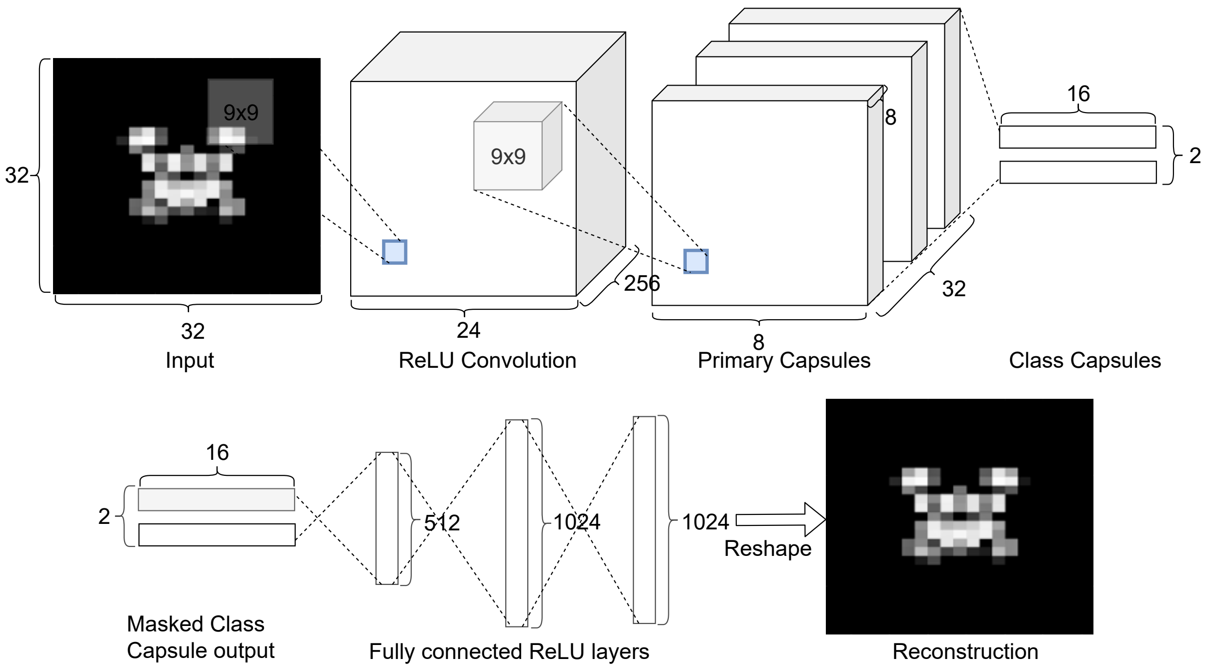


Figure 5.5: Architecture of the capsule network used in this work. The input is fed through a single normal convolutional layer and two convolutional capsule layers to give the output vectors which are used for classification. The output vectors are masked by the predicted class before being used by the reconstruction sub-network to reconstruct the input.

activation since the pixel intensities of the normalised ISAR images are known to lie in the range of zero to one. This network was implemented in Tensorflow 2.0 and was based on the implementation in [112].

5.4.3. Closed-set results

The model was trained on normalised ISAR images of the drone and missile targets which represent known targets. For training, we used a 70 % split of the total number of images in each of the known classes without any added noise (i.e. the same training set used in Sections 4.4.3 and 5.2). The although the model was capable of perfectly separating the training set after roughly 20 epochs, the model was trained for a total of 150 epochs, which significantly improved the quality of the reconstructions. The capsule network was then tested on a closed test set comprised of the remaining 30 % split of the ISAR images of the drone and missile targets, over a range of SNRs. Figure 5.6 plots the F1-score obtained with the closed test set against the SNR. From the results, it can be seen that the capsule network achieves close to perfect classification on the closed set for SNRs above −3 dB. From −6 dB and below, the F1-score drops off, however, it should be noted that at this SNR, the structure of the targets is barely visible from the ISAR images. Figure 5.7 contains a selection of ISAR images of the drone and missile targets at this SNR (−6 dB) to demonstrate the lack of visible target structure. The first two rows of Figure 5.7 are

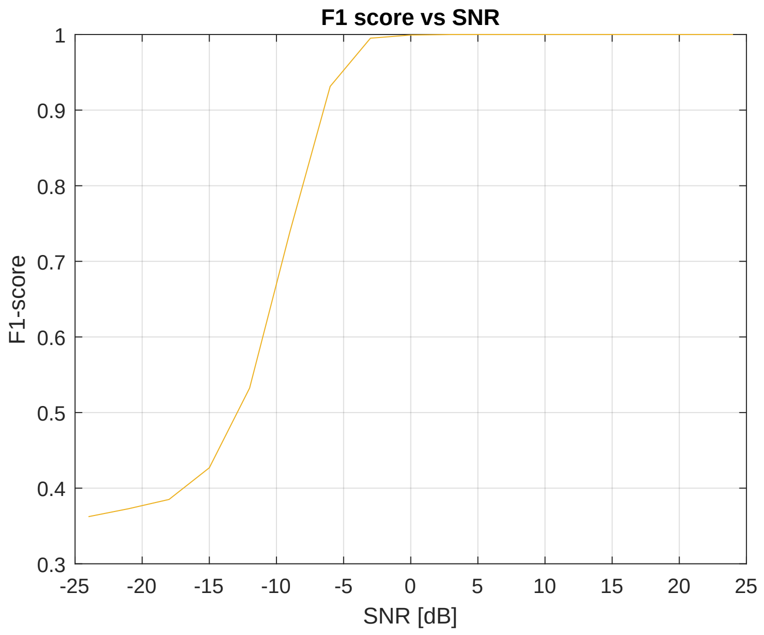


Figure 5.6: Accuracy vs SNR for the capsule network tested on the closed test set. Perfect classification is achieved for all SNRs above -3 dB.

images of the drone target, while the bottom two are images of the missile target. At this SNR, even a trained human operator would have difficulty classifying the targets in these images, while the capsule network still manages to obtain an F1-score of 0.9314.

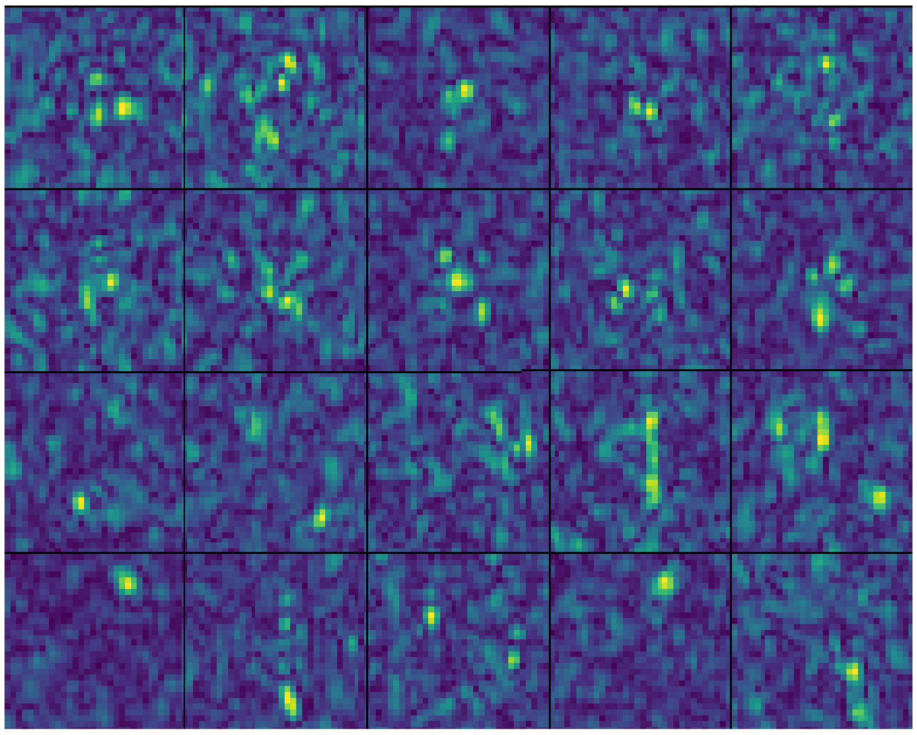


Figure 5.7: Example ISAR images of the drone (top two rows) and missile (bottom two rows) targets at an SNR of -6 dB. At this SNR it is difficult for a human operator to identify the target structure.

5.4.3.1. Prediction times

The average time taken by the capsule network to form a class prediction for a single observation is shown in Table 5.3, along wide the times for the previously discussed classifiers. It can be seen that while the capsule network is an order of magnitude slower than the FCNN (note that this is primarily caused by the inner loop in the routing algorithm), it is still significantly faster than the traditional machine learning algorithms, KNN and SVM. This makes capsule networks a better fit for performing real-time target recognition.

Table 5.3: Prediction times for Capsule Network, FCNN, KNN, SVM classifiers.

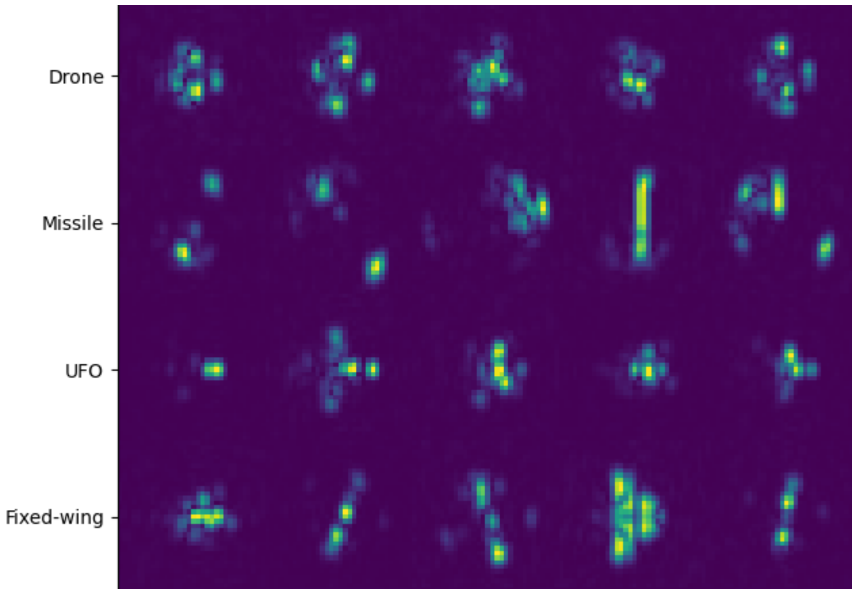
Classifier	**Average prediction time [ms].**
Capsule Network	0.6934
FCNN	0.0445
SVM	1.3000
KNN	48.8000

