One-class classification for low resolution targets discrimination with limited supervision in pulse Doppler radars

Martin Bauw

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à Mines Paris

One-class classification for low resolution targets discrimination with limited supervision in pulse Doppler radars

Classification mono-classe pour la discrimination de cibles de radars Doppler pulsés à faible résolution avec faible supervision

Soutenue par

Martin BAUW

Le 18 janvier 2023

École doctorale n°621

Ingénierie des Systèmes, Matériaux, Mécanique, Énergétique

Spécialité

Morphologie Mathématique

Composition du jury :

Florence TUPIN Professeure, Télécom Paris, LTCI, Institut Polytechnique de Paris	*Présidente*
Marius KLOFT Professor, TU Kaiserslautern	*Rapporteur*
Jean-Philippe OVARLEZ Directeur de Recherche, ONERA	*Rapporteur*
Claude ADNET Expert radar, Thales LAS	*Examinateur*
Santiago VELASCO-FORERO Chargé de Recherche, Mines Paris, PSL University	*Examinateur*
Jesus ANGULO Directeur de Recherche, Mines Paris, PSL University	*Directeur de thèse*

MINES PARIS | PSL

Contents

Acronyms

AD	Anomaly Detection
AE	Autoencoder
AIM	Affine-invariant Metric
AR	Autoregressive
ATR	Automatic Target Recognition
CFAR	Constant-False-Alarm Rate
CNN	Convolutional Neural Network
CVNN	Complex-Valued Neural Network
DFT	Discrete Fourier Transform
ECG	Electrocardiogram
EEG	Electroencephalogram
EOS	End-of-Sequence
FCAE	Fully Convolutional Autoencoder
FCN	Fully Convolutional Network
FCNN	Fully Convolutional Neural Network
FFT	Fast Fourier Transform
GCL	Graph Convolutional Layer
GMM	Gaussian Mixture Model
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
HPD	Hermitian Positive Definite
k-NN	k-Nearest Neighbors
LEM	Log-Euclidean Metric
LSTM	Long Short-Term Memory
MTI	Moving Target Indicator
NBC	Naive Bayes Classifier
OCC	One-Class Classification
OOD	Out-of-Distribution
OODD	Out-of-Distribution Detection
PCA	Principal Component Analysis
PDR	Pulse Doppler Radar
PRF	Pulse Repetition Frequency
PRI	Pulse Repetition Interval
RADAR	Radio Detection and Ranging
RCS	Radar Cross-Section
RNN	Recurrent Neural Network
RPD	Random Projection Depth
RPO	Random Projection Outlyingness
RVNN	Real-Valued Neural Network
SAD	Semi-supervised Anomaly Detection
SAR	Synthetic Aperture Radar
Seq2seq	Sequence-to-sequence
SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise Ratio
SONAR	Sound Navigation and Ranging
SOS	Start-of-Sequence
SOTA	State-of-the-Art
SPD	Symmetric Positive Definite
SSL	Self-supervised Learning
SVDD	Support Vector Data Description
SVM	Support Vector Machine
XAI	Explainable Artificial Intelligence

List of Symbols

$\mathcal{O}$	Outlyingness score
Φ	Function defined by a neural network (except for GELU in 2.2.1)
$\mathbb{R}$	Set of real numbers
$\|\|\cdot\|\|_F^2$	Frobenius norm
λ	(Chap. 1) Electromagnetic wavelength
λ	(Other chapters) Loss term weight
H	Number of range cells in the neighborhood
N	Number of pulses in a burst
Q	Embedding size
$\mathcal{S}_*^+$	Manifold of SPD matrices
$\mathbb{R}_*^{d_k \times d_{k-1}}$	Manifold of full-rank matrices

Avant-propos

Cette thèse de doctorat s'inscrit dans le cadre du développement de l'aide à la décision basé sur des réseaux de neurones et ce qui est communément appelé de l'intelligence artificielle (IA). L'aide à la décision permet l'intervention d'un choix de nature algorithmique, choix modéré par la prise de décision finale d'un opérateur. Dans le cadre d'un système radar et de la présente thèse de doctorat, l'opérateur est l'opérateur radar, et l'aide à la décision consiste en un score traduisant une proximité avec certains types de cibles qui est mis à la disposition de l'opérateur pour interprétation. L'opérateur est ainsi libre de prendre en compte l'information apportée par l'IA. Cette approche se distingue ainsi de systèmes autonomes d'ores et déjà indésirables pour des applications militaires [190], et garantit l'identification d'une responsabilité humaine.

Le contexte sociétal de cette thèse est celui de l'explosion de l'intérêt pour les diverses méthodes assimilées à une forme d'intelligence artificielle. Cette explosion paraît naturelle entre autres du fait de l'omniprésence de la collection des données numériques et du constant développement des capacités de calculs nécessaires à leur valorisation. L'importance future accordée à l'IA fait de cette dernière une technologie stratégique et un enjeu de souveraineté [56, 58]. L'intégration d'un réseau de neurones à une chaîne de traitement radar proposée par cette thèse découle ainsi des axes prioritaires du ministère des Armées [60, 59] et fait écho à la place réservée à l'IA dans la boussole stratégique [180] de l'Union Européenne.

De nombreux développements de l'intelligence artificielle sont de nature duale: proposer une méthode innovante de traitement de signal ou de traitement d'image répond parfois aussi bien à une problématique civile qu'à une problématique militaire. Un exemple concret de développement dual consiste en la segmentation d'images s'appuyant sur des réseaux de neurones: le traitement d'image innovant peut s'appliquer à une image visant à l'aide au diagnostic médical, mais aussi à une image satellite porteuse de renseignement. Le développement mis en avant par ce travail doctoral participe à cette dualité, la discrimination de signaux à échantillonnage variable, faible résolution et faible supervision ne se limitant pas aux systèmes militaires. La dualité des avancées dans l'IA se distingue de celle d'autres technologies en ceci que les avancées proviennent en premier lieu du monde civil [190].

Remerciements

Je remercie pour commencer le ministère des Armées pour le financement Cifre-Défense qui m'a permis de mener ces travaux de thèse de doctorat. La possibilité d'interagir régulièrement avec la Direction générale de l'Armement (DGA) à travers l'Agence de l'innovation de défense (AID) s'est révélée enrichissante et complémentaire à mon encadrement académique et industriel. Je remercie également l'Association nationale de la recherche et de la technologie (ANRT) qui encadre les financements Cifre. Je tiens ensuite à remercier mes encadrants académiques et industriels, en particulier Jesus, Santiago et Claude, pour m'avoir donné cette opportunité de découvrir le monde de la recherche appliquée et l'expérience de l'enseignement. Votre bienveillance et votre patience à mon égard ont été grandement appréciées, et votre disponibilité m'a permis de satisfaire une partie de la soif d'apprentissage motivant cette aventure doctorale. Je remercie les membres de mon jury de thèse, et notamment les rapporteurs, pour l'attention qu'ils ont portée à mon travail. Je suis également très reconnaissant à mes collègues de Thales et de l'école des Mines qui ont su souffrir mes nombreuses questions et interruptions. Parmi ces derniers, j'ai une pensée particulière pour Samy et Daniel dont les explications m'ont beaucoup instruit, et pour Anne-Marie dont le soutien fut de grande valeur.

Je suis par ailleurs très reconnaissant envers mes amis de ou rencontrés à l'école normale supérieure Paris-Saclay, qui m'ont apporté de nombreuses et précieuses réponses sur des sujets techniques et scientifiques tout au long de cette expérience doctorale, aussi bien autour d'une pinte que sur notre glorieux serveur IRC. Je tiens en particulier à remercier Lev-Arcady, Pollion et Talbot, dont les explications de notions mathématiques et algorithmiques m'ont été très utiles. Je finis ces remerciements par ceux dédiés à ma famille ainsi qu'à mes deux soutiens du quotidien Martin et Eloïse, piliers indispensables et inébranlables de ces trois ans de tribulations doctorales.

List of publications

Conferences

- 2020 IEEE Radar Conference: "From unsupervised to semi-supervised anomaly detection methods for HRRP targets" Martin Bauw, Santiago Velasco-Forero, Jesus Angulo, Claude Adnet and Olivier Airiau.
- 2022 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD): "Near out-of-distribution detection for low-resolution radar micro-Doppler signatures" Martin Bauw, Santiago Velasco-Forero, Jesus Angulo, Claude Adnet and Olivier Airiau.

Workshops

- 2020 5th SONDRA workshop (canceled due to covid): "A review of deep and shallow anomaly detection methods for HRRP targets" Martin Bauw, Santiago Velasco-Forero, Jesus Angulo, Claude Adnet and Olivier Airiau.
- 2021 ICML workshop on Uncertainty & Robustness in Deep Learning (UDL): "Deep random projection outlyingness for unsupervised anomaly detection" Martin Bauw, Santiago Velasco-Forero, Jesus Angulo, Claude Adnet and Olivier Airiau.
- 2022 5th SONDRA workshop (retracted due to an intellectual property conflict): "Complex-valued graph neural networks for pulse Doppler radar targets discrimination with small bursts" Martin Bauw, Santiago Velasco-Forero, Jesus Angulo, Claude Adnet and Olivier Airiau.

White paper

- ICLR 2021 white paper of the workshop on geometrical and topological representation learning: "2021 ICLR challenge for computational geometry & topology: design and results" Nina Miolane et al.

Chapters summary

Chapter 1: Pulse Doppler radars, discrimination task and solution overview

The first chapter introduces the radar discrimination problem addressed by the thesis and the machine learning tools used in the solution put forward. The specificity of the discrimination developed in terms of available supervision and targets representation resolution is described. The data format processed, inherited from the radar systems and consisting in complex-valued signals registered over brief time segments with a sampling covering few time steps, is illustrated and enriched with a neighborhood information. The link between the radar constraints from which the originality of this work stems and air surveillance radars with rotating antennas is established. A breakdown of the radar targets discrimination task into a two steps filter is proposed, where a first step encodes representations in a shared representation space, and a second step implements a low-supervision discrimination. The latter step is based on a one-class classification which can be understood as an anomaly detection method.

Chapter contribution:

- Enriched hit format definition to take into account a fixed-size neighborhood information centered on the range cell responsible for the hit detection.
- Two-steps processing proposition, encoding and one-class classification-based discrimination, adapted to the targets representations diversity and resolution, but also to the weak supervision of the separation addressed.

Chapter 2: Encoding IQ signals

The second chapter proposes different options for the implementation of the first encoding step of the filter proposed in the first chapter. Encoding approaches, including neural networks architectures adapted to representation learning, at the scale of one or a neighborhood of range cells are put forward. The combination of representations over a neighborhood of range cells through a graph is suggested. The encoding methods mentioned harness either an autoregressive model, or a neural network that may be trained with a generative or non-generative objective. The generative nature of a neural network is associated with the training potential of the encoding in a weak supervision context, the reconstruction error making it possible to define a training loss without labels. The preferred option in this work being a neural network and the data being complex-valued IQ signals, notions specific to complex-valued neural networks and to the optimization of the latter are introduced. The explainability of the neural networks defined is discussed thanks to the existing equivalence between finite impulse response filters and the one-dimensional convolution, the latter constituting the first layer of the non-recurrent neural networks studied. Preliminary results are presented and suggest the relevance of

an encoding architecture which trains a complex-valued neural network to encode the content of a single range cell described heterogeneously in a shared representation space.

Chapter contribution:

- Proposal of a diversity of approaches for the enriched hit format encoding covering a neighborhood of range cells using neural networks and autoregressive models.
- Integration of a combination of existing tools stemming from the machine learning community for the encoding of IQ signals with a varying sampling: complex-valued neural networks, graph neural networks, recurrent models, loss functions for different levels of training supervision.
- Preliminary results presentation for the encoding of single range cells with simulated pulse Doppler radar data.

Chapter 3: One-class classification for radar targets discrimination

This chapter reviews different one-class classification approaches from the literature and suitable for the implementation of the second step of the radar targets filter proposed in the first chapter. Achieving the radar targets separation with a one-class classification appears relevant due to the low supervision context of the separation conducted: all the targets classes are not necessarily known, and the identified classes are not necessarily described by representative data sets. One-class classification enables the definition of a reference class based on a limited number of labeled data points in order to produce a score translating into the degree of membership to that reference class. For the radar operator's mission, such a score can thus help decide whether a target is close enough to an arbitrary set of classes possibly gathering diverse targets. The definition of an arbitrarily complex reference class leads to the consideration of "near" and "far" out-of-distribution detection concepts which emphasize the difficulty of separating targets that may appear semantically close. A one-class classification method from the literature based on a neural network gathering output representations from the reference class next to a latent centroid is modified. The latent centroid here defines a location estimator of the reference class. Whereas the method from the literature minimizes the Euclidean distance between the output representations of the training data and the latent centroid, the proposed modified version minimizes an outlyingness measure computed with a set of random projections. A set of normalized distances, each distance being defined through a predetermined random projection, allows for the definition of a robust outlyingness measure for multidimensional representations. This measure is directly inspired by a stochastic approximation of a statistical depth, and makes it possible to work with ellipsoids without computing covariance matrices and their inverse to yield a distance. The intuition behind the use of random projections is illustrated and used, as is the Euclidean distance in the literature, to define unsupervised and semi-supervised one-class classifications.

Chapter contribution:

- Proposal of a modified version of an existing one-class classification method. The modified version replaces a Euclidean distance with an outlyingness measure directly inspired by a stochastic approximation of a statistical depth.

- Performances comparison and illustration for different one-class classification methods applied to several data sets, including different kinds of radar data.

Chapter 4: One-class classification for encoded hits

This chapter aims at evaluating the relevance of the two-steps processing proposed in this thesis. It presents preliminary results of one-class classification methods applied to enriched hits as defined in the first chapter and encoded following an approach described in the second chapter. This chapter is thus dedicated to the integration of the approaches and intuitions detailed in the preceding chapters to achieve the filter motivating this work. The range cells neighborhoods defining the enriched input format are artificial. They are built from combinations of individually encoded range cells. Such neighborhoods still allow for the construction of a relevant data set containing different classes of local correlations in terms of Doppler content, these correlations classes defining precisely the radar targets modes whose separation is expected. The few preliminary results presented are not conclusive enough to allow for a proper evaluation of the graph neural network-based enriched hit encoding approach. These results thus lead to the suggestion of follow-up experiments.

Chapter contribution:

- Preliminary results presentation regarding the encoding and the discrimination of enriched hits, combining the methods and intuitions proposed in the preceding chapters. Due to the inconclusive nature of the few results gathered, suggestions are made for follow-up experiments.

Chapter 5: Conclusion and perspectives

This last chapter concludes the thesis with a reminder of the proposals put forward in the manuscript and with suggestions regarding the further developments to be prioritized.