5.4.4. Open-set adaptation

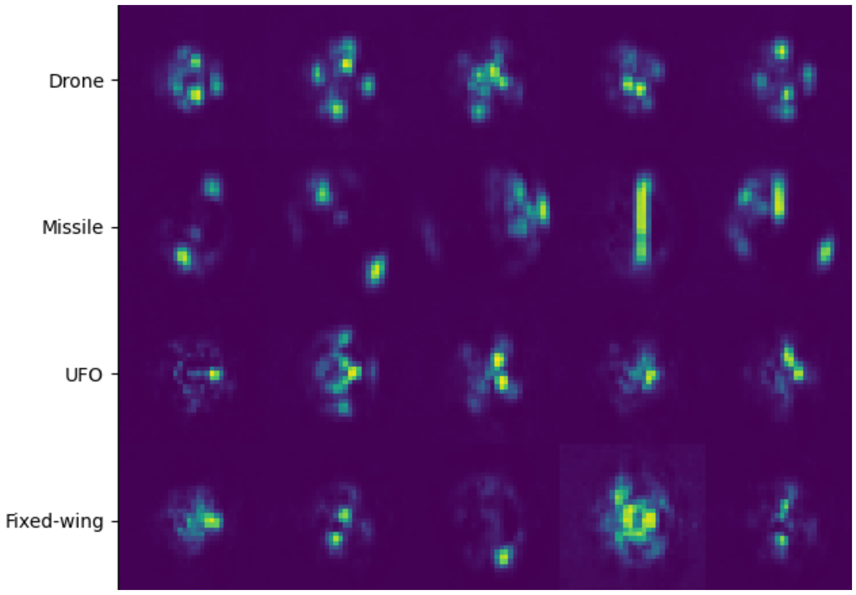
A number of approaches to adapting the capsule network to perform open-set recognition were investigated. Attempts at rejecting predictions with a class prediction score (L2-norm of the class capsule output) below a certain threshold proved unreliable. Using Gaussian mixture model (GMM) clustering to model the distribution of the class capsule outputs was not found to be useful, since it was found that there is a significant overlap between the distributions of the known and unknown classes (hence it is not possible to effectively separate out the unknown classes). One approach of open-set recognition, however, showed much more promise and is discussed below.

5.4.4.1. Using reconstruction for open-set classification

It is recalled that the capsule network was trained to accurately reconstruct ISAR images of the known classes. Additionally, since the input to the reconstruction network is first masked according to the predicted class, it is found that the reconstruction is conditioned on this class prediction. As a result, it is found that while the capsule network is capable of accurately reconstructing the input when the class prediction is correct, the similarity between the reconstruction and the original input is poor when the reconstruction network is conditioned on the incorrect class. Naturally, the reconstruction is conditioned on the correct class whenever the network forms an accurate class prediction, and is conditioned on the incorrect class whenever the prediction is incorrect. This includes instances where the observation belongs to an unknown class, as the network is only capable of conditioning the input to the decoder on one of the known classes. This is demonstrated in Figure 5.8, which visually compares the quality of the reconstructions for five observations from each of the two known targets (the drone and missile targets) and two unknown targets (the fixed-wing model aircraft and the flying-saucer type UFO). It can be seen here that the reconstructions of the unknown classes are notably less similar to the input images. The



(a) Input.



(b) Reconstruction.

Figure 5.8: Example reconstructions (b) corresponding to the input observations (a) for two known and two unknown classes at an SNR of 24 dB. Note that the known targets (top two rows) are more accurately reconstructed than the unknown targets (bottom two rows). The reconstructions are clearly biased towards the known classes.

quality of the reconstruction can be quantified by the mean squared error (MSE) between the input and the reconstruction. By plotting a histogram (Figure 5.9) of the MSE for both the known and unknown classes, it is seen that it is possible to find some threshold value that separates the majority of the unknown classes from the known ones. For any reconstruction with an MSE above this threshold, the observation can be flagged as an

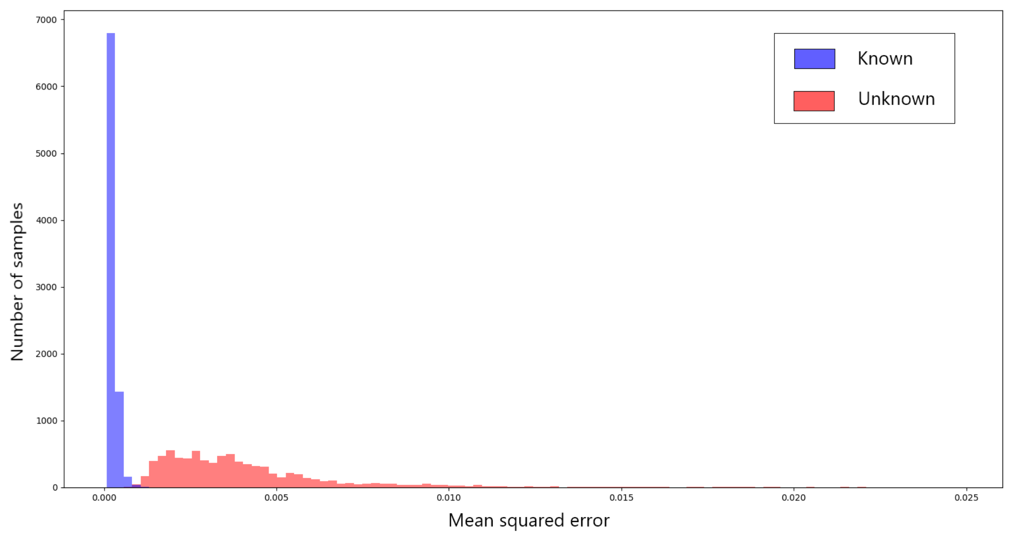


Figure 5.9: Histogram showing the distribution of the MSE between the input and reconstruction for both known and unknown classes at an SNR of 24 dB.

unknown target. The optimal threshold was determined experimentally by plotting the F1-score against the MSE threshold for a small validation set.

5.4.4.2. Results (MSE)

An open test set was created by expanding the closed test set to include ISAR images of the model fixed-wing aircraft and flying-saucer type UFO (i.e. the same open test set described in Section 5.3.4), which represent targets from unknown classes. The capsule network, adapted to flag unknown classes based on the MSE between the input and reconstruction, was then tested on this open test set, and the F1-scores were calculated over a range of SNRs. These results are plotted in Figure 5.10, alongside the results previously obtained on the closed test set. From these results, it can be seen that the modified capsule network performs excellently at high SNRs, with an F1 score above 0.95 on this test set for all SNRs at 15 dB and above. From the confusion matrix in Figure 5.11, it can be seen that at the highest test SNR (24 dB), there are only 54 misclassifications from the 16872 targets tested. The capsule network performing open-set recognition is, however, significantly more sensitive to noise than the network performing closed-set classification. It has previously been noted that for an automatic target recognition system aimed at recognising small targets, the performance at low SNRs is important, and needs to be addressed.

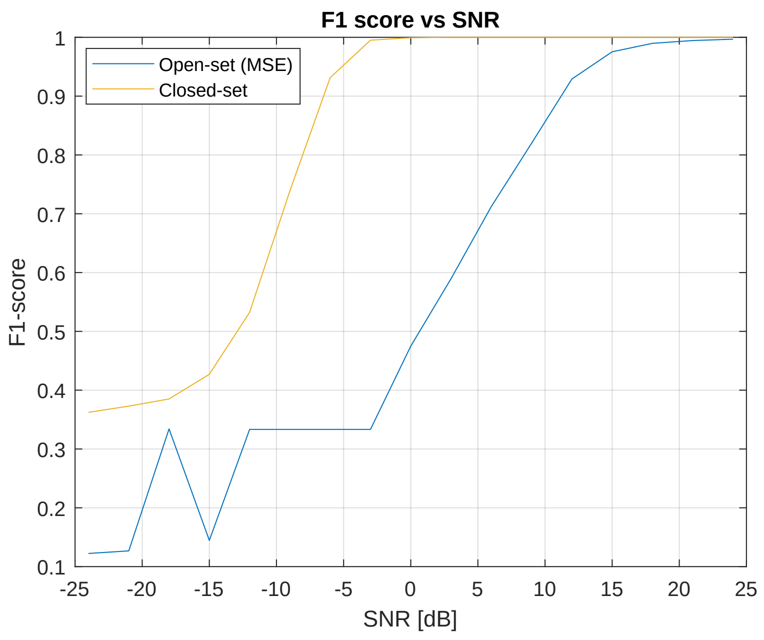
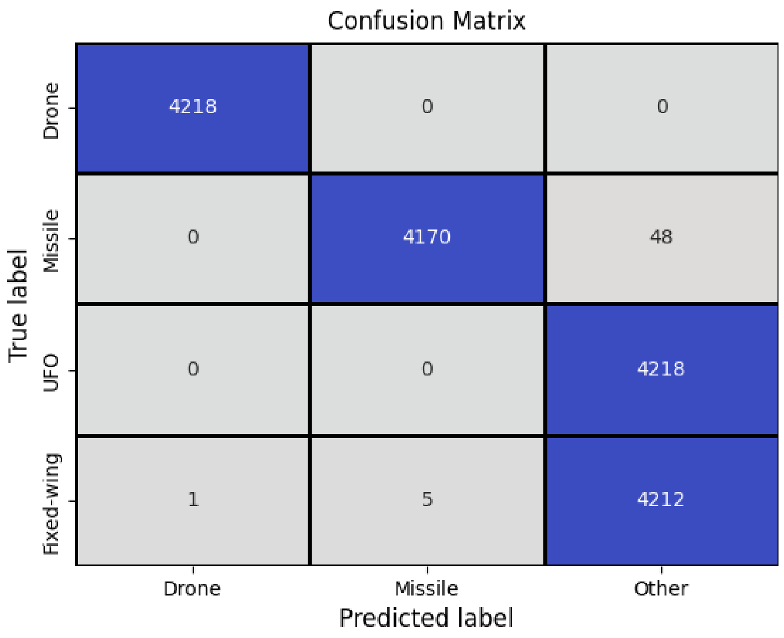


Figure 5.10: Accuracy vs F1-score for the adapted capsule network (using the MSE) tested on the open test set. The closed-set results are included for reference.

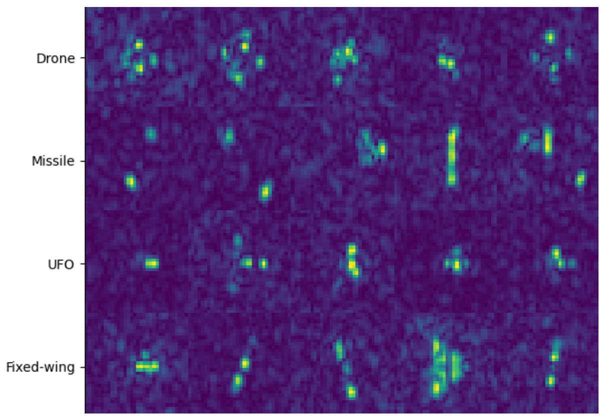

Confusion Matrix
True label
Predicted label
Drone
Missile
UFO
Fixed-wing
Drone
Missile
Other

True label \ Predicted label	Drone	Missile	Other
Drone	4218	0	0
Missile	0	4170	48
UFO	0	0	4218
Fixed-wing	1	5	4212

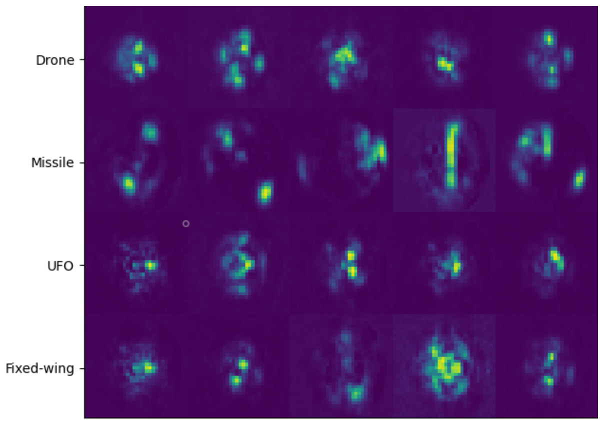
Figure 5.11: Confusion matrix for the adapted capsule network (using the MSE) tested on an open test set at an SNR of 24 dB.

5.4.4.3. Improvement

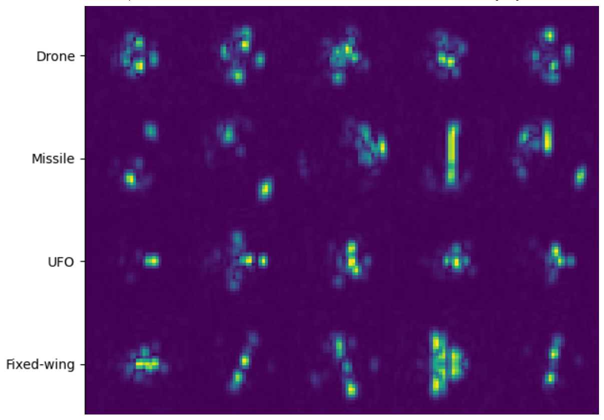
To improve the model's open-set classification accuracy at low SNRs, the results and reconstructions (upon which the open-set recognition system works) were investigated further. Inspecting the reconstructions at low SNRs (given in Figure 5.12), it is observed that the network continues to form reasonably good reconstructions of the target even when there is a significant amount of noise in the input. The obvious issue to note, however, is that the model acts as a denoising filter since it only attempts to reconstruct the signal. The fact that the noise in the input is not reproduced results in a high MSE between the noisy input and reconstructed signal even when the network is producing good reconstructions of the signal and forming accurate class predictions for the known classes. For unknown classes, the signal reconstruction is still notably worse at low SNRs (this is desirable). This shows that the MSE is not the best metric to use, as it is negatively impacted by the



(a) Input (SNR = 3 dB).



(b) Reconstruction.

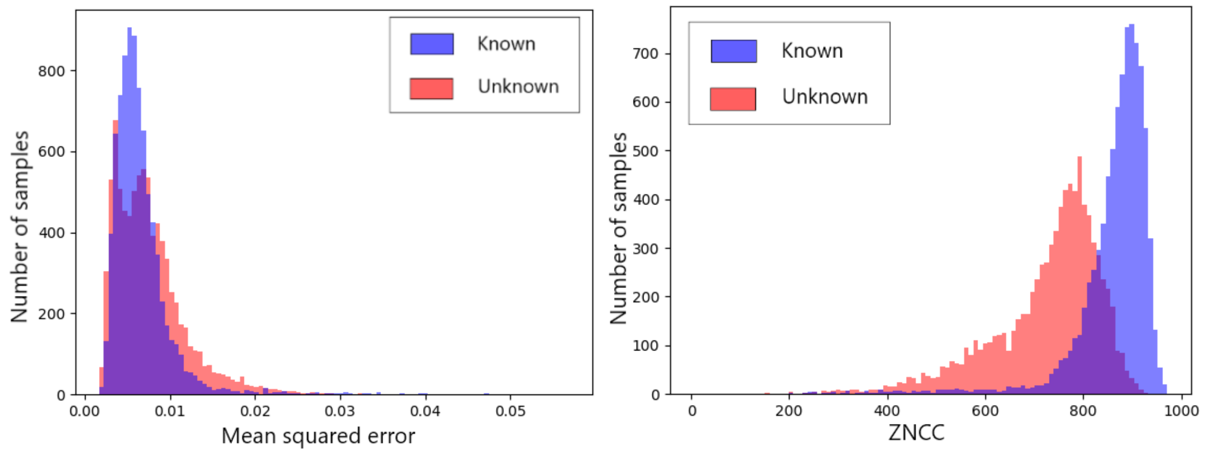


(c) Ground truth.

Figure 5.12: Example reconstructions (b) corresponding to noisy input observations with an SNR of 3 dB (a) for two known and two unknown classes, with the ground truth (ISAR images with no added noise) given in (c). Each row shows five randomly selected examples. The reconstructed images have reduced noise levels and are more similar to the ground truth than the input images (for known targets).

input noise. Instead, this requires some method of comparing the reconstruction to the underlying signal in the input in a way that is less sensitive to the SNR. Although the ground truth is given in Figure 5.12, this information is, obviously, not available during regular operation.

The detection of a known signal in a noise-corrupted sample is a well-known problem in signal processing and is commonly addressed by finding the cross-correlation between the corrupted sample and the reference signal [113]. If the signal is present in the sample, there should be a strong cross-correlation (measured by the cross-correlation coefficient). Applying this principle in two dimensions, this work makes use of the zero-mean normalised cross-correlation (ZNCC) as a metric to estimate the probability that the reconstructed signal found by the capsule network is present in the input. The ZNCC is found by subtracting the mean μ and dividing by the standard deviation σ of the pixel intensities for each image (input and reconstruction) before calculating the cross-correlation. Since the reconstructed signal is expected to be in the same local position as the signal in the input (i.e. there should be no translation), it is only necessary to calculate the cross-correlation at this point to find the peak (cross-correlation coefficient). Figure 5.13 compares the ZNCC and MSE as metrics, showing that using the ZNCC makes it much easier to separate the unknown classes from the known classes for low SNRs.



(a) MSE. **(b)** ZNCC coefficient.

Figure 5.13: Histograms of the MSE (a) and ZNCC coefficient (b) between the reconstructed images and input images at an SNR of 3 dB. Much better separation can be seen with the use of the ZNCC coefficient.

5.4.4.4. Results (ZNCC)

After adapting the capsule network a second time, using the ZNCC coefficient as a metric rather than the MSE, the network was tested again on the open test set. The threshold for the ZNCC coefficient was determined in the same way as that for the MSE, by plotting the F1 against the threshold value for a validation set. These results are given in Figure 5.14. These results show that the use of the ZNCC coefficient improves the capsule network's ability to perform open-set recognition (when compared to the use of the MSE). The

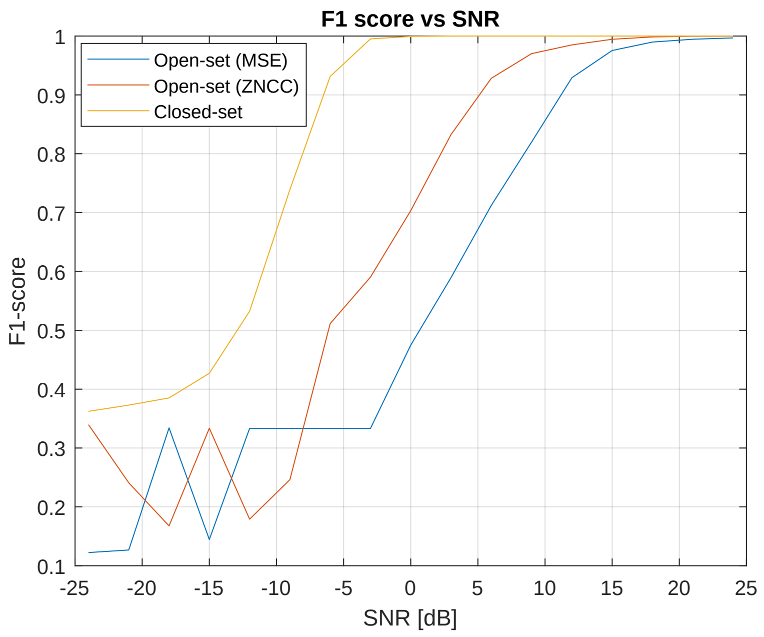


Figure 5.14: F1-score vs SNR for the open test set using the capsule network adapted to use the ZNCC coefficient to identify unknown classes (including the previous results for comparison). Using the ZNCC shows significant improvement over the MSE.

difference is particularly noticeable in the SNR range from -6 dB to 15 dB. The improved open-set classifier maintains an F1-score greater than 0.9 for all SNRs from 6 db and above. While the performance of the open-set classifier is worse than the close-set classifier, this is to be expected given that the open-set problem is significantly more complex.

5.5. Conclusion

In this chapter, two deep-learning approaches, FCNNs and capsule networks, are introduced and applied to the ISAR image classification problem. These methods (particularly the capsule network) are shown to be highly effective when applied to the closed-set classification problem. Open-set adaptations for both types of networks were investigated, giving mixed results. It was found that given the FCNN's discriminative nature, it does not retain sufficient information about the targets to perform open-set classification reliably. On the other hand, the capsule network, which uses reconstruction regularisation to ensure sufficient information is retained to describe the observations fully, could be adapted to handle observations of unknown classes effectively. The capsule network was adapted to recognise observations of unknown classes by comparing the reconstructed image (where reconstruction is conditioned on a known class) to the original input ISAR image. Furthermore, attention was paid to improving the open-set adapted capsule network's performance by reducing its sensitivity to noise, thereby making it more appropriate for classifying small targets.