Chapter contribution:

- Suggestions regarding possible further developments of the proposed approach.

Résumé des chapitres

Chapitre 1: Radar Doppler pulsé, discrimination recherchée et aperçu de la solution proposée

Le premier chapitre introduit le problème de discrimination radar traité par la thèse et les outils issus de l'apprentissage automatique utilisés dans la solution mise en avant. La spécificité de la discrimination recherchée en termes de supervision disponible et de résolution de représentation des cibles est explicitée. Le format des données à traiter, hérité de systèmes radar et consistant en des signaux à valeurs complexes représentés sur de brefs segments temporels avec un échantillonnage variable sur peu d'instants, est illustré et enrichi par une information de voisinage. Le lien entre les contraintes radar faisant l'originalité de ce travail et les radars de surveillance aérienne à antenne tournante est établi. Un découpage de la tâche de discrimination de cibles radar en un filtrage en deux étapes est proposé, avec une première étape d'encodage des représentations dans un espace de représentation commun, et une seconde étape de discrimination à faible supervision. L'étape d'encodage est définie par un réseau de neurones, comme peut l'être l'étape de discrimination. Cette dernière est basée sur une classification mono-classe qui peut-être perçue comme une méthode de détection d'anomalie.

Contribution du chapitre:

- Définition d'un format hit enrichi qui reprend une information de voisinage de taille fixe centrée sur la case distance qui a déclenché la détection d'un hit.
- Proposition du traitement en deux étapes, encodage puis discrimination par classification mono-classe, adapté à la diversité ainsi qu'à la résolution des représentations des cibles, mais aussi à la faible supervision de la séparation recherchée.

Chapitre 2: Encodage de signaux IQ

Le deuxième chapitre propose différentes options pour l'implémentation de la première étape d'encodage du filtre proposé dans le premier chapitre. Des encodages, notamment des architectures de réseaux de neurones adaptées à l'apprentissage de représentation, à l'échelle d'une ou d'un voisinage de plusieurs cases distance sont mis en avant. La combinaison des représentations d'un voisinage de cases distance par le biais d'un graphe est avancée. Les méthodes d'encodage évoquées exploitent soit un modèle autorégressif, soit un réseau de neurones qui peut être décliné en une version générative et une version non-générative. Le caractère génératif d'un réseau de neurones est associé au potentiel d'entraînement de l'encodage dans un contexte de faible supervision, l'erreur de reconstruction pouvant définir une fonction coût sans aucun label. La solution privilégiée par ce travail doctoral étant un réseau de neurones et les données étant des signaux IQ à valeurs complexes, des notions propres aux réseaux de neurones à paramètres complexes

et à l'optimisation de ces derniers sont introduites. L'explicabilité des réseaux de neurones manipulés est discutée grâce à l'équivalence entre un filtre à réponse impulsionnelle finie et une convolution à une dimension, cette dernière définissant la première couche des réseaux non-récurrents exploités. Des résultats préliminaires sont proposés et suggèrent la pertinence d'une architecture d'encodage qui entraîne un réseau de neurones à valeurs complexes à encoder le contenu d'une case distance unique décrite de manière hétérogène dans un espace de représentation commun.

Contribution du chapitre:

- Proposition d'une diversité d'approches pour l'encodage du format hit enrichi couvrant un voisinage de cases distance avec des réseaux de neurones et des modèles autorégressifs.
- Intégration d'une combinaison d'outils existants de la communauté de l'apprentissage automatique pour l'encodage de signaux IQ à l'échantillonnage variable: réseaux de neurones à valeurs complexes, réseaux de neurones pour graphe, modèles récurrents, fonctions coût pour différents niveaux de supervision pendant l'entraînement.
- Présentation de résultats préliminaires pour l'encodage individuel de cases distance avec des données de radar Doppler pulsé simulées.

Chapitre 3: Classification mono-classe pour la discrimination de cibles radar

Ce chapitre passe en revue différentes méthodes de classification mono-classe issues de la littérature et adaptées à l'implémentation de la seconde étape du filtrage proposé dans le premier chapitre. Aborder la séparation de cibles radar avec une classification mono-classe apparaît pertinent du fait du contexte de faible supervision de la séparation réalisée: l'ensemble des classes de cibles n'est pas forcément connu, et les classes identifiées n'étant pas nécessairement décrites par des jeux d'échantillons représentatifs. La classification mono-classe autorise la définition d'une classe de référence à partir d'une quantité limitée de points de donnée labellisés, cela de manière à produire un score traduisant l'appartenance ou non à cette classe de référence. Pour la mission d'un opérateur radar, ce score peut ainsi fournir une aide à la décision en levant ou non une alerte pour des cibles trop proches ou trop loin d'une classe de référence arbitraire, celle-ci pouvant idéalement rassembler différents types de cible. La définition d'une classe de référence arbitrairement complexe mène à une réflexion quant aux notions de détection hors-distribution "proche" et "lointaine" qui traduisent la difficulté de séparation relative à la proximité sémantique des cibles traitées. Une méthode de classification mono-classe tirée de la littérature et basée sur un réseau de neurones qui concentre les représentations de sortie appartenant à la classe de référence autour d'un centroide latent de cette même classe est déclinée. Là où la méthode de la littérature minimise une distance Euclidienne entre les représentations de sortie des données d'entraînement et le centroide latent, la déclinaison proposée par cette thèse minimise une mesure d'anormalité tirée d'un jeu de projections aléatoires. Un ensemble de distances normalisées, chaque distance étant définie par une projection aléatoire fixée, permet de définir une mesure robuste de l'anormalité pour des représentations à plusieurs dimensions. La mesure en question est directement inspirée par une approximation stochastique de la notion de profondeur statistique, et permet de travailler avec des distributions suivant des ellipsoïdes tout en s'affranchissant du calcul d'une matrice de covariance, puis de son inverse pour l'obtention d'une distance. L'intuition derrière l'emploi de projections aléatoires

est illustrée et utilisée, comme le fut la distance Euclidienne dans la littérature, pour définir une classification mono-classe non supervisée puis semi-supervisée.

Contribution du chapitre:

- Proposition d'une déclinaison d'une méthode de classification mono-classe tirée de la littérature. La déclinaison proposée remplace une distance Euclidienne par une mesure d'anormalité directement inspirée d'une approximation stochastique d'une profondeur statistique.
- Comparaison et illustration des performances de différentes méthodes de classification mono-classe sur plusieurs jeux de données, y compris sur des données radar de différentes natures.

Chapitre 4: Classification mono-classe pour hits encodés

Ce chapitre vise à évaluer la pertinence du traitement en deux étapes proposé par cette thèse. Il présente des résultats préliminaires d'application de méthodes de classification mono-classe à des hits au format enrichi tel que défini dans le premier chapitre et encodés suivant une méthode proposée dans le deuxième chapitre. Il s'agit donc ici de l'intégration des méthodes et des intuitions présentées dans les chapitres précédents pour aboutir au filtre motivant cette thèse. Les voisinages de cases distance qui définissent le format d'entrée enrichi sont artificiels. Ils sont élaborés à partir de recombinaisons de cases distance individuelles encodées. Ces voisinages permettent néanmoins l'élaboration d'un jeu de données pertinent qui contient différentes classes de corrélations locales en termes de contenu Doppler, ces classes de corrélations définissant précisément les modes de cibles radar qu'il s'agit de discriminer. Les quelques résultats préliminaires présentés ne permettent cependant pas de conclure quant à la pertinence de l'encodage du format hit enrichi basé sur un réseau de neurones pour graphe, et mène à l'énonciation d'une suite à donner aux expériences menées.

Contribution du chapitre:

- Présentation de résultats préliminaires pour l'encodage et la discrimination de hits au format enrichi combinant l'ensemble des méthodes et des intuitions proposées dans les chapitres antérieurs. Face à l'apparence peu concluante des quelques résultats obtenus, des recommandations sur la suite des expériences à mener sont énoncées.

Chapitre 5: Conclusion et perspectives

Ce dernier chapitre conclut la thèse en rappelant les propositions avancées dans ce manuscrit et en proposant des axes de développement à privilégier pour la poursuite des travaux.

Contribution du chapitre:

- Mise en avant d'axes de développement futurs à privilégier pour l'approche proposée.

Chapter 1

Pulse Doppler radars, discrimination task and solution overview

This thesis aims at proposing novel machine learning-powered processing to discriminate more efficiently between radar targets using Doppler features, with a focus on air surveillance radars as sensors and small and slow objects as targets. The solution put forward will emphasize the choice of one-class classification for low-supervision discrimination and will take into account the resolution constraint associated with pulse Doppler air surveillance radars. This chapter describes the motivation of the thesis from a radar perspective and presents an overview of the answer proposed to tackle the problem addressed. Since this manuscript is written both for radar and machine learning engineers and researchers, footnotes and reminders are present to make notions specific to one field understandable to people from the other field (see for example the brief reminder dedicated to deep learning in 1.3.4, and the one dedicated to signal processing in 1.3.5).

1.1 The radar motivation

Intuitively, radar amounts to sending energy in a certain direction and detecting objects with the energy reflected back. Whereas sonar uses sound, *i.e.* a mechanical wave, radar harnesses electromagnetic waves. Radar mostly relies on two attributes to discriminate between targets: the target apparent size and its velocity with respect to the sensor, *i.e.* its radial velocity. When the radar itself sends energy that will reflect on targets, one considers active radars. On the other hand, when the radar only harnesses reflections of signals it has no control over, radars are said to be passive. The same distinction exists between active and passive sonar. One of the key advantages of passive sensors is their electromagnetic stealth, the reliance on external transmitters remaining a challenge. This work concerns itself with active radars, whose transmitted signal is thus deliberately designed to characterize potential targets. The fundamental equation behind the electromagnetic remote sensor intuition is the radar equation which defines the power received by the antenna P_R as a function of the transmitted power P_T and wavelength λ, the antenna gain G, the distance to the target R, and the effective target area σ called radar-cross section (RCS) [113]:

$$P_R = \frac{P_T G^2 \lambda^2 \sigma}{(4\pi)^3 R^4} \tag{1.1}$$

Radars come in all shapes and sizes. Antenna design, transmit power, waveform,

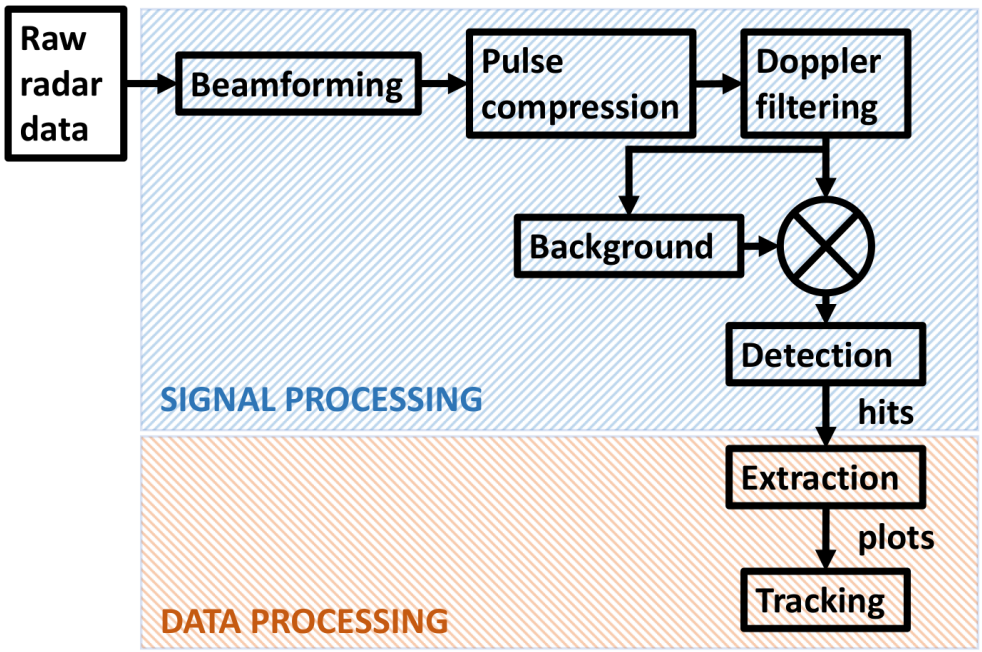


Figure 1.1: Radar processing pipeline. The processing proposed uses the output of the detection step, the hits, as input. The processing steps can be describe in simple terms as follows: before detection, there is beamforming to locate the target direction, pulse compression to increase range resolution and Doppler filtering to generate velocity descriptions of targets. The background block consists in a filter to discard potential detections that are too consistent with the weather. The detection step uses a threshold to define one hit each time a backscatter deserves discriminative processing, the extractor step refines the hits by agglomerating them when relevant, and the tracking step creates tracks outlined by successive plots. The steps located between the beamforming and detection blocks, the two latter being included, constitute the actual signal processing pipeline within the radar receiver chain.

and other sensor parameters widely vary depending on the radar mission. This thesis concentrates on air surveillance radars with frequent updates and long range constraints. Historically, many relevant targets to civilian and military radars dedicated to air surveillance could count on the fact that the targets they needed to detect were either relatively big, or relatively fast, or both at the same time. This allowed for the elimination of numerous clutter-related false alarms, where clutter consists of undesirable reflected signals. Indeed, such targets allow the sensor to immediately discard small and slow detections, which populate the target domain containing most clutter. However, with the advent of commercially available drones accessible to the general public, small and slow detections can not be as easily removed anymore. In addition to a possibly very discrete radar cross-section, a drone can have a null bulk speed while airborne and active. Air surveillance radars should now ideally be able to cherry pick relevant small and slow targets, such as drones, before discarding the latter category altogether.

In order to ensure the detection of relevant small and slow targets, the radars for which this work is intended use decreased detection thresholds at the *Detection* stage of the pipeline depicted on Fig. 1.1, letting through potentially relevant targets as well as clutter. The fact that supplementary clutter will be part of the detections let through by the lowered thresholds is unavoidable, the small and slow targets domain usually containing a lot of unwanted backscatter. Lowering the detection thresholds can thus lead to a dramatic increase in the number of detections, each of which requires processing in the radar processing pipeline. This increase in detections can in turn lead to an excessive amount of false alarms (false positives), imposing an additional false alarm filter. This false alarm filter should be implemented as early as possible once the additional detections are admitted into the processing pipeline, this to avoid spending scarce computing resources and the operator's attention on pointless detections.

The proposed targets discrimination aims at alleviating the false alarms excess and will therefore take the detections called *hits* as input, and filter the latter before passing on the filtered detections to the *Extraction* step, as suggested on Fig. 1.1. This positioning choice in the radar processing pipeline makes this work complementary to existing approaches working on subsequent detection representations such as aggregated hits, *i.e. plots*, and *tracks* [205]. Hybrid approaches also combine several of the aforementioned representations in setups falling within the domain of sensor fusion [132]. Whereas processing plots could seem similar since features are similar to the ones attached to hits, discriminating between tracks moves the targets separation task away from radio signal processing and closer to common time-series or spatial trajectories processing.

It is important to realize that processing the excess of false alarms stemming from lowered thresholds only at these later stages is ill-advised, since it would likely imply wasting computing resources. Now, what does processing *hits* actually mean ? As *hits* is a detection defined by a variety of features, one needs to choose which features to process to discriminate targets. The proposed filter will be based on a single feature of the *hit* object within the radar processing pipeline. This feature is defined in 1.2.

1.2 Pulse Doppler radars

1.2.1 Pulse Doppler radar targets backscatter

This work focuses on a specific kind of radar called pulse Doppler radar (PDR). Pulse Doppler radars send bursts of pulses of modulated electromagnetic waves toward targets to detect and characterize them. After one pulse is sent in the scanned direction, the received signals are sampled to retrieve the pulse reflections. The sampling frequency of the radar receiver defines a space discretization in the scanned direction, *i.e.* leads

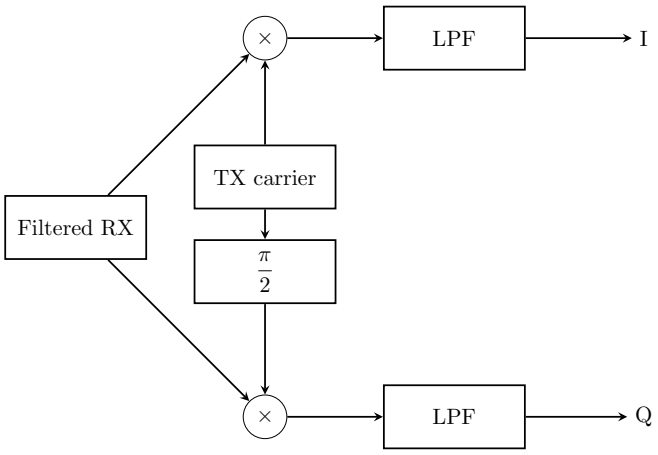


Figure 1.2: The proposed detections filter relies on I/Q samples stemming from an *I/Q detector* [114]. This I/Q detector takes the received (RX) signal as input, either at its carrier frequency or at a lower intermediate frequency. The received signal is passed through a bandpass filter beforehand to limit the noise [113]. The I/Q signal processed is complex-valued of the form $I + jQ$. This corresponds to a usual quadrature sampling setup, which can also be called complex sampling.

to the description of range bins contents with the pulse reflections. Each range bin is described by one complex coefficient per pulse, the coefficient being complex because of the I/Q reception of the radar [15, 113].

This vector of complex coefficients defines the *hit* object feature used by the proposed filter to discriminate targets, and is called *I/Q sweep response*. In this thesis, it will also be called I/Q signal since no ambiguity is possible. In order to provide as much information as possible to the targets discrimination, the existing feature is enriched to take into account a neighborhood of range cells centered around the cell of the detection instead of only the latter. It is important to understand that this is a neighborhood in ranges and not in azimuths. This transforms the 1D complex-valued vector input into a 2D complex-valued matrix input, as illustrated on Fig. 1.3. Using the notation of the latter where H is the size of the neighborhood of range cells and N the number of pulses in the burst, the input feature can be described by:

$$Z_{I/Q} = \begin{pmatrix} z_{11} & \dots & z_{1H} \\ \vdots & \ddots & \vdots \\ z_{N1} & \dots & z_{NH} \end{pmatrix} \tag{1.2}$$

In the previous equation, the number of rows in the matrix is the number of pulses in the burst creating the hit, while the number of columns is the number of neighboring range cells in the spatial context centered around the hit range cell. The discriminatory power of the feature chosen to filter detections lies in the Doppler information it contains. Any object reflecting the radar pulses with a non-zero radial velocity can induce a Doppler effect frequency modulation in the backscattered pulses. Let us identify this effect in the naive case of a sinusoidal pulse. At time t, the signal backscattered and received by the sensor receiver (RX) has a phase delay τ indicating the target range $x(t)$ and radial velocity $\dot{x}(t)$ (see Fig. 1.5):

$$s_{rx}(t) = sin\left(2\pi f_{tx}\left(t - \tau\right)\right) \tag{1.3}$$

$$s_{rx}(t) = sin\left(2\pi f_{tx}\left(t - \frac{2x(t)}{c}\right)\right) \tag{1.4}$$

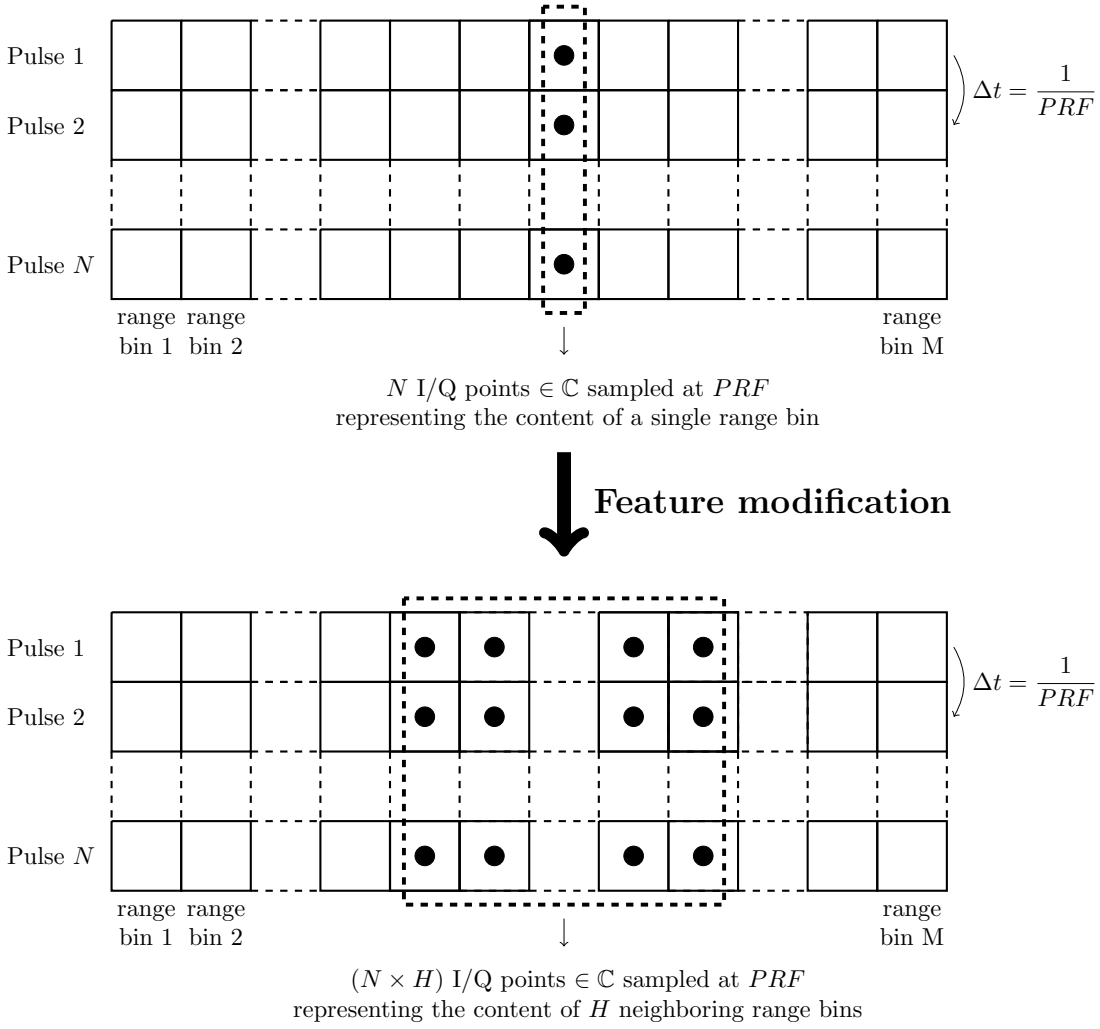


Figure 1.3: The proposed detections filter modifies the existing I/Q attribut of hits to integrate a neighborhood of range cells over the whole burst of pulses to provide the discrimination with additional information. In this figure the vertical axis describes the slow time of the sampling happening at PRF, *i.e.* between pulses, while the horizontal axis describes the fast time of the sampling happening at a much higher frequency and harbouring the range cells. The PRF is the sampling frequency that varies between hits, introducing a challenging physical diversity between input data points, the overall diversity also stemming from the varying number of pulses. The number of pulses here corresponds to N, the number of rows. Respecting the low-resolution constraint, we will consider $N \in [\![8, 32]\!]$, as these are typical values for *Pulse-Burst radars* (PBR) [167, p.188][133, p.470], which will be considered as equivalent to air surveillance PDRs in this thesis. The fast time, much higher, sampling frequency is considered to be constant across all hits and systems considered. An illustration of this diversity is proposed on Fig. 1.4. **Top:** original hit object I/Q feature (1D complex-valued vector) **Bottom:** new hit object I/Q feature (2D complex-valued matrix).

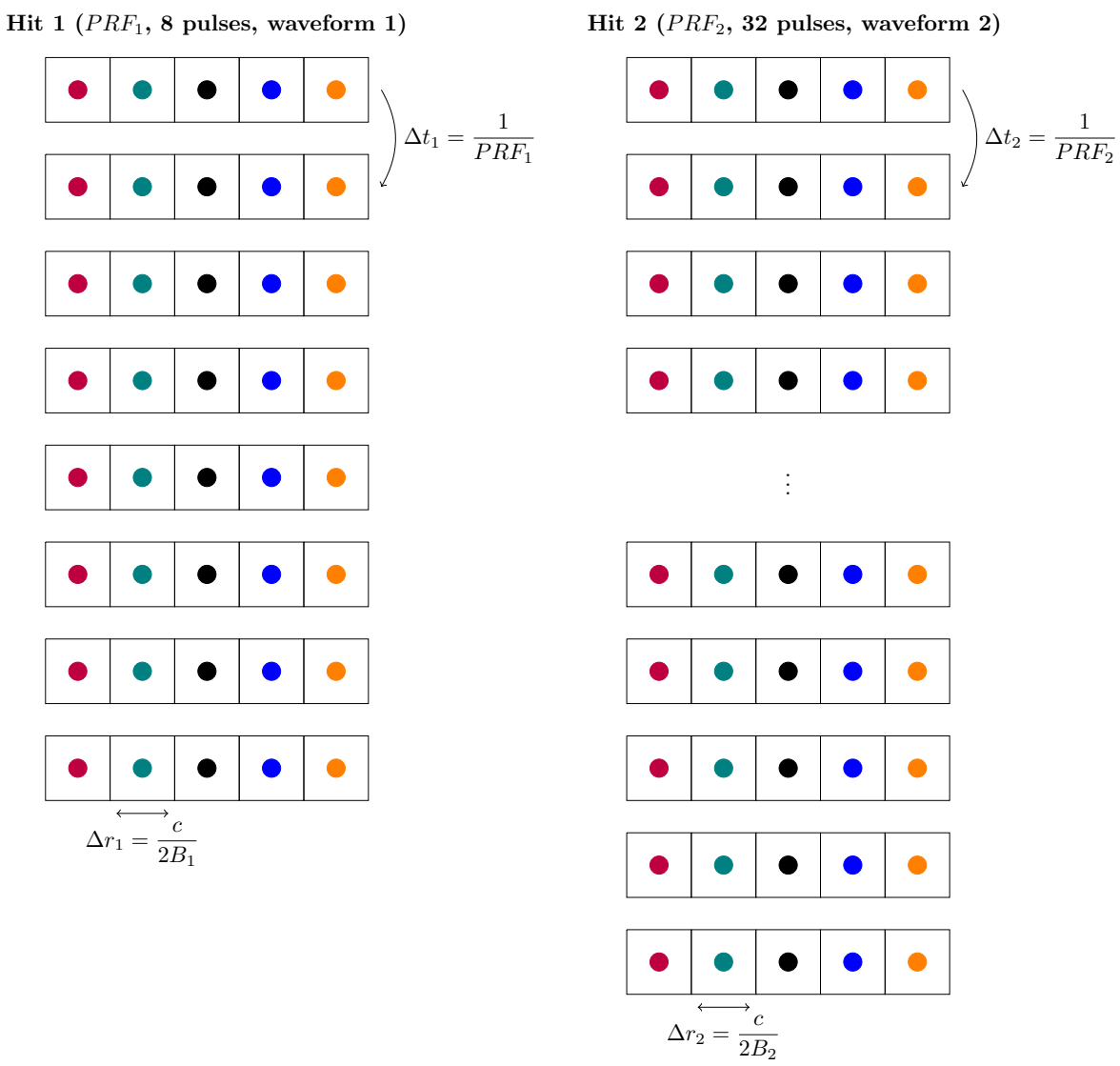


Figure 1.4: Illustration of the diversity of the input representation with two example inputs to our processing pipeline: the PRF and the number of pulses may vary between hits, forbidding direct comparison. This suggests a common encoding space is necessary. On this figure, we set the size of the range cells neighborhood, denoted as H on Fig. 1.3, to 5. The number of pulses per burst considered in this work, under the resolution constraint associated with air surveillance PDRs, belong to the integers interval $[\![8, 32]\!]$, in accordance with the typical values put forward in [167, p.188][133, p.470]. This will be the example case considered all along this document. The neighborhood includes the central range cell carrying the actual hit detection, *i.e.* the passing of a detection threshold. Each range cell is depicted with its own color to translate the possibility of each range cell carrying a specific target or Doppler information. In the case of air surveillance PDR the range cells spread widely in range because of the bandwidth as explained in 1.2.2, so they may very well contain unrelated information. Neighborhood correlation could also point to a weather phenomenon, the effective detection of the latter being of paramount importance to extract small and slow targets from the clutter. Bandwidth, *i.e.* pulse signal duration, is considered constant, meaning that on the illustration $B_1 = B_2$, thus $\Delta r_1 = \Delta r_2$. Said in other words, the spatial sampling does not change between hits. This illustration emphasizes how each input I/Q sweep response can be viewed as a multidimensional complex-valued time-series of varying length and sampling frequency. In a realistic setup, the waveform also differs between pulses.