Chapter 6

Conclusion and Recommendations

6.1. Conclusion

In this study, a dataset of synthetic ISAR images was generated using CEM simulation and a Fourier-based ISAR processing technique. The dataset contains ISAR images of four small complex targets (two of which were used to represent known classes and two to represent *unknown unknown* classes) and one simple corner reflector target (used as a *known unknown* target). Chapter 3 detailed the process that was used to generate and process the synthetic ISAR data and provides some examples of the generated images.

The preliminary investigation into the application of traditional machine learning algorithms, reported on in Chapter 4, produced several findings. It was found that a simple linear classification technique, LR, was unable to effectively separate the known classes with sufficient accuracy. The non-linear traditional machine learning techniques, KNN and SVM, were significantly more successful at accurately classifying new observations of known classes. It was further found that introducing examples of ISAR images partially corrupted with noise (up to an SNR of approximately 15 dB) into the training set reduced the classifiers' sensitivity to noise in the training set. Fourier interpolation, however, while producing more aesthetically pleasing images, did not benefit the performance of these algorithms and resulted in lower computational efficiency. Despite the high classification accuracy on the closed set, two major limitations of these (KNN and SVM) algorithms were noted. Firstly, the time required to form predictions of the class membership of new observations using the SVM and KNN algorithms scaled with the size of the training data. Even with a relatively small training set of only two targets, the average prediction times were found to be larger than desirable for real-time classification. Secondly, while these techniques showed promise in the closed-set classification problem, these algorithms only provide the class prediction and do not estimate the probability that a given observation belongs to each class. This makes it difficult to gauge the uncertainty in each prediction, which is required in some way to enable the classifier to handle observations of targets which do not belong to any of the known classes.

The shortcomings of the traditional machine learning algorithms used for image classification prompted the investigation into deep learning approaches using ANNs.

The first type of ANN investigated, an FCNN, demonstrated a classification accuracy comparable to the traditional machine learning approaches, lying somewhere between that achieved by the SVM and the near-perfect classification accuracy achieved by the KNN classifier for SNRs above 0 dB. Although the FCNN did not improve on the closed-set classification accuracy, it was found that its similar classification accuracy came at a significantly lower computational cost compared to KNN. The prediction times for the FCNN were two to three orders of magnitude faster than those of the two non-linear traditional machine learning models. Furthermore, it was shown that the class scores given by the FCNN could be used to some degree to handle observations of targets belonging to unknown classes. The best-performing open-set adaptation of the FCNN did not, however, rely on the FCNN's known class prediction scores, but rather used an explicit 'other' class in training to approximate all classes other than the known classes. With this method, the FCNN achieved an F1-score of at most 0.918 on the open test set (at the highest SNR, 24 dB). Although three such open-set adaptations to FCNNs were explored, none of these approaches resulted in satisfactory open-set classification accuracy. It was suggested that the fact that the FCNN was trained to specifically extract discriminative features for the known classes, imposes a limit on the effectiveness of open-set adaptations to FCNNs. For a classifier to minimise open-set risk, it should retain more information about the target than just that required to differentiate between the known classes. From this, it appears evident that for effective open-set classification, it is insufficient to simply adapt a closed-set classifier and expect reasonable results. When selecting an appropriate classification algorithm, the open-set classification task should be considered from the beginning. This partially motivated the investigation of the second type of ANN discussed in this study, a capsule network.

Chapter 5 outlines how reconstruction regularisation is used in conjunction with capsule networks to ensure that sufficient information about each target observation is retained to fully describe the observation (for known classes). It is demonstrated how a capsule network can be leveraged to perform open-set classification — recognising when the known-class prediction is a poor fit, and flagging the corresponding observation as an unknown class. To do this, a novel approach is proposed whereby a comparison of the original ISAR image input and the ISAR image reconstructed from the latent representation encoded by the capsule network is used to separate known and unknown classes. Owing to this and the way in which capsule networks handle variations in the observation as the target undergoes different transformations, excellent results were obtained for both the closed-set and open-set classification tests. It is further shown that changing the metric used to compare the reconstructed images to the input image (from MSE to ZNCC), improves the capsule network's open-set classification performance at lower SNRs.

The final capsule network model was ultimately shown to maintain an F1-score greater than 0.9 for SNRs at 6 dB and above when tested on a test set containing observations of targets from two known classes and two unknown classes. The capsule network is thus a viable option for the classification of ISAR images for known and unknown targets. There still is room for improvement in classification at low SNR levels. This meets the project aims that were set out in Chapter 1.

6.2. Improvements and recommendations

The classifiers presented in this study are trained and tested on a limited set of synthetic ISAR images generated from CEM simulation. Verifying the results obtained in this work using ISAR images generated from measured data is therefore an obvious extension. Gathering and processing measured data to form ISAR images would likely require more sophisticated processing techniques, such as the use of compressive sensing to handle sparse measurements and motion compensation techniques. It would be particularly useful if it could be verified that models trained on synthetic data, which is more easily obtainable, remain effective when tested on measured data. Additionally, the inclusion of more known classes would increase the complexity of the classification task at hand. Future work could be done to investigate how well the proposed techniques perform under these conditions.

There is also room for improvement on the final open-set classifier (the capsule network) at low SNRs. While this work showed some improvement in the classifier's sensitivity to noise, the performance on the open test set is still far below that obtained on the closed set at lower SNRs. Preprocessing of the noisy inputs could potentially be used to further improve the system's sensitivity to noise. These preprocessing techniques could include denoising techniques, such as the use of a denoising auto-encoder, to improve the SNR. Alternatively, the use of a blob-detection algorithm could be used to extract the structure of the target from the ISAR image — potentially reducing the effect of noise in the inputs to the classifier.

Various improvements to capsule networks have previously been proposed and demonstrated on other datasets, such as the MNIST dataset in other studies [111]. Some of these improvements, such as the use of matrix capsules [114] might be demonstrated to show similar improvements when applied to the classification of ISAR images.

Reference List

[1] M. Pieraccini, L. Miccinesi, and N. Rojhani, "RCS measurements and ISAR images of small UAVs," *IEEE Aerospace and Electronic Systems Magazine*, vol. 32, no. 9, pp. 28–32, 2017.

[2] W.-K. Lee and K.-M. Song, "Enhanced ISAR Imaging for Surveillance of Multiple Drones in Urban Areas," in *2018 International Conference on Radar (RADAR)*, 2018, pp. 1–4.

[3] J. É. Cilliers, "Information Theoretic Limits on Non-cooperative Airborne Target Recognition by Means of Radar Sensors," Ph.D. dissertation, UCL (University College London), 2018.

[4] A. Stove and S. Sykes, "A Doppler-based target classifier using linear discriminants and principal components," in *2003 Proceedings of the International Conference on Radar (IEEE Cat. No.03EX695)*, 2003, pp. 171–176.

[5] S. Dodge and L. Karam, "A Study and Comparison of Human and Deep Learning Recognition Performance under Visual Distortions," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–7.

[6] Cohen, "An overview of radar-based, automatic, noncooperative target recognition techniques," in *IEEE 1991 International Conference on Systems Engineering*, 1991, pp. 29–34.

[7] G. E. Smith, K. Woodbridge, and C. J. Baker, "Micro-Doppler Signature Classification," in *2006 CIE International Conference on Radar*, 2006, pp. 1–4.

[8] E. Reid, M. Lewis, E. Hughes, E. Reid, M. Lewis, and E. Hughes, "The Application of Speech Recognition Techniques to Radar Target Doppler Recognition: A Case Study," in *2006 IET Seminar on High Resolution Imaging and Target Classification*, 2006, pp. 145–152.

[9] A. Hanif, M. Muaz, A. Hasan, and M. Adeel, "Micro-Doppler Based Target Recognition With Radars: A Review," *IEEE Sensors Journal*, vol. 22, no. 4, pp. 2948–2961, 2022.

[10] Z. Shi, J. Li, Y. Zhang, and X. Lu, "HRR radar imaging based on compressed samples using dynamic dictionaries," in *IET International Conference on Radar Systems (Radar 2012)*, 2012, pp. 1–6.

[11] H.-J. Li and S.-H. Yang, "Using range profiles as feature vectors to identify aerospace objects," *IEEE Transactions on Antennas and Propagation*, vol. 41, no. 3, pp. 261–268, 1993.

[12] A. Zyweck and R. Bogner, "Radar target classification of commercial aircraft," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 2, pp. 598–606, 1996.

[13] E. R. Keydel, S. W. Lee, and J. T. Moore, "MSTAR extended operating conditions: a tutorial," in *Algorithms for Synthetic Aperture Radar Imagery III*, E. G. Zelnio and R. J. Douglass, Eds., vol. 2757, International Society for Optics and Photonics. SPIE, 1996, pp. 228 – 242. [Online]. Available: https://doi.org/10.1117/12.242059

[14] S. Wang, Y. Wang, H. Liu, and Y. Sun, "Attribute-guided multi-scale prototypical network for few-shot SAR target classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 12 224–12 245, 2021.

[15] C. Li and H. Ling, "An Investigation on the Radar Signatures of Small Consumer Drones," *IEEE Antennas and Wireless Propagation Letters*, vol. 16, pp. 1–1, 01 2016.

[16] Novak, "A comparison of 1D and 2D algorithms for radar target classification," in *IEEE 1991 International Conference on Systems Engineering*, 1991, pp. 6–12.

[17] J. Seybold and S. Bishop, "Three-dimensional ISAR imaging using a conventional high-range resolution radar," in *Proceedings of the 1996 IEEE National Radar Conference*, 1996, pp. 309–314.

[18] J. Wei, S. Shao, H. Ma, P. Wang, L. Zhang, and H. Liu, "High-Resolution ISAR Imaging with Modified Joint Range Spatial-Variant Autofocus and Azimuth Scaling," *Sensors*, vol. 20, no. 18, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5047

[19] W. Zhou, C.-m. Yeh, R.-j. Jin, J. Yang, and J.-s. Song, "Rotation estimation for wide-angle inverse synthetic aperture radar imaging," *Journal of Sensors*, vol. 2016, 2016.