Figure 1.5: The Doppler effect provides an effective way to discriminate between radar targets through the modulation of backscattered pulses due to the movement of targets with respect to the sensor. The phase of the backscattered signal received at time t carries the Doppler information.

where f is the frequency of the pulse transmitted, c the speed of light. If one considers a target with constant speed, *i.e.* $x(t) = x_0 + \dot{x}t$, the received signal becomes:

$$s_{rx}(t) = sin\left(2\pi f_{tx}\left(t - \frac{2(x_0 + \dot{x}t)}{c}\right)\right) \tag{1.5}$$

$$s_{rx}(t) = sin\left(2\pi f_{tx}t - 2\pi\left(\frac{2\dot{x}}{c}f_{tx}\right)t - 4\pi\frac{x_0}{c}f_{tx}\right) \tag{1.6}$$

Eq. (1.6) brings forth the constant phase term $\phi_0 = 4\pi\frac{x_0}{c}f_{tx}$ revealing the initial range of the target, and the time-dependent phase term $\phi_d = 2\pi\left(\frac{2\dot{x}}{c}f_{tx}\right)t$ which carries the Doppler information. The Doppler frequency of the moving target is therefore:

$$f_d = \frac{2\dot{x}}{c}f_{tx} = \frac{2\dot{x}}{\lambda_{tx}} \tag{1.7}$$

indicating the impact of the emitted wavelength over the Doppler shift. The Doppler shift is greater for smaller wavelengths, *i.e.* the radial velocity of a target is more noticeable with radars transmitting high frequencies.

The Doppler information can be extracted from the I/Q samples using either a discrete Fourier transform (DFT) or a filters bank. The filters bank is useful to extract Doppler content in frequency bins while adapting the sensibility of the filters to sidelobes, the latter being particularly problematic in the presence of clutter. The ability to characterize targets and distinguish between similar Doppler signatures is related to radar parameters such as the pulse repetition frequency (PRF) and the number of pulses available. For instance, if one retrieves the Doppler information of the I/Q coefficient for one range cell for a burst of N pulses with a DFT, one gets a spectrum of $N-1$ Doppler (*i.e.* frequency) bins describing $\left[-\frac{PRF}{2}; \frac{PRF}{2}\right]$ as a target signature. This emphasizes how critical the PRF and the number of pulses available are when it comes to targets discrimination. An illustration of the impact of the number of pulses over the Doppler spectrum is proposed on Fig. 1.6. As the I/Q samples sampling frequency for each range cell, the PRF effectively defines the frequencies limiting the unambiguous measure of a Doppler frequency shift, a high PRF allowing for greater unambiguous Doppler frequencies. This is due to the Shannon-Nyquist sampling theorem [66]. As many systems, a pulse Doppler radar is a sum of trade-offs, and adopting a high PRF lowers the unambiguous range of the sensor. For a target range to be unambiguously defined, a pulse reflection needs to be received before the next pulse is transmitted. The maximum unambiguous range r_{max} of a PDR is thus defined by the pulse repetition interval (PRI), where $PRI = \frac{1}{PRF}$, and is determined by the following equation [15, 113]:

$$r_{max} = \frac{PRI \times c}{2} \tag{1.8}$$

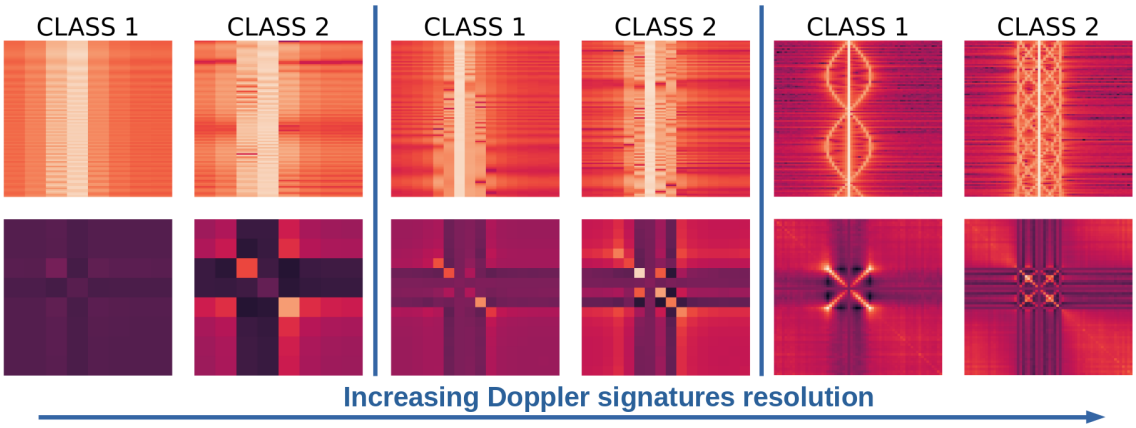


Figure 1.6: Illustration of the impact of the number of pulses over the Doppler spectrum of two helicopter-like targets belonging to classes differing only in the number of rotating blades creating the specificity of the modulation pattern, the remaining radar parameters remaining constant. **Top row:** Doppler signature. **Bottom row:** Covariance matrix computed over the rows of the Doppler signature. **Left:** 8 pulses. **Center:** 16 pulses. **Right:** 64 pulses. Provided with enough pulses, *i.e.* enough samples to compute a spectrum, one ends up with modulation patterns adapted to drone discrimination even without strong domain expertise, and easily identified using usual image processing methods and more general machine learning (see 1.2.2).

This additional middle ground between range and velocity ambiguity adds itself to the compromise on the transmitted wavelength. The trade-off on wavelength can be understood with Eq. (1.1) and Eq. (1.7), where the received backscattered power of Eq. (1.1) favors large wavelengths whereas the Doppler frequency shift analysis of Eq.(1.7) necessitates relatively shorter ones. The relevance of shorter wavelengths, *i.e.* higher frequencies, to make the Doppler features more visible is clearly identified where it's not only explicitly mentioned in [143], but also simply due to the micro-Doppler experiments put forward often relying on higher frequency bands than the L and S bands usually selected for medium and long range air surveillance [143, 131].

To overcome the range and Doppler ambiguities without completely sacrificing either the maximum unambiguous range or the maximum unambiguous Doppler frequency, one can regularly change the PRF of the operating radar. A varying number of pulses per burst can also be introduced to provide a diversity of Doppler resolutions for a given PRF, *i.e.* for a given range of unambiguous Doppler frequency shifts. This PRF and pulses quantity agility implies producing I/Q sweep responses of varying sizes and physical attributes. Handling such a variety of inputs is a key specificity of the proposed approach, this specificity being imposed by the real radar data to be processed. This variation of the input data points concretely translates into a changing number of rows in the 2D enriched input feature of Fig. 1.3, and into a possibly inconsistent sampling period separating rows from one sample to another.

The radars associated with the proposed targets discrimination are the Thales Ground Master (GM) radars, a line of air surveillance radars depicted on Fig. 1.7. In terms of resolution constraints, this line of radars has much lower Doppler resolutions than radars specialized in countering UAVs such as the Aveillant Gamekeeper [2]. The GM radars all have rotating antennas and operate on the S band between 2 and 4 GHz, *i.e.* with wavelength between 7.5 and 15 cm [11]. The GM radars range goes from 80 to 515 Km, with 1.5 to 6 seconds update period [9, 7, 8]. In comparison with these air surveillance radar parameters, the Aveillant Gamekeeper radar is in the L band between 1 and 2

Figure 1.7: The Ground Master air surveillance radars. Notice the rotating antenna, a key factor in the resolution constraints motivating this work. **Top left:** GM 200, medium-range radar **Top right:** GM 60, short-range radar **Bottom:** GM 400, long-range radar. ©*Thales*

GHz, has less than 10 Km of range and is equipped with a stationary antenna that allows for large numbers of pulses in bursts. High resolution range profiles (HRRP) describing generated by a different radar, a Thales Coast Watcher 100 (see Fig. 1.8), will also be used to conduct some of our one-class classification (OCC) experiments. This other radar, however, is not of interest regarding our hits encoding and was just taken advantage of in order to diversify the data employed to evaluate discrimination methods.

1.2.2 Doppler characterization of small and slow targets

Improving the detection and discrimination of small and slow targets in air surveillance radars is the goal of this work. Small and slow targets constitute a challenge in radar signal processing since they belong to the same target domain as most of the clutter, and have already motivated other dedicated works [17]. Targets having low RCS and low velocities can thus easily be confused with natural phenomena. Nonetheless, relevant small and slow targets could be distinguished from clutter using high-resolution Doppler features, or high-resolution range profiles, though obtaining HRRPs discriminative enough for small targets could be prohibitive since this would require a very large bandwidth. The relationship between the range resolution Δr and the reception

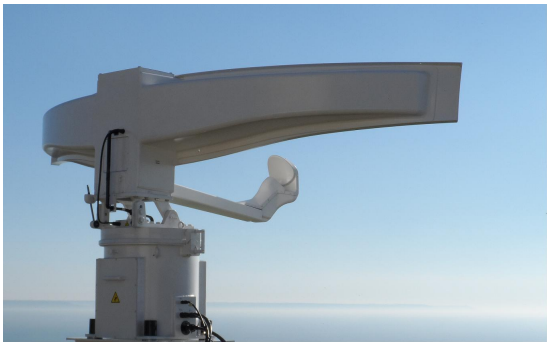


Figure 1.8: Thales Coast Watcher 100 radar, which generated the HRRPs used in this work to evaluate the possibility to discriminate radar targets with one-class classification approaches. See chapter 3 for the associated experiments. ©*Thales*

matched filter bandwidth B is described by:

$$\Delta r = \frac{c}{2B} \tag{1.9}$$

where c is the speed of light. The bandwidth is directly determined using the pulse width in time τ through the expression $B = \frac{1}{\tau}$, and is constrained by the antenna and the microwave hardware [15]. Intuitively, stretching out over too many frequencies can complicate processing since so many critical parameters are functions of λ. One can also mention that increasing the bandwidth also increases the noise which negatively impacts the signal-to-noise ratio (SNR) in the receiver, and thus the radar performances. The bandwidth accepted by the receiver may be chosen to be wider than the transmitted bandwidth in order to accommodate the bandwidth distortion due to the Doppler effect [6]. Many drones of interest for radar detection, such as the consumer market drones equipped with several rotors, would require centimeter-scale spatial resolution or even better to capture discriminative enough geometric features of the target [131]. As mentioned in the previous section, we do manipulate HRRPs for some of our OCC experiments, but the HRRPs in question depict large targets such as boats, thus not contradicting the the incompatibility of small targets discrimination and HRRP characterization, at least for the GM radars which have an inadequate bandwidth anyway contrary to the Coast Watcher 100. This work rather aims at using Doppler features, *i.e.* targets Doppler signatures, to discriminate between small and slow targets. Doppler features remain relevant even for slow targets since any vibration or movement on the target can induce a Doppler modulation [49]. Hence, target Doppler characterization is not limited to the bulk speed of a target, which can be flanked by Doppler spectrum modulations on both sides as it appears on Fig. 1.6. When such modulations are generated by moving parts on the target, one can talk about micro-Doppler effect [50].

Micro-Doppler modulations can reveal a bird wings movements, or the rotating blades of a drone, making it a suitable air surveillance radar targets discrimination basis [143, 33, 131]. This radar spectral feature is impressively versatile since it can be used in many other discrimination tasks, such as human activity identification [100], wind turbine detection [104], or ground vehicles classification [168]. The fact that it is not limited to the radar fingerprinting of aerial targets is an argument by itself emphasizing its descriptive power, and how relevant the choice of such features for our small and slow targets filter is. The micro-Doppler modulation patterns are such that they are adapted to image processing classification approaches. As such, state-of-the-art (SOTA) image processing methods such as Convolutional Neural Networks (CNN) are naturally applied to micro-Doppler data [72, 144, 98]. Other machine learning-based discrimination approaches were commonly used on micro-Doppler information before the advent

Figure 1.9: Radar targets domain conceptual split. The small and slow targets domain is especially hard to handle since it contains most of the clutter, *i.e.* the detections irrelevant to the operator's mission. This is the targets domain motivating this work.

of modern deep learning such as Support Vector Machines (SVM) [28, 131], k-nearest neighbor (k-NN) [168], naive Bayes classifiers (NBC) and regression trees [28]. Direct replacement of human classification on pre-existing target classification tasks were permitted by the application of machine learning to micro-Doppler data: [168] describes the case of micro-Doppler modulations brought down to baseband so that ground targets could be classified by soldiers listening to the signal.

Still, as previously suggested, not all radars are equal when it comes to Doppler spectrum resolution and range. As Eq. 1.7 pointed out, the carrier frequency of the pulses should be high enough not to crush the micro-Doppler modulation against the bulk speed Doppler frequency. This can be said to be a Doppler effect intensity limitation. Additionally, in a pulse Doppler radar, one needs numerous pulses to get numerous frequency bins to successfully isolate Doppler spectrum components with a sufficient velocity resolution. This can be said to be a Doppler cell resolution limitation. Since we are working with air surveillance radars equipped with rotating antennas, these two means of obtaining Doppler modulations noticeable enough to directly harness micro-Doppler effect signatures components are severely challenged. On the one hand the carrier frequency can not be too high on air surveillance radars in order to avoid excessive absorption by the atmosphere, where oxygen and water vapor resonance can even forbid certain frequencies [183, 184]. Atmospheric absorption is even more of a problem for ground-based air surveillance radars, like the GM radars with which we work, since absorption is stronger at lower altitudes because of greater molecular density [166]. On the other hand, the sensor can not afford to pause its rotation to send a large number of pulses in every direction, and then wait after each pulse to acquire the backscattered energy, as this would imply insufficiently frequent situation updates.

1.3 Overview of the proposed solution

1.3.1 Two-steps processing: encoding and discrimination

The previous sections of the chapter described where in the radar processing chain our targets filter should be implemented, what kind of targets the filter should focus on and under what resolution constraints the proposed filter is to work. We can now provide an overview of the filter put forward in this thesis with Fig. 1.10: the filter can be divided

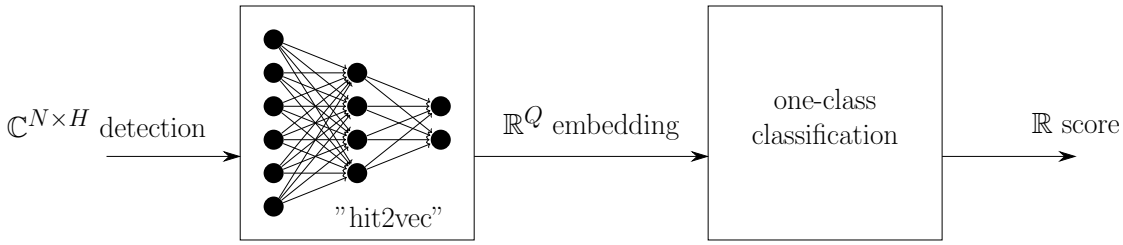


Figure 1.10: High-level description of the proposed targets discrimination. Hits are processed in two steps: they will first be encoded and projected to a shared representation space before discrimination using one-class classification. Note that the multi-dimensional complex-valued input of the "hit2vec" encoder has a varying size (due to a varying number of pulses per burst) and is defined by a variable sampling frequency (PRF). Here the output scalar score can either be a continuous real-valued abnormality score or a binary one-class classification score indicating whether the hit processed is in-distribution or not.

into two steps, encoding the enriched hit feature of varying size using deep learning and then filtering encoded hits with a one-class classification approach. The notation used on Fig. 1.10 and Fig. 1.11 corresponds to the one already used on Fig. 1.3, except for the introduction of an embedding size Q. Encoding data points before classification or other forms of discrimination is a common task in the literature. In a way, it amounts to a more or less thorough preprocessing of the data before classifying it.