[20] B. Haywood, R. Kyprianou, and A. Zyweck, "ISARLAB: A radar signal processing tool," in *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. v, 1994, pp. V/177–V/180 vol.5.

[21] J. Maxwell and T. Torrance, *A Dynamical Theory of the Electromagnetic Field*, ser. Torrance collection. Wipf & Stock, 1996.

[22] W. Gibson, *The Method of Moments in Electromagnetics.* CRC Press, 2014.

[23] S. Makarov, *Antenna and EM Modeling in MATLAB*, ser. A Wiley-Interscience publication. Wiley, 2002.

[24] D. P. Xiang, "Fast mesh-based physical optics for large-scale electromagnetic analysis," 2016. [Online]. Available: http://hdl.handle.net/10019.1/100185

[25] D. Davidson, *Computational Electromagnetics for RF and Microwave Engineering.* Cambridge University Press, 2005.

[26] C. Balanis, L. Sevgi, and P. Ufimtsev, "Fifty years of high frequency diffraction," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 23, 07 2013.

[27] R. Bhalla and H. Ling, "ISAR image simulation of targets with moving parts using the shooting and bouncing ray technique," in *Proceedings of IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*, vol. 3, 1994, pp. 1994–1997 vol.3.

[28] F. Wang, T. F. Eibert, and Y.-Q. Jin, "Simulation of ISAR Imaging for a Space Target and Reconstruction Under Sparse Sampling via Compressed Sensing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 6, pp. 3432–3441, 2015.

[29] E. Lezar and U. Jakobus, "GPU-acceleration of the FEKO electromagnetic solution kernel," *2013 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, pp. 814–817, 2013.

[30] T. Küçükkılıç, "ISAR imaging and motion compensation," Master's thesis, Middle East Technical University, 2006.

[31] C. Ozdemir, *Inverse synthetic aperture radar imaging with MATLAB algorithms.* John Wiley & Sons, 2012, vol. 210.

[32] N. Blomerus, J. Cilliers, and J. de Villiers, "Development and Testing of a Low Cost Audio Based ISAR Imaging and Machine Learning System for Radar Education," in *2020 IEEE International Radar Conference (RADAR)*, 2020, pp. 766–771.

[33] A. K. Roy, S. A. Gangal, and C. Bhattacharya, "Demonstration of backprojection algorithm for ISAR image formation with FMCW radar," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2016, pp. 1078–1081.

[34] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, 1986.

[35] J. Odendaal, E. Barnard, and C. Pistorius, "Two-dimensional superresolution radar imaging using the MUSIC algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 42, no. 10, pp. 1386–1391, 1994.

[36] C. Hu, L. Wang, Z. Li, and D. Zhu, "Inverse Synthetic Aperture Radar Imaging Using a Fully Convolutional Neural Network," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 7, pp. 1203–1207, 2020.

[37] A. Bilal, S. M. Hamza, Z. Taj, S. Salamat, and M. Abbas, "ISAR Imaging using FFT with Polar Reformatting of Measured RCS," in *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2020, pp. 1–5.

[38] A. Khan and H. Farooq, "Principal component analysis-linear discriminant analysis feature extractor for pattern recognition," *arXiv preprint arXiv:1204.1177*, 2012.

[39] S.-h. Park, J.-h. Jung, S.-h. Kim, and K.-t. Kim, "Efficient classification of ISAR images using 2D Fourier transform and polar mapping," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 3, pp. 1726–1736, 2015.

[40] S. Fulin, L. Dafang, N. Liang, and S. Huadong, "Feature extraction and selection based on ATR of ISAR image," in *Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04. 2004.*, vol. 2, 2004, pp. 1415–1418 vol.2.

[41] E. Botha, "Classification of aerospace targets using superresolution ISAR images," in *Proceedings of COMSIG '94 - 1994 South African Symposium on Communications and Signal Processing*, 1994, pp. 138–145.

[42] J.-C. Cexus, A. Toumi, and M. Riahi, "Target recognition from ISAR image using polar mapping and shape matrix," in *2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2020, pp. 1–6.

[43] J.-G. Lee, S. Jun, Y.-W. Cho, H. Lee, G. B. Kim, J. B. Seo, and N. Kim, "Deep learning in medical imaging: general overview," *Korean journal of radiology*, vol. 18, no. 4, pp. 570–584, 2017.

[44] H. Yang, Y.-s. Zhang, C.-b. Yin, and W.-z. Ding, "Ultra-lightweight CNN design based on neural architecture search and knowledge distillation: A novel method to build the automatic recognition model of space target ISAR images," *Defence Technology*, vol. 18, no. 6, pp. 1073–1095, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214914721000763

[45] A. Avadhani, S. Chaudhari, P. Gacheria, and S. Ahuja, "Inverse Synthetic-Aperture Radar (ISAR) Images Recognition Using Deep Learning," in *2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA)*, 2020, pp. 293–298.

[46] B. Xue and N. Tong, "Real-World ISAR Object Recognition Using Deep Multimodal Relation Learning," *IEEE Transactions on Cybernetics*, vol. 50, no. 10, pp. 4256–4267, 2020.

[47] S. Zaied, A. Toumi, and A. Khenchaf, "Target classification using convolutional deep learning and auto-encoder models," in *2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2018, pp. 1–6.

[48] Y. Li, B. Yang, Z. He, and R. Chen, "An ISAR Automatic Target Recognition Approach Based on SBR-based Fast Imaging Scheme and CNN," in *2020 IEEE MTT-S International Wireless Symposium (IWS)*, 2020, pp. 1–3.

[49] X. He, N. Tong, and X. Hu, "Automatic recognition of ISAR images based on deep learning," in *2016 CIE International Conference on Radar (RADAR)*, 2016, pp. 1–4.

[50] X. Zhou, X. Bai, L. Wang, and F. Zhou, "Robust ISAR Target Recognition Based on ADRISAR-Net," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–1, 2022.

[51] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International conference on artificial neural networks*. Springer, 2011, pp. 44–51.

[52] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability Models for Open Set Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2317–2324, 2014.

[53] C. Geng, S.-J. Huang, and S. Chen, "Recent Advances in Open Set Recognition: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3614–3631, 2021.

[54] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.

[55] B. Gao and L. Pavel, "On the properties of the softmax function with application in game theory and reinforcement learning," *arXiv preprint arXiv:1704.00805*, 2017.

[56] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, and T. Naemura, "Classification-reconstruction learning for open-set recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4016–4025.

[57] W. Zhao, A. Heng, L. Rosenberg, S. T. Nguyen, L. Hamey, and M. Orgun, "ISAR Ship Classification Using Transfer Learning," in *2022 IEEE Radar Conference (RadarConf22)*, 2022, pp. 1–6.

[58] S. Dang, Z. Cao, Z. Cui, and Y. Pi, "Open Set SAR Target Recognition Using Class Boundary Extracting," in *2019 6th Asia-Pacific Conference on Synthetic Aperture Radar (APSAR)*, 2019, pp. 1–4.

[59] E. Zelnio and A. Pavy, "Open set SAR target classification," in *Algorithms for Synthetic Aperture Radar Imagery XXVI*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 10987, Jul. 2019, p. 109870J.

[60] X. Ma, K. Ji, L. Zhang, S. Feng, B. Xiong, and G. Kuang, "An Open Set Recognition Method for SAR Targets Based on Multitask Learning," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[62] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[63] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[64] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 104–12 113.

[65] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, "Coca: Contrastive captioners are image-text foundation models," *arXiv preprint arXiv:2205.01917*, 2022.

[66] B. Xiao, J.-F. Ma, and J.-T. Cui, "Combined blur, translation, scale and rotation invariant image recognition by Radon and pseudo-Fourier–Mellin transforms," *Pattern Recognition*, vol. 45, no. 1, pp. 314–321, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320311002883

[67] X. Wang, B. Xiao, J.-F. Ma, and X.-L. Bi, "Scaling and rotation invariant analysis approach to object recognition based on Radon and Fourier–Mellin transforms,"

Pattern Recognition, vol. 40, no. 12, pp. 3503–3508, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320307002063

[68] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *Advances in neural information processing systems*, vol. 30, 2017.

[69] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[70] Z. Yang and S. Jing, "SAR image classification method based on improved capsule network," *Journal of Physics: Conference Series*, vol. 1693, no. 1, p. 012181, dec 2020. [Online]. Available: https://doi.org/10.1088/1742-6596/1693/1/012181

[71] H. Ren, X. Yu, L. Zou, Y. Zhou, X. Wang, and L. Bruzzone, "Extended convolutional capsule network with application on SAR automatic target recognition," *Signal Processing*, vol. 183, p. 108021, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165168421000608

[72] R. Shah, A. Soni, V. Mall, T. Gadhiya, and A. K. Roy, "Automatic Target Recognition from SAR Images Using Capsule Networks," in *Pattern Recognition and Machine Intelligence*, B. Deka, P. Maji, S. Mitra, D. K. Bhattacharyya, P. K. Bora, and S. K. Pal, Eds. Cham: Springer International Publishing, 2019, pp. 377–386.

[73] C. Schwegmann, W. Kleynhans, B. Salmon, L. Mdakane, and R. Meyer, "Synthetic Aperture Radar Ship Detection Using Capsule Networks," in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018, pp. 725–728.

[74] Y. Gao, F. Gao, J. Dong, and H.-C. Li, "SAR image change detection based on multiscale capsule network," *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 3, pp. 484–488, 2020.

[75] S. Björklund, "Target detection and classification of small drones by boosting on radar micro-doppler," in *2018 15th European Radar Conference (EuRAD)*, 2018, pp. 182–185.

[76] X. Zou, A. Deng, Y. Hu, S. Hua, L. Zhang, S. Xu, and W. Zou, "High-resolution and reliable automatic target recognition based on photonic ISAR imaging system with explainable deep learning," *arXiv preprint arXiv:2212.01560*, 2022.

[77] H. Yang, Y. Zhang, and W. Ding, "A fast recognition method for space targets in ISAR images based on local and global structural fusion features with lower dimensions," *International Journal of Aerospace Engineering*, vol. 2020, pp. 1–21, 2020.

[78] ——, “Multiple Heterogeneous P-DCNNs Ensemble With Stacking Algorithm: A Novel Recognition Method of Space Target ISAR Images Under the Condition of Small Sample Set,” *IEEE Access*, vol. 8, pp. 75 543–75 570, 2020.