Neural networks owing their success to their ability to discover and extract patterns [77, 110], the literature is unsurprisingly rich with examples of deep learning being used to extract intermediate representations capturing the main components of diversity within a dataset. For instance, an autoencoder (AE) trained to reconstruct input samples and remove noise at the same time can provide intermediate representations as encoding after an unsupervised training of the generative neural network [191]. Producing compact representations for a downstream discrimination with AEs does not necessarily require another task than the reconstruction itself [159], although when the sole reconstruction error is minimized during training a lower dimensionality bottleneck can help prevent an AE from learning the identity function [191]. One should note that a lower dimensionality bottleneck, implying an undercomplete AE is used, does not suffice to guarantee the identity function is not learned. An AE with a one-dimensional latent code could theoretically learn to map each training sample to its own unshared scalar representation before mapping it back to the input representation, assuming the encoder is powerful enough [77, p.494]. This reminds the machine learning practitioner the need to remain suspicious of what was actually learned by the neural network. One of the key ideas put forward by this thesis is a neural network-based encoding of radar hits I/Q features. The final processing step is one-class classification due to the lack of labeled data points, a choice thus stemming from a low supervision constraint. This final step could however more generally be described as an encoding discrimination process, if one ignores the lack of supervision, as suggested on Fig. 1.11.

Decomposing the filter into these two stages decouples the hits representation problem from the hits separation problem. It enables the independent development of each of the two stages, and makes each stage useful even if taken alone. For instance, whatever the level of hits labels availability, *i.e.* whether one is going for out-of-distribution detection (OODD) or fully supervised multi-class classification, an effective *hit2vec*[1] en-

[1]Using a terminology similar to the one already found in the deep learning literature, we call *hit2vec* a neural network architecture that translates radar hits features into a real-valued vector, *i.e.* a common

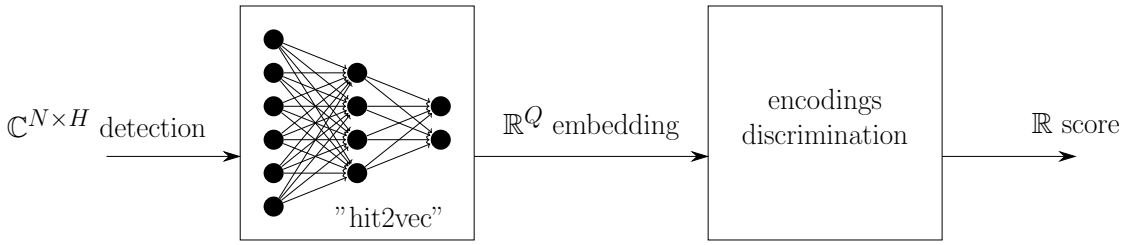


Figure 1.11: High-level description of the proposed targets without our problem-specific supervision constraint: once they have been projected to a shared representation space, encoded hits can be separated with any discrimination adapted to the availability of labels. One can think of clustering, one-class classification, or in the ideal case supervised classification. This makes the proposed pipeline relevant to most targets separation tasks with PDR, whatever the supervision level. In the case of clustering, the output scalar is a cluster label. For one-class classification, the output scalar can either be a continuous real-valued abnormality score or a binary one-class classification score indicating whether the hit processed is in-distribution or not. If enough labels are available and supervised classification is used to discriminate between encoded hits (also called hits embeddings), the output scalar would be a class label, which may include a rejection class to filter out out-of-distribution embeddings.

coding neural network architecture would be equally relevant. This also emphasizes how this work is relevant beyond the very specific scope of air surveillance radars and small and slow targets. If a successful model is found to embed enriched hits I/Q sweep responses into an ordinary vector representation, such a model could be used for targets discrimination task in any pulse Doppler radar, for any level of training supervision.

1.3.2 One-class classification for radar targets discrimination

The low supervision constraint is tackled using a data points separation approach adapted to the low availability of labels, *i.e.* one-class classification. This OCC can be understood as an anomaly detection (AD) problem, which relates to the evaluation of the distance between a test point and a given data distribution. In this manuscript we allow ourselves to use interchangeably OCC, AD and OODD, although the literature sometimes introduces distinctions between these three tasks [156]. In consequence, we will also use the terms in-distribution samples and one-class samples to refer to the same samples. Anomaly detection can as well be called outlier detection, and be associated with novelty detection. As [47] pointed out, various kinds of anomalies can be distinguished: points anomalies, contextual anomalies, and collective anomalies. Whereas point anomalies can be detected thanks to their features alone, contextual anomalies admit normal features values but are unusual given their data context. Collective anomalies are multiple related data points deemed abnormal because of their collective realization. One can note that given the previous definitions, one can integrate additional contextual information to switch from a point or collective anomaly detection problem to a contextual anomaly detection.

All of these types of anomalies can be found in radar targets processing, where one test data point amounts to one target. For instance, a drone-type target having a very high speed, or an odd Doppler signature or an odd HRRP, could constitute a point anomaly. On the other hand, a normal-looking drone with a unique trajectory

representation in machine learning pipelines. This is analogous to *word2vec* which represents words with vectors [128], or *wav2vec* which encodes tens of milliseconds of audio in a single vector [163]. The architecture behind *wav2vec* will be discussed in chapter 2.

when compared with the other drones of the targets dataset would define a contextual anomaly. A collective anomaly could be created by a set of drones that alone have normal individual features and trajectories, but whose trajectories combined are unusual. This not only shows how relevant anomaly detection is to radar targets processing, but also emphasizes deep and non-deep AD can contribute to different stages of the radar processing pipeline as it was depicted on Fig. 1.1. For example, in addition to the implementation of our hits filter located before the *Extraction* step, patterns of suspicious trajectories over a short period of time could be analyzed after the *Tracking* step. The literature offers numerous examples of AD being used to separate radar targets. For instance, a Gaussian Mixture Model (GMM) is used as a cluster model in [107] to achieve AD on velocities and positions routinely generated by a surface radar watching over ships. In [193], an AE intermediate representation is used as a code to enable the computation of an *averaged background code* of radar range profiles. A substantial difference with this reference code can then be used to reveal an out-of-distribution test sample.

In the specific case of our hits filter, we consider one enriched hit, *i.e.* a neighborhood of range cells described by their respective I/Q sweep, as an input data point to be evaluated. This implies we are working on a point anomaly detection task. Our filter could be seen as a contextual anomaly detection task if the data point processed was a single I/Q sweep feature describing one single range cell, since we are computing an AD score based on the I/Q sweep of the center range cell responsible for the hit detection, and its relationship with the neighboring I/Q sweeps. In the end the unit of data that needs to be described as in or out-of-distribution (OOD) with respect to reference targets is the target behind the central range cell in the enriched I/Q sweep representation, that is we are considering a single target with a local I/Q context, and not a neighborhood of hits or targets as it would be the case for a collective anomaly detection task. Considering a neighborhood of hits would not fit in with the existing processing chain of the GM radars. Indeed, the latter currently processes hits individually, and creating new data flows combining hits on a local basis to process them would be computationally too demanding. This will have an impact on the proposed hits processing, and will be discussed more deeply in chapter 2.

The one-class classification approaches compared allow for both unsupervised and semi-supervised anomaly detection (SAD), based on the definitions provided by [152, 153]. We will identify the experiments where the training data is considered to be "normal", *i.e.* the data points belong to the "one-class", as unsupervised AD experiments. On the other hand experiments where in addition to a majority of normal samples, a non-representative minority of "anomalies", *i.e.* out-of-distribution samples, is available during training to refine the one-class classification boundary will be identified as SAD. As reminded in [152], this is not necessarily aligned with the literature, where rightly so, what we call unsupervised AD is already semi-supervised learning since the possibility to count training samples as normal implies some level of supervision. Such a definition is put forward in [47], where unsupervised AD refers to AD without training data and under the assumption that the test data is made of a large majority of in-distribution samples. This can be seen as the common outlier detection conducted as preprocessing when one works with a dataset outside of a machine learning context. The data types used for the OCC experiments of this thesis are diverse: encoded hits, standard images datasets from the deep learning literature, 2D spectrums, covariance matrices but also 1D range profiles. Self-supervised learning (SSL) will also contribute to some of our experiments through transformations converting in-distribution data into negative samples to enable SAD based on artificial anomalies. The Toeplitz and covariance matrices representations are put forward as relevant representations for signals discrimination, as

recently reminded in [43, 32, 127].

1.3.3 The choice of deep learning for I/Q encoding

As indicated in 1.3.1, deep learning was selected to build the encoding step of our hits discrimination. We mentioned a basic use case stemming from the literature where AEs are used to generate compact representations of data points that will be fed to a discrimination function in 1.3.1. We can first add that such an embeddings generating mechanism through a generative architecture has been reproduced several times for various kinds of inputs, for instance in [88] with the evolved version of recurrent neural networks (RNN) called long short-term memory (LSTM) architectures [87], further supporting the encoding power of deep learning. One can also notice that the handcrafted features, such as the renowned SIFT [122] in image processing, have disappeared in favor of both feature extraction and decision taking being handled by backpropagation-guided neural networks, the switch being motivated by much improved performances. The key remaining choice for the deep architecture designer being perhaps the information flow constraints set by the architecture itself. Works actually indicate that a very substantial part of a deep architecture relevance can stem from its architecture alone, which can already lead to pleasing performances even when left untrained [69, 74, 160]. This point could nevertheless be balanced by the fact that neural networks are usually sensible to initialization, this sensibility bringing about the need to systematically evaluate neural networks over a series of distinct random initializations.

The choice to favor deep learning is not only due to the pattern identification power of deep neural networks, but also to their ability to integrate a diversity of input information through the filters and operations already available in the deep architectures of the literature. Deep learning is not limited to the now almost aged dense, simple recurrent and convolutional architectures, it is also the continuously renewed tale of deep network adapted to non-Euclidean data lying on graphs and Riemannian manifolds [31], or innovative sequence-to-sequence architectures such as Transformers [185]. As we will see both in chapter 1 and chapter 2, this neural networks architectural wealth will be leveraged in our work, where geometric deep learning (GDL) and sequence-to-sequence (seq2seq) models will be employed. In the case of geometric deep learning, graph neural networks (GNN) will be put forward in chapter 2 to handle the neighborhood of range cells describing a hit and integrate the varying radar parameters from one it to another in the graph edges. Symmetric positive definite (SPD) manifold neural networks are proposed in chapter 3 to conduct OCC on SPD representations, several of which will be mentioned in this work in the context of radar targets discrimination.

Since we just mentioned how quickly deep learning evolves and redefines its own SOTA, one could wonder what the point is of exploring the aged models of tomorrow for our specific application. Deep learning keeps relying on backpropagation and gradient descent mechanisms to learn to extract features and produce classes or regression results, therefore we argue that understanding how such a learning mechanism can complement the usual radar signal processing will be useful even if the architecture experimented on does not belong to SOTA. Furthermore, the need to physically understand and interpret the operations, recalling the rising demand for eXplainable artificial intelligence (XAI), constrains the relative arbitrary complexity of neural architectures anyway. This need for interpretable sets of learned transformations is rooted, as in other critical applications of artificial intelligence, in the required guarantees widespread in the military sensors industry. In the end, deep learning filter or not, the targets discrimination sold is required to stay within the limits of the advertised requirements existing for any radar system.

On the contrary, deep learning was never meant to be a definite choice regarding

the second step of our filter, *i.e.* the target embeddings discrimination. This is why chapter 3 considers a variety of non-deep machine learning OCC among the conducted experiments. Our pipeline remains completely reliant on machine learning, which, simply put, aims at letting a neural network extract what is believed to be suitable Doppler information and use it to produce a useful score for the radar mission. The search for automatic or semi-automatic Doppler, and more generally, spectral information extraction and encoding dates back to decades ago, when expert systems were already considered to help improve spectral information analysis [14].

1.3.4 A brief reminder on deep learning

To make this document friendlier to readers with *only* a radar background, a short non-exhaustive reminder on deep learning, the machine learning method put forward in this work, is now proposed. If confusion remains on deep learning notions readers may refer to [77] for supplementary explanations, this reference being the inspiration of most of the elements presented here. Deep learning gathers a wide and diverse ensemble of machine learning methods based on neural networks architectures. Neural networks can be described as a succession of parameterized linear and non-linear transformations applied to numerical input data whose parameters are trainable. These trainable parameters are organized in successive layers, each layer producing intermediate representations supposedly yielding higher level features than the previous ones. The non-linear transformations are defined using non-linear functions called activation functions. Specific architectures include parallel and independent successions of layers, residual connections bypassing layers to provide the training phase with the opportunity to recombine information from distinct depths, convolutions to extract multi-scale spatial or temporal patterns and recurrent computing mechanisms for time-series processing. To determine the values of the parameters a computationally expensive, data-dependent, training phase is executed during which a scalar objective quantity often called a cost or a loss function is minimized thanks to the optimization of the already mentioned parameters. Assuming a joint distribution D of the samples x and of their associated targets y, the training phase yields the following minimization problem:

$$\min_{W} \mathbb{E}_{(x,y)\sim D} \left[\mathcal{L}(\Phi_W(x), y) \right] \tag{1.10}$$

where W are the optimized trainable parameters organized within the neural network defined by the function Φ, and $\mathcal{L}$ is the loss function defining the learning task. The minimization is said to be empirical since the training is actually based on a finite number of inputs and outputs pairs drawn from D, *i.e.* only a sampled version of D is available to optimize W. Various loss functions will be put forward in this work. The latter will be regression losses, which can be opposed to the equally widespread classification losses. An intuitive way to distinguish these two types of losses is to see regression losses as the prediction of a continuous output representation, for instance based on the minimization of a metric, while the classification losses aim at predicting a discrete class label. One can note that the prediction of a discrete label can still translate into the prediction of a continuous quantity such as a float in $[0; 1]$ interpreted as a class probability. A typical approach to classification involves the minimization of a cross-entropy.

This optimization is usually implemented using a variation of the gradient descent called stochastic gradient descent (SGD). The SGD gradient descent is deemed stochastic because it estimates the gradient over a large dataset using a smaller set of randomly sampled data points called a minibatch. For instance, if one considers a minibatch of n vectorized samples $x_i \in \mathbb{R}^d$ and their associated targets y_i picked in the training set for

a given task, the update to the parameters W can happen as follows:

$$W \leftarrow W - \epsilon \widehat{g} \tag{1.11}$$

where ϵ is the learning rate and $\widehat{g}$ is the gradient estimate over the set of n samples of the loss with respect to the parameters W:

$$\widehat{g} = \frac{1}{n} \nabla_W \sum_{i}^{n} \mathcal{L}(\Phi_W(x_i), y_i) \tag{1.12}$$

The obtained sequence of transformations outputs a representation that translates into a discriminative or generative result and can handle unsupervised and supervised tasks, the supervision level being the result of the amount and relevance of the labels available in the training data. Based on the scalar objective computed after each complete forward pass going through all layers in a unique order, one computes the scalar objective gradient with respect to the neural network parameters and uses this gradient to update the parameters so that the previous objective is minimized. Since the trainable parameters are distributed in several transformations layers, the chain rule of calculus is used to backpropagate the learning signal under the form of derivatives and produced by the scalar objective gradient back through the layers. For instance, for a scalar x and two functions f and g, if one defines $z = f(g(x)) = f(y)$, the chain rule corresponds to:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = \frac{df(y)}{dy}\frac{dg(x)}{dx} \tag{1.13}$$

The previous equation clearly illustrates the intuition behind the chain rule: it allows for the appearance of the intermediate function g and representation y in the derivative of the functions composition with respect to the initial input x. So-called intermediate derivatives can thus be combined to compute the gradient with respect to the final loss at any stage within the neural network.

This rule is used because layers are compositions of the ones preceding them and is easily transposed from scalars to vectors and tensors. In multidimensional cases the Jacobian matrix of first-order partial derivatives thus naturally appears in the learning phase. This process constitutes one of the pillars of deep learning and is called backpropagation. This simple yet intuitive description of backpropagation already suffices to understand that one of the difficult points in deep neural network training is the ability to maintain a usable training signal until the first layers when backpropagating the training loss gradient. This difficulty motivates the choice of operations within neural networks, and notably led to the exploration of a diversity of non-linear activation functions.

Assuming relevant data quantity and quality, each deep learning application necessarily leads to a set of choices which make the difficulty of creating a successful deep learning setup. Along with the neural network architecture itself, finding the right combination of hyperparameters, *i.e.* all the parameters of the architecture outside of the parameters optimized during the training phase, can be particularly challenging. The recent developments in every dimension underlying the task of defining a deep learning architecture, whether it is optimization, constraints on weights, or data augmentation methods, contribute to the difficulty of making the right design choice. Constraints on the weights and the output representations can relate to a major hypothesis of deep learning called the manifold hypothesis. This hypothesis implies that relevant intermediate and output representations only span over a limited part of the representation space. Assuming this hypothesis is valid suggests finding a way to learn parameters and representations limited to the relevant subset of points is crucial to effective neural network training, in addition to easing the understanding of the trained parameters

and representations. The black-box nature of complex architectures complicate the acceptance of discrimination and representations generated by neural networks, especially when it comes to critical systems such as radars. This legitimate defiance calls for a careful evaluation of the trained neural networks performances using thoughtfully selected metrics, and for an attentive analysis of the learned parameters after training. Several methods such as Grad-CAM [165] relying on gradient and activation values are already popular, and sometimes learned parameters can be compared to finite impulse response (FIR) filters or Fourier atoms [38, 37]. Directly visualizing the learned filters can be an option, typically in image processing [191].

The necessity of thoroughly explaining and evaluating neural networks was already emphasized in the literature, the latter putting forward the necessity of constraining weights in a regularization effort and of tackling the sensibility of neural networks to slightly noisy inputs that can be called adversarial examples, or their sensibility to parameters initialization. The fact that a lucky random initialization of weights changes performances, or that random weights can already yield interesting high level features highlights how important it is to understand what was gained during the training phase. Knowing the right amount of parameters optimization that should be extracted from a given training set is another important part of successful neural network training. Indeed, one can train a neural network too much with a given training set, leading to a situation called overfitting, where the neural network is becoming so specialized with respect to the training data that it loses generalization capabilities on unseen data for the task at hand. Among other key elements neural networks and backpropagation, the foundations of deep learning, have existed for decades but became widely popular only a little more than a decade ago thanks to the combination of the availability of large datasets for unsupervised and supervised training and of new specialized computing hardware and software [110]. Nowadays large, suitable datasets are even more widespread and distributed computing capabilities are harnessed to train enormous neural networks made of hundreds of millions or even billions of parameters [53, 39, 61]. This thesis distances itself from such very large models since it develops small neural networks with few parameters and little data that are meant to be easily investigated, explained and implemented on computationally constrained platforms.

1.3.5 A brief reminder on signal processing

As for deep learning above, a few fundamental notions of signal processing are reminded to provide the reader with some perspective. This thesis implements complex-valued signals processing pipelines in the context of radar systems. The proposed solution combines machine learning and traditional signal processing concepts to improve existing pipelines which are void of machine learning. The complex-valued signals are sampled signals which are assumed to respect the condition brought by the Shannon-Nyquist sampling theorem. This implies the frequency content of the signals processed does not contain frequencies above half of their sampling frequency, the latter being here defined by the PRF. This condition relates to the concept of aliasing, which translates into the impossibility of reconstructing a signal based on the sampled points due to the violation of the Shannon-Nyquist sampling theorem. Said in other words, we need enough samples per time span to accurately and uniquely capture the frequency content of a sampled signal. Here, this requirement is critical not to reconstruct the signal but to ensure the discriminative information our proposed processing needs is not limited by the sampling. This criteria is assumed to be systematically met in our work, and basically means the targets Doppler spectrums do not go beyond the Doppler ambiguity boundaries from a radar engineer point of view. This is made easy by our focus on slow targets. More generally, an anti-aliasing filter can be used to avoid aliasing. This anti-

aliasing can translate into a low-pass filter forbidding any information to lie outside of the bandwidth covered by the sampling frequency. Such a low-pass filter would be equivalent in our PDR case to the rejection of targets with velocities beyond the ambiguous velocity allowed by the sampling PRF. The usual bandlimited signal hypothesis applied to PDR results in the unambiguous velocities range in which we will place ourselves.

Since we face a discrimination between signals with varying sampling frequencies, usual signal resampling approaches should be kept in mind. A common way of resampling a signal is to combine upsampling and downsampling by integer factors, resulting in a final rate conversion corresponding to a rational factor. More precisely, the upsampling can be associated with an integer interpolation factor I and the downsampling can be associated with an integer decimation factor D, yielding the rational factor $\frac{I}{D}$. Upsampling by an integer factor can be achieved by adding zeros between existing samples and then applying a low-pass filter, which acts as interpolation. Downsampling on the other hand first applies a low-pass filter and then keeps a fraction of the existing samples. This rate conversion is not necessarily relevant in our case because of the possibly close and too diverse PRFs at play in a set of input signals to be compared. Both the proximity and the diversity of the sampling frequencies in our radar pulses bursts make it likely to require conversion rates where either the factor I or the factor D, or both, are large. This, in addition to the very few signal samples available, would likely lead to inefficient conversions. For instance, the larger the upsampling factor I, the larger the computational and memory loads the processing needs to face due to the additional samples generated. The specificity of our PDR application case encourages us to evaluate the possibility of bringing the diversity of input signals to a shared representation space with less common and more complex approaches and motivates our deep learning-based experiments. One should note that more subtle signal interpolation could allow for the conversion rates necessary in our particular PDR case study, but this is outside of the scope of our work. Details regarding the resampling notions presented in this paragraph as well as complementary information can be found in [66].

Assuming an effective rate conversion with a proper rational factor $\frac{I}{D}$ could be found to bring back all bursts of pulses to a common PRF and a common number of I/Q samples, one could imagine a resampling-based "hit2vec" encoder. Such an encoder could output the DFT generated features for each resampled single range cell I/Q signal as representation in a shared representation space. The combination of the DFT output for each range cell in a given neighborhood of H range cells (see Fig. 1.3), for instance through the computation of a mean, would then constitute a fixed size representation of the enriched input format matrix. This possible machine learning-free encoder baseline is depicted on Fig. 1.12. The latter respects the hypothesis according to which the discriminative information among the small and slow targets we focus on and the confusing clutter lies in the Doppler *i.e.* frequency domain, since it relies on the DFT to extract the representations fed to the ensuing separation step.

A notion central to both general signal processing and radar signal processing, the signal-to-noise ratio, will not be discussed in this work, the latter focusing on the proposal of a coherent machine-learning based processing adapted to the discrimination task and the specific input matrix. A favorable SNR nonetheless remains a sine qua non condition for the success of the proposed pipeline, despite the lowering of detection thresholds. The requirement of acceptable SNR is actually closely tied to the few radar notions highlighted by our processing. For instance, hits stem from a detection step (see Fig. 1.1) that may rely on a coherent or non-coherent pulses integration to accumulate the energy of a train of pulses in order to decide whether to detect a target or not. Here, the pulses integration aims at improving the SNR to ease the decision to detect or not. This detection translates into the passing of a threshold, while the SNR increase

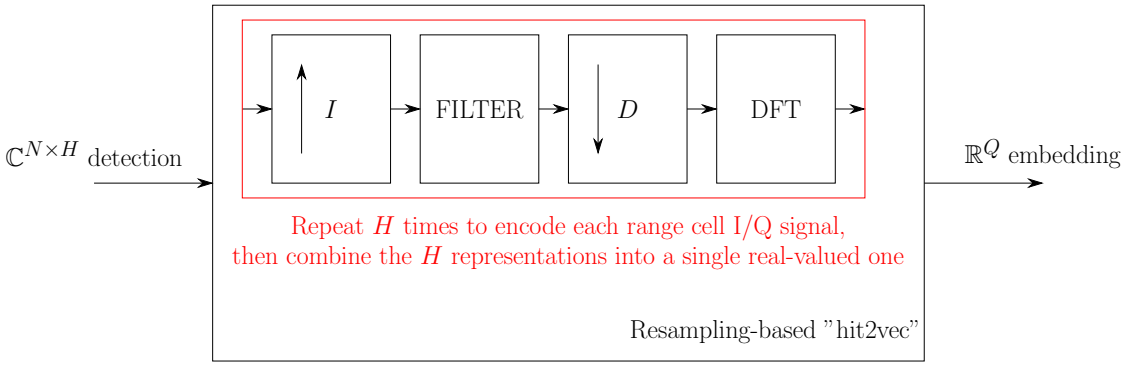


Figure 1.12: Example of "hit2vec" resampling-based baseline for the first of the two steps processing proposed and already depicted on Fig. 1.10 and Fig. 1.11. Assuming the succession of an upsampling by the integer factor I, an intermediate filter and a downsampling by the integer factor D can allow to represent all input signals with a common sampling frequency and an equal number of samples, the neighborhood of H range cells (see Fig. 1.3) could be encoded by the mean of the DFT of each individual range cell signal.

can be seen as making the signal energy spike more visible with respect to the ambient clutter. The detection threshold is in turn decided by a constant-false-alarm-rate (CFAR) directly tied to the false alarm filtering this thesis puts forward. In the PDR processing pipeline, the described pulses integration should not be confused with pulse compression, the latter aiming at improving the range resolution [15].

The history of signal processing suggests the combination of signal processing and machine learning is a natural development of the field. Signal processing was originally very close to the field of physics and less so to the field of mathematics due to its analog nature. With the advent of microprocessors and modern computing chips, signal processing took the digital turn, digital signal processing became widespread and moved the field towards data processing and applied mathematics. This shift can be illustrated by the impact of the Fast Fourier Transform (FFT) algorithm [68]. The development of machine and deep learning to process all kinds of data was thus set to equally impact signal processing. Key deep learning tools can be defined with equivalent concepts stemming for signal processing. As was already mentioned in 1.3.4, a convolutional layer is a FIR filter, and the DFT can be learned by the coefficients within a neural network [38, 37, 189]. On a side note, one could add that the FFT has also been used to make the computation of convolutions more efficient thanks to the equivalence between convolutions in the spatial or temporal domain and the Hadamard pointwise product in the Fourier domain [142].

Deep neural networks have been adapted to process the representations usually handled by signal processing such as complex-valued vectors stemming from a DFT or I/Q signal. This adaptation did not wait for the recent explosion of deep learning research and was engaged decades ago [86, 99, 91, 71, 112]. Audio signal [19, 163, 55], EEG [162] and ECG [88] processing are among the most common and successful applications of deep learning to real and complex-valued valued signal. Denoising, one of the most fundamental tasks of image and signal processing, has been a successful application of deep learning [57, 111, 192]. Closer to our radar application, beamforming [173] and radio modulation discrimination [119, 106, 138] define other applications of deep learning to signal processing. Among the applications of deep learning to signals discrimination, one can notably distinguish the processing of signal based on 2D spectrums, such as spectrograms, from the processing based on raw temporal or spatial signal. This sepa-

ration is relevant since methods used on 2D spectrums can be seen as closer to image processing than to signal processing, even though such a separation is debatable due to the proximity between signal and image processing operators.

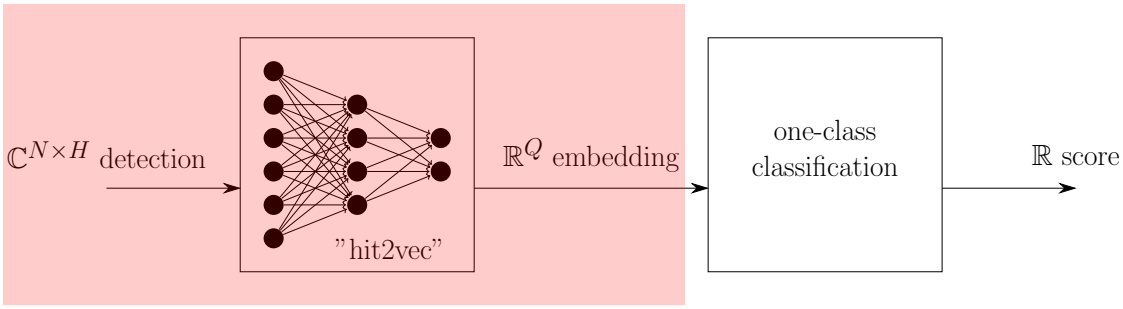
1.4 Organization of the chapters and contributions

This chapter has now introduced each element of the thesis title: OCC and low supervision refer to the need to tackle a lack of labels, the sensors generating the data are PDRs with a low Doppler resolution constraint, and the task at hand is targets discrimination using I/Q samples carrying supposedly discriminative Doppler information. The assumption regarding the relevance of the Doppler information for our discrimination task is supported by the considerable literature using exclusively micro-Doppler information to detect and classify small and slow targets such as drones and birds. The second chapter of this thesis will describe the encoding stage of the proposed hits processing pipeline, while the third chapter will outline the final OCC discrimination stage. The encoding stage will put forward the use of RNNs, CNNs, complex-valued neural networks (CVNN) and also of GNNs to produce a relevant real-valued vector representation of the hit feature described by Eq. (1.2). The acronym RVNN for real-valued neural network will be used in opposition to CVNN. The third chapter will on the other hand present OCC baselines considered in our comparisons, but also original work and OCC methods developments. A fourth chapter will then analyze experiments conducted with the complete two-steps pipeline, and is followed by a concluding chapter that ends with perspectives. Deep learning definitions regarding specific deep architectures put forward in our experiments and complementary to the reminder proposed in 1.3.4 will be proposed in the second chapter.