[79] “Simulation for Connectivity, Compatibility, and Radar — Altair FEKO,” 2020. [Online]. Available: https://www.altair.com/feko/

[80] “GrabCAD Community.” [Online]. Available: https://grabcad.com/dashboard

[81] M. Zhan, P. Huang, X. Liu, G. Liao, Z. Zhang, Z. Wang, and H. Fan, “An ISAR imaging and cross-range scaling method based on phase difference and improved axis rotation transform,” *Digital Signal Processing*, vol. 104, p. 102798, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1051200420301433

[82] X.-g. Xia, G. Wang, and V. Chen, “Quantitative SNR analysis for ISAR imaging using joint time-frequency analysis-Short time Fourier transform,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 2, pp. 649–659, 2002.

[83] C. M. Bishop, “Training with Noise is Equivalent to Tikhonov Regularization,” *Neural Computation*, vol. 7, no. 1, pp. 108–116, 01 1995. [Online]. Available: https://doi.org/10.1162/neco.1995.7.1.108

[84] D. Böhning, “Multinomial logistic regression algorithm,” *Annals of the institute of Statistical Mathematics*, vol. 44, no. 1, pp. 197–200, 1992.

[85] C. G. BROYDEN, “The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations,” *IMA Journal of Applied Mathematics*, vol. 6, no. 1, pp. 76–90, 03 1970. [Online]. Available: https://doi.org/10.1093/imamat/6.1.76

[86] R. Fletcher, “A new approach to variable metric algorithms,” *The Computer Journal*, vol. 13, no. 3, pp. 317–322, 01 1970. [Online]. Available: https://doi.org/10.1093/comjnl/13.3.317

[87] D. Goldfarb, “A family of variable-metric methods derived by variational means,” *Mathematics of computation*, vol. 24, no. 109, pp. 23–26, 1970.

[88] D. F. Shanno, “Conditioning of quasi-newton methods for function minimization,” *Mathematics of computation*, vol. 24, no. 111, pp. 647–656, 1970.

[89] M. A. Chandra and S. Bedi, “Survey on SVM and their application in image classification,” *International Journal of Information Technology*, vol. 13, no. 5, pp. 1–11, 2021.

[90] P. L. Cross, “Maritime automated targets recognition algorithm test bed for high resolution isar imagery,” in *2013 IEEE International Conference on Technologies for Homeland Security (HST)*, 2013, pp. 369–374.

[91] F. Ma, Y. He, and Y. Li, “Comparison of aircraft target recognition methods based on ISAR images,” in *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 10, 2022, pp. 1291–1295.

[92] F. N. Khan, Q. Fan, C. Lu, and A. P. T. Lau, “An Optical Communication’s Perspective on Machine Learning and Its Applications,” *Journal of Lightwave Technology*, vol. 37, no. 2, pp. 493–516, 2019.

[93] D. Sculley, “Large scale learning to rank,” 2009.

[94] C. Cortes and V. Vapnik, “Support-Vector Networks,” in *Machine Learning*, 1995, pp. 273–297.

[95] S. J. Wright, “Coordinate descent algorithms,” *Mathematical Programming*, vol. 151, pp. 3–34, 2015.

[96] M. A. Aizerman, “Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning,” *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.

[97] J. Mercer, “Xvi. functions of positive and negative type, and their connection the theory of integral equations,” *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, vol. 209, no. 441-458, pp. 415–446, 1909.

[98] J. Platt, “Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods,” *Adv. Large Margin Classif.*, vol. 10, 06 2000.

[99] K. Taunk, S. De, S. Verma, and A. Swetapadma, “A Brief Review of Nearest Neighbor Algorithm for Learning and Classification,” in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, 2019, pp. 1255–1260.

[100] B. Everitt, S. Landau, and M. Leese, *Cluster Analysis*, ser. A Hodder Arnold Publication. Wiley, 2001. [Online]. Available: https://books.google.co.za/books?id=htZzDGlCnQYC

[101] R. Souza, R. Lotufo, and L. Rittner, “A Comparison between Optimum-Path Forest and k-Nearest Neighbors Classifiers,” in *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, 2012, pp. 260–267.

[102] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. Awwal, and V. K. Asari, “A state-of-the-art survey on deep learning theory and architectures,” *Electronics*, vol. 8, no. 3, p. 292, 2019.

[103] M. A. Nielsen, *Neural networks and deep learning.* Determination press San Francisco, CA, USA, 2015, vol. 25.

[104] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[105] J. Heaton, *Introduction to Neural Networks with Java.* Heaton Research, 2008. [Online]. Available: https://books.google.co.za/books?id=Swlcw7M4uD8C

[106] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[107] G. W. Leibniz, *The early mathematical manuscripts of Leibniz.* Courier Corporation, 2012.

[108] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[109] W. J. Scheirer, A. Rocha, R. Michaels, and T. E. Boult, “Meta-Recognition: The Theory and Practice of Recognition Score Analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 33, pp. 1689–1695, 2011.

[110] A. Neupane and W. Z. E. Amri, “aadeshnpn/OSDN,” https://github.com/aadeshnpn/OSDN, 2020.

[111] M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty, and B. Y. Edward, “Capsule Networks – A survey,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 1, pp. 1295–1310, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157819309322

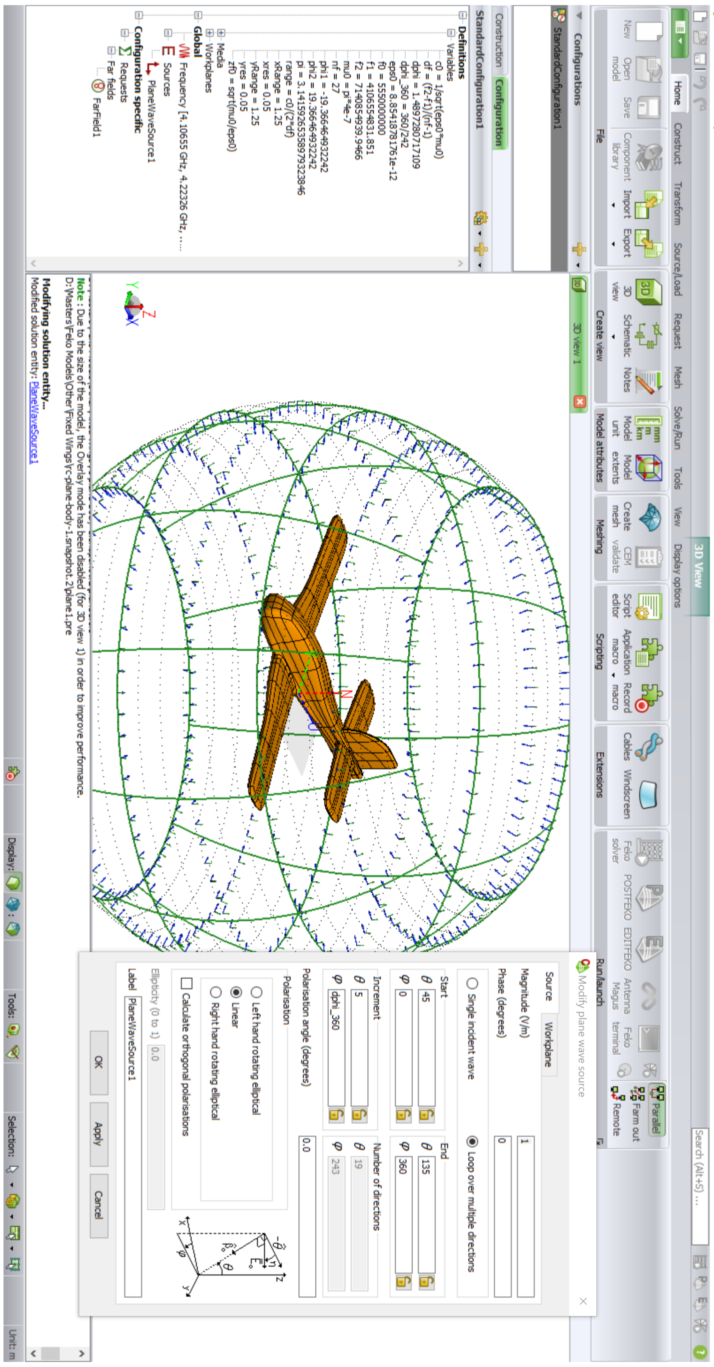
[112] X. Guo, “Xifengguo/capsnet-keras,” https://github.com/XifengGuo/CapsNet-Keras/tree/tf2.2, 2020.

[113] S. Adrián-Martínez, M. Bou-Cabo, I. Felis, C. D. Llorens, J. A. Martínez-Mora, M. Saldaña, and M. Ardid, "Acoustic signal detection through the cross-correlation method in experiments with different signal to noise ratio and reverberation conditions," in *International conference on Ad-Hoc Networks and wireless.* Springer, 2014, pp. 66–79.

[114] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *International conference on learning representations*, 2018.

Appendix A

Example FEKO simulation set-up



StandardConfiguration1

Definitions
- Variables
  - c0 = 1/sqrt(eps0*mu0)
  - df = (f2-f1)/(nf-1)
  - dphi = 1.4897280717109
  - dphi_360 = 360/242
  - eps0 = 8.85418781761e-12
  - f0 = 5550000000
  - f1 = 4106554831.851
  - f2 = 7140854939.9466
  - mu0 = pi*4e-7
  - nf = 27
  - phi1 = -19.366464932242
  - phi2 = 19.366464932242
  - pi = 3.14159265358979323846
  - range = c0/(2*df)
  - xRange = 1.25
  - xres = 0.05
  - yRange = 1.25
  - yres = 0.05
  - zf0 = sqrt(mu0/eps0)
- Media
- Workplanes

Global
- Frequency [4.10655 GHz, 4.22326 GHz, ...
- Sources
  - PlaneWaveSource1

Configuration specific
- Requests
  - Far fields
    - FarField1

Modify plane wave source

Source | Workplane

Magnitude (V/m)	1
Phase (degrees)	0

Single incident wave [✓ Loop over multiple directions]

	Start	End
θ	45	135
φ	0	360

	Increment	Number of directions
θ	5	19
φ	dphi_360	243

Polarisation angle (degrees) 0.0

Polarisation
Left hand rotating elliptical [✓ Linear] Right hand rotating elliptical

☐ Calculate orthogonal polarisations

Ellipticity (0 to 1) 0.0

Label PlaneWaveSource1

OK Apply Cancel

Note : Due to the size of the model, the Overlay mode has been disabled (for 3D view 1) in order to improve performance.
D:\Masters\Feko Models\Other\Fixed Wings\rc-plane-body-1.snapshot.2\plane1.pre