The contributions of this thesis are disseminated in the chapters 2, 3 and 4. As previously mentioned, an original contribution to OCC is put forward in chapter 3 in section 3.1.3. This contribution consists in the modification of an existing Deep OCC method along with the proposal of a latent space regularization scheme in section 3.1.5. Another modification of the same existing Deep OCC is proposed in section 3.2.3 without being supported by experimental results. Although the building blocks of the encoding architectures presented in chapter 2 stem from the literature, their use, combination and adaptation to the specific nature of PDR data and to the problem of small and slow targets discrimination constitute another contribution to the literature. In addition to the previous contributions, we can also note that the choice of the enriched input features format with subsequent representation learning and machine learning-based discrimination constitute a novelty in the field of radar signal and data processing. At the time of writing, the research papers associated with this thesis, either submitted or published, all stem from the content of chapter 3. The chapters 2 and 4 will remain mostly exploratory with the presentation and comparison of encoding ideas supported by a few preliminary results.

Chapter 2

Encoding IQ signals



Chapter 2

Let us now consider the first stage of our hits filter, *i.e.* the hits encoding stage. Learning robust latent embeddings which concentrate latent representations of similar targets next to each other is the purpose of this stage. As explained in chapter 1 and depicted on Fig. 1.3 and Fig. 1.4, targets are represented by a neighborhood of range cells centered on the range cell actually carrying the potential target that has triggered the detection of a hit due to the passing of a threshold. This neighborhood spans over range cells but not over the neighboring azimuths (see Fig. 2.1). Ideally, distinct classes of targets will be projected in distinct latent clusters by the encoding neural network. The output representation should be real-valued vectors of fixed size in order to enable the downstream discrimination to be achieved by common machine learning approaches, without further adaptation or the need for specific interpretation of the latent components. The adjective robust here emphasizes the necessity for comparable radar targets to be projected similarly, even with small perturbations in the backscattered radar signal. As we will see, this encoding neural network will associate different types of deep learning architectures to handle the particularity of the input data.

2.1 The common representation space necessity

As we have seen in Chapter 1, the input I/Q signals we wish to transform are of varying size and varying physical nature, since the number of pulses and the PRF change during the radar operation (see Fig. 1.4). This input heterogeneity renders it irrelevant to directly compare the coefficients of several hits. In order for common data points classification and discrimination methods to be applicable, the coefficients constituting each hit representation should have the same mathematical nature and belong to a representation space of unique dimensionality. Obtaining such a shared representation is the aim of the first stage of the filter. In a way, we are here looking for representations

Input variation cause	Ideal embedding variation
PRF	Invariant
# of pulses in burst	Invariant
Target-aspect sensitivity	Invariant
Transmitted frequency	Invariant
Target type	Varying

Table 2.1: The ideal PDR targets I/Q backscatter embeddings invariances. All of the embeddings invariances need to be learned, implying the availability of dedicated data samples to enable the learning of the invariances during training. There is no way to integrate operations within the encoding hit2vec neural network to make the neural network intrinsically invariant to the cause of input representations variations. In such a case, learning invariances is similar to the use of geometric transformations for data augmentation to make the result of classification neural networks invariant to the same geometric transformations. In our experiments only the target type and the number of pulses per burst change. Such a setup already appeared as challenging.

of targets insensible to so-called signal sampling perturbations: one target should have the same output representation for different number of pulses, *i.e.* different numbers of samples, and different PRFs. We can talk of target-aspect invariance in the output space of the encoding architecture, in reference to the usual target-aspect sensitivity of radar targets [116]. Another target-aspect invariance ideally expected from the proposed encoding that we will not further mention is the invariance to waveform changes, since in addition to varying PRF and number of pulses, hits of the GM air surveillance radars are defined by flexible modulation frequencies and other waveforms parameters. The number of pulses, the PRF, and the waveform are all parameters of the sensor generating the target raw representation. Additional invariance to external factors should ideally be ensured for a discrimination relevant to a radar operator to be produced. Such external factors are for example the weather or the target range. For instance, two drones sharing a common geometrical and propellers configuration should be encoded similarly, independently from the range at which they have been detected. Interesting encoding invariances are summarized in table 2.1.

One can note that we chose to process the backscatter describing a neighborhood of range cells centered around a single hit, instead of processing a spatio-temporal neighborhood of hits, *i.e.* several hits at once. This is imposed by the radar processing pipeline in which our filter should be integrated, where hits are processed alone before the extraction stage. Adding a search mechanism among local combinations of hits would be computationally prohibitive for the embedded hardware of the radar system. This computational barrier is even truer in the context of the lowered detection thresholds motivating this thesis, since lowering filters increases the number of hits requiring real-time processing as explained in section 1.1. The increase in the number of hits would in turn increase the number of possible hits combinations to be considered if the hits filter had been based on spatio-temporal neighborhoods of hits instead of fixed-size range cells neighborhoods. Opting for spatio-temporal neighborhoods also implies answering challenging questions: how far can a hit be in time and space in order to be accepted within a local processing ? How many hits would the local processing require or would accept at most ? To be fair, the increase of hits due to lower thresholds also impact the number of fixed-size neighborhoods of range cells to process, but this computational load is linear since no combinations are considered. This lighter impact on hardware is much more embedded systems-friendly, while the fixed-size of the range spanned by the input

complex-valued matrices makes the computational load more predictable. In the spirit of fair comparison, one can furthermore notice that the fixed-size of the neighborhood of range cells is a choice similar to the spatio-temporal delimitation of hits combinations.

Another way to understand the input we have is to see the enriched I/Q feature as a contextualized I/Q feature, where the central I/Q sweep response of the range cell generating the hit detection is associated with the neighboring I/Q sweeps on the discretized radial axis. Neighboring range cells could be carrying other hits themselves, integrating the initially processed hits into their own enriched I/Q format. Detecting a correlation in the Doppler content among neighboring ranges cells is actually critical for the effective discrimination of small and slow targets. Since weather phenomena and irrelevant objects also belong to the small and slow targets domain, a correlation between range cells close to each other may help discerning weather phenomena and other large scale motion. This is in particular due to the fact that interesting small and slow targets are too small to spread their Doppler signature across several range cells. The necessity of taking into account the relation between range cells within the enriched input representation depicted on Fig. 1.3 will have an impact on how we extract an encoding from this representation as explained in 2.4.2. An illustration of the intuition behind the motivation to identify any local correlation, and of the previously mentioned alternative defined by a spatio-temporal neighborhood of hits is proposed in Fig. 2.1.

The choice to encode a neighborhood of I/Q features makes our processing analogous to the processing of data generated by a sensors network. From this perspective, our neighborhood of range cells described by their respective I/Q signals could be a regularly-placed sensors network, the regular nature of the network stemming from the consistent range sampling. We find it particularly relevant to mention this sensors network interpretation of our problem since sensors network data processing seems of interest for a diversity of radar targets processing approaches. Indeed, in addition to our neighboring range cells I/Q sweeps, one could choose to process spatially or temporally neighboring hits, which in turn could be seen as processing data generated by an irregularly-placed sensors network. In that case, the irregularity comes from the fact that targets and clutter, and thus hits, are not necessarily regularly distributed in time and space around the radar. The latter approach, however, goes against the limitations of the radars we are working on as already mentioned, since it would imply processing hits combined and not individually. The apparent relevance of the sensors network interpretation of our input data naturally leads to considering graph neural networks (GNN) as they are a usual tool to process sensors network data [137]. The use of GNNs will be detailed in 2.4.2, but first we will go through encoding methods for a single range cell I/Q sweep in 2.3, and propose a simple baseline for the encoding of a neighborhood of range cells in 2.4.1. Since the neural networks models defined are built with complex-valued parameters, a short introduction to the implications of complex-valued neural networks is made beforehand in 2.2.

2.2 Processing complex-valued data with deep learning

Complex-valued neural networks (CVNN) were chosen to process the complex-valued I/Q signal input matrices of our discrimination task. As reminded in 1.3.5, such networks have existed for decades and were already used to process signals in their beginnings [86, 99, 91, 71, 112]. The recent popularity boom of deep learning was accompanied by a visible continued interest in CVNNs [22, 177, 179, 79]. This notably translated into the adaptation of the most common deep learning frameworks to CVNNs, for instance with PyTorch [4]. The need for CVNNs was actually strong enough among the application domains that smaller scales initiatives appeared to develop specialized pro-

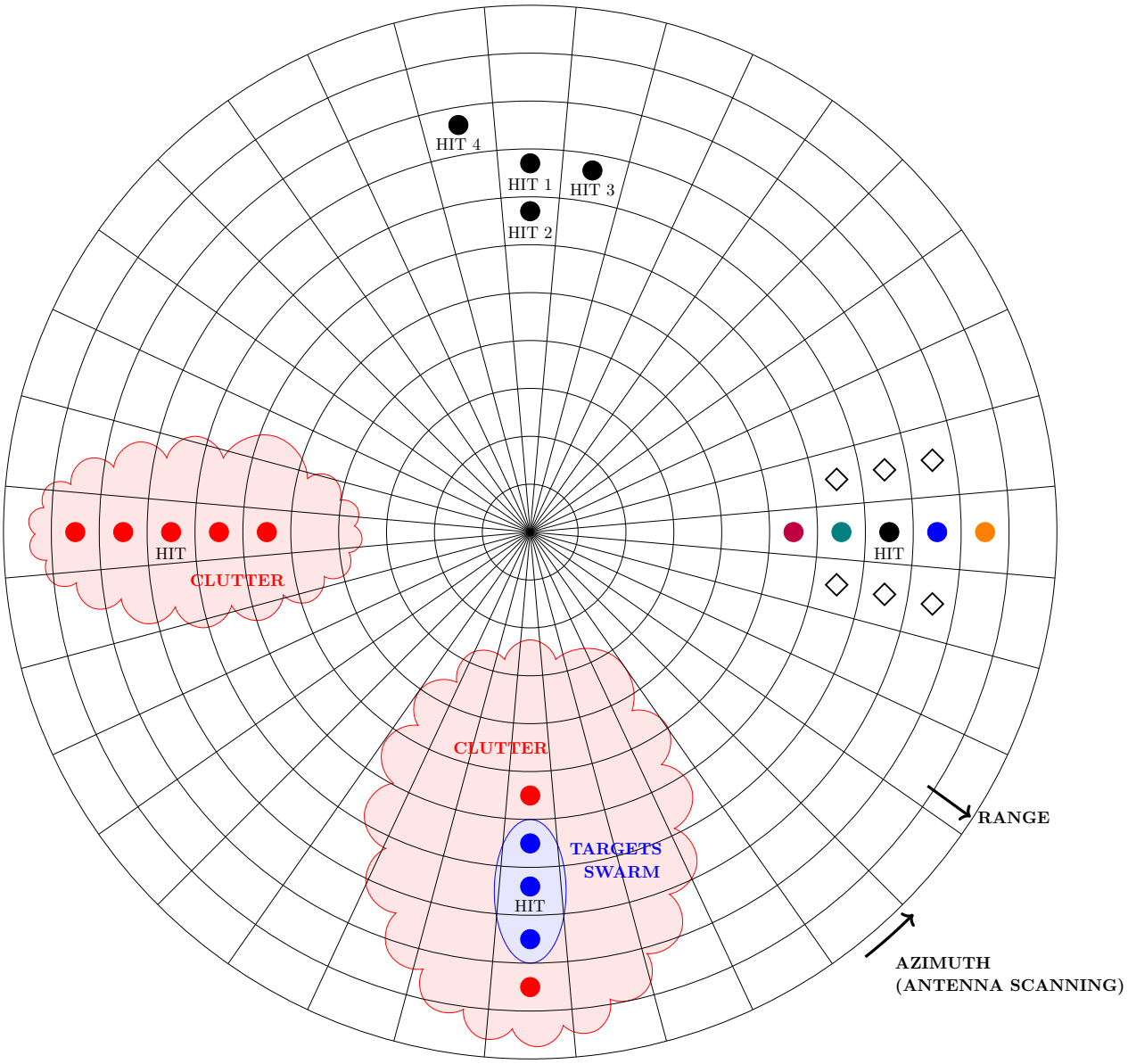


Figure 2.1: Example neighborhoods of range cells and spatio-temporal neighborhood of hits mentioned as a forbidden alternative to put our processing choice in perspective. The dot colors describe the nature of the Doppler content of the range cells, *i.e.* range cells filled with the same color contain similar Doppler signatures. Here, the neighborhood size is $H = 5$, as is the case in our experiments. **Right:** Generic enriched input format of our processing as described in chapter 1 (see Fig. 1.4). Each dot has a different color since by default the range cells Doppler content can be uncorrelated. We process a spatial neighborhood spanning over a radial axis and centered over the range cell that detected a potential target, *i.e.* whose content passed a threshold. The neighboring cells in azimuths are ignored, *i.e.* the cells marked with diamonds do not contribute any information to the neighborhood representation. **Left:** Neighborhood of range cells with highly correlated Doppler signatures. This means the neighboring cells and the central cell carrying the detection triggering the definition of a hit may contain the same kind of potential target. **Top:** Forbidden alternative to our enriched input representation, defined by a flexible neighborhood of hits. In such a neighborhood, each cell taken into account carries an actual hit, which is not necessarily the case in the fixed-size neighborhoods centered around each individual hit we chose as input **Bottom:**Intermediate neighborhood correlation case. The neighborhood embedding produced by the first hit2vec stage of our filter (see Fig. 1.10) should lead to the effective separation of this neighborhood and the two other valid ones represented here.

gramming libraries dedicated to CVNNs [21, 126][1], confirming the relevance of these neural network architectures. A distinct kind of radar, synthetic aperture radar (SAR), is among the common applications benefiting from CVNNs [20, 81]. Closer to our own application, CVNNs have been applied to complex-valued micro-Doppler spectral features in [37]. The use of trainable complex parameters is key to take advantage of the modulus and phase information contained within complex-valued data. To avoid the resort to proper complex analysis, the deep learning literature sometimes opted to take into account complex-valued data by doubling the number of real-valued channels of a neural network, treating the real and imaginary parts of a complex representation as two distinct features. Doing so doubles the number of inputs. This was also done with one channel for the modulus and a second one for the phase. However, such a neural network construction discards the simultaneous dependency of a complex representation on both values. The potential superiority of complex neural networks over real-valued equivalents in specific setups was confirmed in diverse experiments by the literature [20, 37].