Modifying solution entity...
Modified solution entity: PlaneWaveSource1

Appendix B

ISAR processing script for parallel processing

```
%Constants
c = 2.997e8;

%Get input data
rcs = ReadFFE('D:\Masters\Feko Results\plane1_FarField1.ffe');

 field = fieldnames(rcs);
 field = field{1};
phi = unique(rcs.(field).Phi);
n_a_tot = length(phi);
 freq = unique(rcs.(field).freq);
 f_range = freq(end)-freq(1);
 n_f = length(freq);
 n_os = 1;
 n_a = 21;
 ang_range = phi(n_a)-phi(1);
theta = unique(rcs.(field).Theta);
n_theta = length(theta);

%set up data matrix
r1 = complex(zeros(n_f, n_a_tot, n_theta));
for m = 1:n_theta
    for n = 1:n_a_tot
        ind = (rcs.(field).Phi == phi(n)) & (rcs.(field).Theta == theta(
   m));
        r1(:, n, m) = rcs.(field).Re_Etheta(ind)+1j*rcs.(field).
   Im_Etheta(ind);
    end
end

%Split the data into multiple azimuth ranges
prev_size = 0;
for m = 1:n_theta
    data_temp = azSplit(r1(:, :, m), n_a, 1);
    if(~exist('data', 'var'))
        data = data_temp;
    else
        data = cat(3, data, data_temp);
    end
end

%Set up grid for interpolation to cartesian grid
fxStart = freq(1);
fyStart = tan(deg2rad(phi(1)))*fxStart;
fyEnd = tan(deg2rad(phi(n_a)))*fxStart;
fxEnd = sqrt(freq(end)^2 - fyEnd^2);

fx_d = (fxEnd - fxStart)/(n_f-1);
fy_d = (fyEnd - fyStart)/(n_a-1);

x_samp = fxStart:fx_d:fxEnd;
y_samp = fyStart:fy_d:fyEnd;

[X_samp, Y_samp] = meshgrid(x_samp, y_samp);

f_samp = sqrt(X_samp.^2 + Y_samp.^2);

ang_samp = rad2deg(atan(Y_samp./X_samp));

f_samp_ind = min(max((f_samp - freq(1))*n_f/f_range + 1, 1), n_f);
ang_samp_ind = min(max((ang_samp - phi(1))*n_a/ang_range + 1, 1), n_a);

%Generate noise-free example
ISAR = complex(zeros(n_os*n_f, n_os*n_a, size(data, 3)));
for i = 1:size(data, 3)
    ISAR(:, :, i) = GenerateISAR(data(:, :, i), n_os);
end

%save noise-free example
filename = sprintf('ISAR/plane1ISAR');
%save(filename, 'ISAR', '-v7.3');%, '-nocompression');

snrs = -24:3:24;
parfor j = 1:length(snrs)
    snr = snrs(j);
    n_examples = 10;
    ISAR = complex(zeros(n_os*n_f, n_os*n_a, n_examples*size(data, 3)));
    for example = 1:n_examples
        %add white gausian noise to the signal
        data_noisy = complex(zeros(n_f, n_a, size(data, 3)));
        for i = 1:size(data, 3)
```

```
            %add noise according to Xiang-Gen Xia, Genyuan Wang and V. C. Chen, "Quantitative SNR analysis for ISAR imaging using joint time-frequency analysis-Short time Fourier transform," in IEEE Transactions on Aerospace and Electronic Systems, vol. 38, no. 2, pp. 649-659, April 2002
            dat = data(:, :, i);
            beta = abs(dat).^2 >= 0.5*max(abs(dat).^2, [], 'all');
            Psignal = 10*log10(sum(abs(dat(beta)).^2)/sum(beta(:)));
            p = 10*log10(bandpower(dat(:)));
            data_noisy(:, :, i) = awgn(dat, snr, p);
        end

        for i = 1:size(data_noisy, 3)
            ISAR(:, :, i+(example-1)*size(data_noisy, 3)) = GenerateISAR(data_noisy(:, :, i), n_os);
        end
    end
    ISAR = ISAR./max(abs(ISAR), [], [1 2]);%normalise
    filename = sprintf('ISAR_norm/plane1ISAR_%ddB_norm', snr);
    parsave(filename, ISAR);%, '-nocompression');
end

f_inc = freq(2)-freq(1);
a_inc = (phi(2)-phi(1))*pi/180;
UA_r = c/(2*f_inc);
UA_cr = c/(freq(1)*2*a_inc);

range = (0:(n_os*n_f-1))/(n_os*n_f)*UA_r - UA_r/2;
crange = (0:(n_os*n_a-1))/(n_os*n_a)*UA_cr - UA_cr/2;

%plot ISAR image
figure;
imagesc(crange, -range, 1000*abs(ISAR(:, :, 1)));
colormap 'jet';
xlabel('Crossrange [m]');
ylabel('Range [m]');
colorbar;

function parsave(fname, ISAR)
  save(fname, 'ISAR', '-v7.3');
end

function ISAR = GenerateISAR(r1, n_os)
n_f = size(r1, 1);
n_a = size(r1, 2);
%HRR generation
window = hamming(n_f);
```

```
G_win = sum(window);
win = repmat(window,[1 n_a]);
HRR = fftshift(fft(win.*r1,n_os*n_f,1), 1)/G_win;

%ISAR generation
window = hamming(n_a).';
G_win = sum(window);
win = repmat(window,[(n_f*n_os) 1]);
ISAR = fftshift(fft(win.*HRR,n_os*n_a,2), 2)/G_win;
end
```

Listing B.1: ISAR processing script.

Appendix C

KNN implementation

```
function [y_pred] = KNN(x_new, X_ref, y_ref, k)
%KNN
%   Input Arguments:
%       x_new
%           The unlabelled observation to be classified.
%       X_ref
%           A list of known observations.
%       y_ref
%           The class labels corresponding to X_ref.
%       k
%           The number of nearest neighbours to consider
%
%   Output Arguments:
%       y_pred
%           The predicted class label of x_new.
%
%   Description:
%       Predicts the class of x_new according to the KNN algorithm.

%Calculate the distances.
d_vec = X_ref - x_new;
d = vecnorm(d_vec, 2, 2);

%Assign x_new to the modal class of the k nearest neighbours.
[~, ind] = sort(d);
y_pred = mode(y_ref(ind(1:k)));
end
```

Listing C.1: KNN.m

Appendix D

Capsule network code

```

'''
Capsule network code adapted from https://github.com/XifengGuo/CapsNet-Keras:

MIT License

Copyright (c) 2017 Xifeng Guo

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
'''
```

```python

import numpy as np
from sklearn.metrics import confusion_matrix
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras import backend as K
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
from utils import combine_images
from PIL import Image
from capsuleLayers import CapsLayer, PrimaryCaps, Length, Mask
import h5py as h5

K.set_image_data_format('channels_last')


def CapsNet(input_shape, n_class, routings, batch_size):

    dim_caps = 16
    x = layers.Input(shape=input_shape, batch_size=batch_size)
    conv1 = layers.Conv2D(filters=256, kernel_size=9, strides=1, padding
   ='valid', activation='relu', name='conv1')(x)
    primcaps = PrimaryCaps(conv1, dim_caps=8, n_chanels=32, kernel_size
   =9, strides=2, padding='valid')
    digitcaps = CapsLayer(num_caps=n_class, dim_caps=dim_caps, routings=
   routings, name='digitcaps')(primcaps)
    out_caps = Length(name='capsnet')(digitcaps)


    #Decoder
    y = layers.Input(shape=(n_class, ))
    masked_by_y = Mask()([digitcaps, y])
    masked = Mask()(digitcaps)

    decoder = models.Sequential(name='decoder')
    decoder.add(layers.Dense(512, activation='relu', input_dim=dim_caps
   * n_class))
    decoder.add(layers.Dense(1024, activation='relu'))
    decoder.add(layers.Dense(np.prod(input_shape), activation='sigmoid')
   )
    decoder.add(layers.Reshape(target_shape=input_shape, name='out_recon
   '))

    train_model = models.Model([x, y], [out_caps, decoder(masked_by_y)])
    eval_model = models.Model(x, [out_caps, decoder(masked)])

    noise = layers.Input(shape=(n_class, dim_caps))
    noised_digitcaps = layers.Add()([digitcaps, noise])
    masked_noised_y = Mask()([noised_digitcaps, y])
    manipulate_model = models.Model([x, y, noise], decoder(masked_noised_y))
    return train_model, eval_model, manipulate_model


def margin_loss(y_true, y_pred):
    L = y_true * tf.square(tf.maximum(0., 0.9 - y_pred)) + \
        0.5 * (1 - y_true) * tf.square(tf.maximum(0., y_pred - 0.1))

    return tf.reduce_mean(tf.reduce_sum(L, 1))


def train(model,  # type: models.Model
          data, args):
    """
    Training a CapsuleNet
    :param model: the CapsuleNet model
    :param data: a tuple containing training and testing data, like `((x_train, y_train), (x_test, y_test))`
    :param args: arguments
    :return: The trained model
    """
    # unpacking the data
    (x_train, y_train), (x_test, y_test) = data

    # callbacks
    log = callbacks.CSVLogger(args.save_dir + '/log.csv')
    checkpoint = callbacks.ModelCheckpoint(args.save_dir + '/weights-{epoch:02d}.h5', monitor='val_capsnet_accuracy',
                                           save_best_only=True, save_weights_only=True, verbose=1)
    lr_decay = callbacks.LearningRateScheduler(schedule=lambda epoch: args.lr * (args.lr_decay ** epoch))

    # compile the model
    model.compile(optimizer=optimizers.Adam(lr=args.lr),
                  loss=[margin_loss, 'mse'],
                  loss_weights=[1., args.lam_recon],
                  metrics={'capsnet': 'accuracy'})

    def train_generator(x, y, batch_size):
        train_datagen = ImageDataGenerator()
        generator = train_datagen.flow(x, y, batch_size=batch_size)
        while 1:
            x_batch, y_batch = generator.next()
            yield (x_batch, y_batch), (y_batch, x_batch)

    model.fit(train_generator(x_train, y_train, args.batch_size),
              steps_per_epoch=int(y_train.shape[0] / args.batch_size),
              epochs=args.epochs,
              validation_data=((x_test, y_test), (y_test, x_test)),
    batch_size=args.batch_size,
              callbacks=[log, checkpoint, lr_decay])

    model.save_weights(args.save_dir + '/trained_model.h5')
    print('Trained model saved to \'%s/trained_model.h5\'' % args.
    save_dir)

    from utils import plot_log
    plot_log(args.save_dir + '/log.csv', show=True)

    return model


def test(model, data, args):
    from sklearn.mixture import GaussianMixture as GMM
    from sklearn.metrics import confusion_matrix, f1_score
    from scipy.signal import correlate2d
    import pickle
    x_test, y_test = data
    print(x_test.shape)
    print(y_test.shape)
    print(np.argmax(y_test, 1))
    y_pred, x_recon = model.predict(x_test, batch_size=114)
    zncc = []
    xc = []
    x_recon_norm = x_recon - np.mean(x_recon, axis=(1, 2))[:, np.newaxis
    , np.newaxis]
    x_test_norm = x_test - np.mean(x_test, axis=(1, 2))[:, np.newaxis,
    np.newaxis]
    sigma_recon = np.std(x_recon, axis=(1, 2))
    sigma_test = np.std(x_test, axis=(1, 2))
    x_recon_norm = x_recon_norm/sigma_recon[:, np.newaxis, np.newaxis]
    x_test_norm = x_test_norm/sigma_test[:, np.newaxis, np.newaxis]
    for i in range(0, x_recon.shape[0]):
        xc = np.append(xc, correlate2d(np.squeeze(x_test_norm[i, :, :]),
     np.squeeze(x_recon_norm[i, :, :]), 'valid'))
    zncc = xc
    zncc_known = zncc[np.argmax(y_test, 1) <= 1]
    zncc_unknown = zncc[np.argmax(y_test, 1) > 1]
    histo = plt.figure()
    print(zncc.shape)
```

```python

    threshold = args.threshold
    if(threshold == 0):
        #find threshold
        threshold = get_threshold(zncc_unknown, zncc_known)
        print("threshold = %10f" % threshold)
        with h5.File(args.save_dir_snr + "/threshold.hdf5", "w") as f:
            f.create_dataset("threshold", data=threshold)

    b = np.histogram(zncc, bins=100)[1]
    plt.hist(zncc_unknown, b, color='red', alpha=0.5)
    plt.hist(zncc_known, b, color='blue', alpha=0.5)
    histo.savefig(args.save_dir_snr + "/histogram.png")

    y_pred_am = np.argmax(y_pred, 1)
    y_pred_am[zncc<threshold] = 2
    conf_mat = confusion_matrix(np.argmax(y_test, axis=1), y_pred_am)
    conf_mat_plt = plt.figure()
    plot_conf_mat(conf_mat)
    with open(args.save_dir_snr + "/zncc.pickle", 'wb') as f:
        pickle.dump(zncc, f)
    conf_mat_plt.savefig(args.save_dir_snr + "/conf_mat.png")
    print('-' * 30 + 'Begin: test' + '-' * 30)
    y_expected = np.argmax(y_test, 1)
    y_expected[y_expected > 1] = 2
    f1 = f1_score(y_expected, y_pred_am,  average="weighted")
    print('F1-score:', f1)
    print('Test acc:', sum(y_pred_am == y_expected)/y_test.shape[0])

    img = combine_images(np.concatenate([x_test[16:16856:421], x_recon
   [16:16856:421]]), height=8)
    image = img * 255
    Image.fromarray(image.astype(np.uint8)).save(args.save_dir_snr + "/
   real_and_recon.png")
    print()
    print('Reconstructed images are saved to %s/real_and_recon.png' %
   args.save_dir_snr)
    print('-' * 30 + 'End: test' + '-' * 30)
    plt.imshow(plt.imread(args.save_dir_snr + "/real_and_recon.png"))

    #return F1 score
    return (f1)

def get_threshold(dist1, dist2):
    #this is not very efficient, but should work
    overlap_max = np.max(dist1)
    overlap_min = np.min(dist2)
    overlap_1 = np.sort(dist1[dist1 > overlap_min])
    overlap_2 = np.sort(dist2[dist2 < overlap_max])
    thresholds = np.linspace(overlap_min, overlap_max, 1000)
    costs = np.full_like(thresholds, np.Infinity)
    i = 0
    for threshold in thresholds:
        costs[i] = np.sum(overlap_1 > threshold) + np.sum(overlap_2 < threshold)
        i+=1
    return thresholds[np.argmin(costs)]

def plot_conf_mat(conf_mat):
    conf_mat_new = conf_mat[:, 0:3]
    n_classes = 4
    import seaborn as sns
    mask = [[1, -1, -1],\
            [-1, 1, -1],\
            [-1, -1, 1],\
            [-1, -1, 1]]
    conf_color = conf_mat_new*mask
    ax = sns.heatmap(conf_color, cmap = 'coolwarm_r', annot=conf_mat_new, fmt="d",\
    vmin=-y_test.shape[0]/n_classes, vmax=y_test.shape[0]/n_classes, snap=True,\
        linewidths=1, linecolor='k', xticklabels=["Drone", "Missile", "Other"],\
            yticklabels=["Drone", "Missile", "UFO", "Fixed-wing"], cbar=False, square=False)
    ax.set_xlabel("Predicted label", size=12)
    ax.axvline(x=3,color='k',linewidth=2)
    ax.axhline(y=4,color='k',linewidth=2)
    #ax.set_xticklabels(["Drone", "Missile", "Other"], rotation=30)
    ax.set_ylabel("True label", size=12)
    ax.set_title("Confusion Matrix")
    #plt.show()

def load_isar(snr):
    import h5py as h5
    import math
    os.chdir("D:\Masters\Classifiers\ISAR-Classification")
    n_examples = 3
    n_classes = 2
    obs_per_class = 4598*2
    X = np.empty((0, 32, 32), dtype = [('real', '<f8'), ('imag', '<f8')])
    y = np.empty((0,32, 32), int)
    indx = 0
    for c_name in ("droneISAR_32_"+str(snr)+"dB_norm", "missileISAR_32_"
   +str(snr)+"dB_norm"):
        with h5.File("ISAR_norm\\"+c_name+".mat", 'r') as f:
            c = f['ISAR']
            s = 13794
            X = np.append(X, c[0:s, :, :], axis = 0)
            y = np.append(y, np.full((s, 1), indx))
            indx = indx + 1
    with h5.File("shuffle_ind.mat", 'r') as f:
        shuffle_ind = np.squeeze(np.array(f.get('shuffle_ind'), int))-1

    for c_name in ("ufoISAR_32_"+str(snr)+"dB_norm", "plane1ISAR_32_"+
   str(snr)+"dB_norm"):
        with h5.File("ISAR_norm\\"+c_name+".mat", 'r') as f:
            c = f['ISAR']
            s = 13794
            X = np.append(X, c[0:s, :, :], axis = 0)
            y = np.append(y, np.full((s, 1), indx))
            indx = indx + 1

    X = X[:, :, :, np.newaxis]
    X = np.sqrt(np.square(X['real']) + np.square(X['imag']))
    train_ind = np.zeros(242, dtype=bool)
    train_ind[shuffle_ind[1:math.floor(0.7*242)]] = True
    train_ind = np.tile(train_ind, 19*n_classes*n_examples)
    train_ind_ext = np.append(train_ind, np.full_like(train_ind, False))
    test_ind = ~np.append(train_ind, train_ind)
    X_train = X[train_ind_ext]

    #convert y to 1-hot encoding
    y = tf.one_hot(y, 4)
    y_train = y[train_ind_ext]
    X_val = X[test_ind]
    y_val = y[test_ind]
    del X
    os.chdir("D:\Masters\Classifiers\ISAR-Classification\CapsNet")
    return (X_train, y_train), (X_val, y_val)

if __name__ == "__main__":
    import os
    import argparse
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras import callbacks

    # setting the hyper parameters
    parser = argparse.ArgumentParser(description="Capsule Network for
   ISAR image classification.")
```

```
    parser.add_argument('--epochs', default=10, type=int)
    parser.add_argument('--batch_size', default=114, type=int)
    parser.add_argument('--lr', default=0.001, type=float,
                        help="Initial learning rate")
    parser.add_argument('--lr_decay', default=0.9, type=float,
                        help="The value multiplied by lr at each epoch. Set a larger value for larger epochs")
    parser.add_argument('--lam_recon', default=0.392, type=float,
                        help="The coefficient for the loss of decoder")
    parser.add_argument('-r', '--routings', default=3, type=int,
                        help="Number of iterations used in routing algorithm. should > 0")
    parser.add_argument('--debug', action='store_true',
                        help="Save weights by TensorBoard")
    parser.add_argument('--save_dir', default='./result')
    parser.add_argument('-t', '--testing', action='store_true',
                        help="Test the trained model on testing dataset")
    parser.add_argument('-w', '--weights', default=None,
                        help="The path of the saved weights. Should be specified when testing")
    parser.add_argument('--threshold', default=0, type=float,
                        help="The threshold value to use when separating known/unknown classes")
    args = parser.parse_args()
    print(args)

    if not os.path.exists(args.save_dir):
        os.makedirs(args.save_dir)

    # load data
    (x_train, y_train), (x_test, y_test) = load_isar(24)

    # define model
    model, eval_model, manipulate_model = CapsNet(input_shape=x_train.shape[1:],
                                                  n_class=len(np.unique(np.argmax(y_train, 1))),
                                                  routings=args.routings,
                                                  batch_size=args.batch_size)
    model.summary()

    # train or test
    if args.weights is not None:  # init the model weights with provided one
```

```python
        model.load_weights(args.weights)
    if not args.testing:
        train(model=model, data=((x_train, y_train), (x_test, y_test)), args=args)
    else:  # as long as weights are given, will run testing
        if args.weights is None:
            print('No weights are provided. Will test using random initialized weights.')
        eval_model.load_weights(args.weights)
        snrs = range(24, -25, -3)
        f1 = []
        for snr in snrs:
            (x_train, y_train), (x_test, y_test) = load_isar(snr)
            args.save_dir_snr = args.save_dir + "/" + str(snr) + "dB"
            if not os.path.exists(args.save_dir_snr):
                os.makedirs(args.save_dir_snr)
            f1.append(test(model=eval_model, data=(x_test, y_test), args=args))
        plt.plot(snrs, f1)
        plt.show()
        with h5.File(args.save_dir + "/F1_vs_snr.hdf5", "w") as f:
            f.create_dataset("SNR", data=snrs)
            f.create_dataset("f1", data=f1)
```

Listing D.1: Capsule network code. (Adapted from [112])