2.2.1 Complex neural networks operations

The parameters of the linear operations building the neural networks of the encoding stage will all be complex-valued. The conversion of common deep learning operators to complex parameters is made quite simple by the modern deep learning libraries. In the case of PyTorch for example, it only requires a datatype conversion at the neural network initialization [5]. In 1.3.5, we mentioned the close relation between signal processing and deep learning operators with the example of the convolutions defining FIR filters. For instance for a real-valued FIR filter of order K applied to a real-valued input signal x we compute the filtered output y:

$$y(n) = \sum_{k=0}^{K-1} h(k)x(n-k) \tag{2.1}$$

This can rewritten using the convolution operator $*$:

$$y(n) = (h * x)(n) \tag{2.2}$$

This is also true in the context of complex-valued neural networks and complex-valued signals $z = z_r + jz_i$, where the subscript r indicates the real part of a complex representation, while the subscript i indicates the imaginary part. A complex-valued FIR filter $h = h_r + jh_i$ applied to the complex signal equally translates into the computation of a convolution, which can be developed as follows:

$$\begin{aligned} y(t) &= (z * h)(t) && (2.3)\\ &= (z_r + jz_i)(t) * (h_r + jh_i)(t) && (2.4)\\ &= (z_r * h_r - z_i * h_i)(t) + j(z_r * h_i + z_i * h_r)(t) && (2.5) \end{aligned}$$

One can note that the use of a complex filter allows for a non-symmetrical frequency response [1]. This contrasts with the application of a unique real-valued FIR filter to both the real and imaginary parts of an input complex-valued signal, and has a direct impact on the interpretation of the trained parameters of complex-valued convolutions. Such an interpretation is one way of explaining the relevance of the neural networks trained when they involve convolutions.

[1]Libraries accessible on GitHub: `https://github.com/NEGU93/cvnn` `https://github.com/wavefrontshaping/complexPyTorch` Accessed: 20/11/2022

In the experiments conducted, we considered two non-linear activation functions applicable to a complex input $z = z_r + jz_i$. The first one is a complex-valued adaptation of the usual rectified linear unit (ReLU):

$$\mathbb{C}ReLU(z) = ReLU(z_r) + jReLU(z_i) \tag{2.6}$$

where the usual ReLU is defined as follows:

$$ReLU(x) = \max(0, x) \tag{2.7}$$

The second activation function we considered for our CVNN architectures was the same adaptation where instead of the ReLU we used the Gaussian error linear unit (GELU) [84]:

$$\mathbb{C}GELU(z) = ReLU(z_r) + jGELU(z_i) \tag{2.8}$$

where the usual GELU is defined as follows:

$$GELU(x) = x\Phi(x) \tag{2.9}$$

where $\Phi(x)$ is the standard normal distribution cumulative distribution function. This activation function is actually approximated by the following expression:

$$GELU(x) = 0.5x(1 + \tanh(\frac{2}{\pi})(x + 0.044715x^3))) \tag{2.10}$$

The consideration of GELU was motivated by its use in the state-of-the-art wav2vec 2.0 [19] framework which also encodes raw signal, and due to the willingness to explore a more complex activation scheme which proved effective in other deep learning applications. Instead of simply gating inputs according to their sign like ReLU, GELU outputs an activation based on the value of the input. It also benefits from a probabilistic interpretation, and again in opposition to ReLU, is non-convex, non-monotonic and not linear in the positive domain. The two different element-wise applications of common RVNN activation functions to the real and imaginary parts of a complex-valued input corresponds to one of the two usual adaptations of activation functions to CVNNs. These two types are respectively called the split-complex and the joint-complex activation adaptations, and can be generically defined as follows [22, 81]:

$$\texttt{act}_{SPLIT}(z) = \texttt{act}(z_r) + j\texttt{act}(z_i) \tag{2.11}$$

$$\texttt{act}_{JOINT}(z) = \texttt{act}(|z|)\exp(j\arg(z)) \tag{2.12}$$

Following this naming convention, we opted for split-complex activation functions in Eq. (2.6) and Eq. (2.8). The joint-complex type processes the real and imaginary parts of the input jointly instead of separating them in the application of a non-linearity. We ended up privileging the $\mathbb{C}$ReLU activation since it was a common choice in the CVNN literature [22, 79] and it has been shown to perform better than alternative ReLU adaptations for CVNNs [177]. The definition of $\mathbb{C}$ReLU enables the function to choose to preserve the phase and the magnitude, or to project the phase to zero or $\frac{\pi}{2}$ to cancel the imaginary or the real part respectively, or alternatively to cancel both parts [177]. This follows the intuition of the real-valued ReLU which has a component-wise nullification power.

A similarly naive adaptation of the batch normalization (BN) [92] was also harnessed in the CVNN architectures put forward. Batch normalization was initially introduced to reduce the internal covariate shift, *i.e.* to control the input distribution fed to successive layers in neural networks. Batch normalization was meant to enable more efficient learning and reduce the criticality of the learned parameters initialization [177, 92]. It

was later shown that, although successful in improving the training process stability and performance, the BN success would actually be due to a smoothing effect on the optimization landscape [158]. The BN adaptation we used is the independent application of one real-valued BN to the real and imaginary parts of the inputs:

$$\mathbb{C}BN(z) = BN_r(z_r) + jBN_i(z_i) \tag{2.13}$$

Here, as opposed to the activation functions expressions, the input z is a batched input. A less naive complex-valued BN was proposed in [177] but was described as a limited source of performances improvement in the CVNN-focused programming library associated with [126][2]. It is important to keep in mind that this normalization operation, in addition to having a different behavior in the training and testing phases, also maintains two trainable parameters γ and β:

$$BN(x) = \gamma \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} + \beta \tag{2.14}$$

where x is a batched input features tensor. In the case of the naive adaptation to a CVNN application described by Eq. (2.13), this implies that the $\mathbb{C}BN$ has four trainable parameters $(\gamma_r, \gamma_i, \beta_r, \beta_i)$ instead of two.

2.2.2 Complex neural networks backpropagation

The adaptation of deep learning to complex-valued parameters involves redefining the usual linear and non-linear operations of neural networks as we have seen in 2.2.1. Once parameters are complex and the transformations of the forward pass are adapted to the complex format, the gradient also requires some adaptations to ensure a relevant backpropagation iteratively modifies the trainable parameters. In order to compute the necessary complex derivatives for training, CVNNs can harness the two Wirtinger derivatives [197], which for a complex-valued function $\theta : \mathbb{C} \to \mathbb{C}$ are:

$$\frac{\partial \theta}{\partial z} = \frac{1}{2}\left(\frac{\partial \theta}{\partial x} - j\frac{\partial \theta}{\partial y}\right) \tag{2.15}$$

$$\frac{\partial \theta}{\partial \bar{z}} = \frac{1}{2}\left(\frac{\partial \theta}{\partial x} + j\frac{\partial \theta}{\partial y}\right) \tag{2.16}$$

where $z = x + jy \in \mathbb{C}$. Thus, Eq. (2.15) and Eq. (2.16) define complex differentiation as a transposition of the real domain differentiation. These Wirtinger derivatives unlock the notion of complex derivative by making it applicable to non-holomorphic functions. Holomorphic functions are the functions usually associated with the definition of a complex differentiation, but are too restrictive to enable the transposition of RVNNs concepts and tools to the complex domain [22, 81, 4].

PyTorch [3][3], the library we mostly used, computes the gradient with respect to the conjugate $\frac{\partial}{\partial \bar{z}}$ to compute the gradient of a real-valued loss function to train a CVNN. Using a real-valued loss function is intuitive since there is no natural order among complex numbers, and we mostly want to minimize metrics during training.

In the complex domain and for a complex-valued input z, the chain rule of Eq.(1.13) critical to the backpropagation introduced in 1.3.4 becomes [81]:

$$\frac{\partial u(v(z))}{\partial z} = \frac{\partial u(v(z))}{\partial v(z)}\frac{\partial v(z)}{z} + \frac{\partial u(v(z))}{\partial \overline{v(z)}}\frac{\partial \overline{v(z)}}{\partial z} \tag{2.17}$$

[2]Library accessible on GitHub `https://github.com/wavefrontshaping/complexPyTorch` Accessed: 20/11/2022

[3]Library accessible on GitHub: `https://github.com/pytorch/pytorch` Accessed: 20/11/2022

where $u : \mathbb{C} \to \mathbb{C}$ and $v : \mathbb{C} \to \mathbb{C}$ are arbitrary complex functions. This chain rule can be rewritten to make a real-valued loss function $\mathcal{L} : \mathbb{C} \to \mathbb{R}$ and the gradient with respect to the conjugate computed by some of the common deep learning libraries as mentioned earlier [22]:

$$\frac{\partial \mathcal{L}(v(z))}{\partial \bar{z}} = \frac{\partial \mathcal{L}(v(z))}{\partial v_r(z)} \frac{\partial v_r(z)}{\bar{z}} + \frac{\partial \mathcal{L}(v(z))}{\partial v_i(z)} \frac{\partial v_i(z)}{\partial \bar{z}} \tag{2.18}$$

2.3 Encoding a single hit range cell

Let us now discuss the encoding of a single I/Q signal, *i.e.* the generation of an embedding for a single range cell described by a single burst of pulses. This task amounts to encode a column $(z_{11} \dots z_{N1})^T \in \mathbb{C}^{N \times 1}$ of the complex-valued matrix $Z_{I/Q}$ defined in Eq. (1.2). As indicated by Fig.1.11, we aim at producing a single embedding $\mathbb{R}^Q$ for the enriched input defined by the complete matrix. A single I/Q signal will first be mapped to a real-valued vector in $\mathbb{R}^Q$, combinations of such vectors to form a neighborhood embedding being proposed in section 2.4. As put forward in section 1.3.5, a naive way to encode the diversely sampled input signals in comparable representations would be to resample them and use their subsequent and equally sized representation in the Fourier domain to compare them. This approach comes with the limitations posed by the resampling method. The reader may refer to section 1.3.5 for details in that regard. Here, we will first review an approach recently proposed by [43] precisely developed for radar backscatter processing which can be interestingly coupled with an upcoming manifold-aware OCC presented in chapter 3. We will then go on with recurrent and fully convolutional neural networks (FCNN), with which we conducted preliminary experiments. Both the RNN and the FCNN will be used to define unsupervised range cell encoding thanks to generative architectures in section 2.3.2, and will then be extended to take advantage of labels and supervision during training in section 2.3.3. Each of these methods is adapted to the processing of input signal with different number of input signal samples. In opposition to the resampling approach, these methods do not explicitly take into account the varying sampling frequency of the input signals, *i.e.* the PRF. This can be rectified later for our neighborhood of range cells as we will see in section 2.4.

2.3.1 A straightforward approach with AR models

One way of representing the complex-valued I/Q radar signals is to use their autocorrelation matrix. This was done in [44] to encode PDR range cells just like what we wish to do. For a 1D I/Q signal z with N time steps sampled $(z_1, ..., z_N)$ in compliance with the notations of Eq. (1.2), the autocorrelation coefficient r_t can be defined as follows for a time lag of t time steps:

$$\begin{aligned} r_t &= \mathbb{E}\left[z(k+t)\bar{z}(k)\right] && (2.19)\\ &= \mathbb{E}\left[z(k)\bar{z}(k+t)\right] && (2.20)\\ &= \overline{\mathbb{E}\left[z(k-t)z(k)\right]} && (2.21)\\ &= \overline{r}_{-t} && (2.22) \end{aligned}$$

The signal is assumed to be stationary, turning the computation of a correlation into a function of the time steps lag. This directly leads to the definition of the Toeplitz autocorrelation matrix for an arbitrary order of time steps lag, as long as there are enough signal samples (N large) to estimate the individual correlations. For an autocorrelation

of up to t time steps lag, the autocorrelation matrix is:

$$R_t = \begin{pmatrix} r_0 & \overline{r}_1 & \overline{r}_2 & \dots & \overline{r}_{t-1} \\ r_1 & r_0 & \overline{r}_1 & \dots & \overline{r}_{t-2} \\ r_2 & r_1 & r_0 & \dots & \overline{r}_{t-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{t-1} & r_{t-2} & r_{t-3} & \dots & r_0 \end{pmatrix} \tag{2.23}$$

The autocorrelation matrix is additionally Hermitian positive definite (HPD), which is the complex-valued equivalent of a symmetric positive definite matrix. An SPD matrix would be processed instead of an HPD one had the input signal or time-series been real-valued. The autocorrelation coefficients can be estimated using the empirical mean, for each t:

$$\widehat{r_t} = \frac{1}{N-t} \sum_{k=0}^{N-1-t} z(k+t)\overline{z(k)} \tag{2.24}$$

Estimating the autocorrelation matrix through the empirical mean may however produce a matrix R that is not hermitian positive definite. The relevance of the autocorrelation coeffcients estimated with the empirical mean $\widehat{r_t}$ of Eq. (2.24) depends on the length of the input signal, which translates into the statistical significance of the mean estimation.

To grasp the representative power of an autocorrelation matrix, one can think of white noise which yields zero as autocorrelation for any time steps lag. The autocorrelation matrix is directly related to the autoregressive (AR) coefficients of an equivalent AR model. The AR model is said to be equivalent since knowing the autocorrelation coefficients suffice to determine the AR coefficients. The relationship between both kinds of coefficients is defined by the Yule-Walker equation [43]. Assuming a signal z with zero mean and an AR model of order P, the AR coefficients $(a_0, \dots, a_P)$ are the coefficient minimizing the mean square prediction error E as defined by the following expression:

$$\sum_{p=0}^{P} a_p z_{N-p} = E_N \tag{2.25}$$

The autocorrelation can appear in a development of Eq. (2.25) to produce the following expression linking the AR coefficients to the autocorrelation ones up to an autocorrelation order t, under the hypothesis that the mean square prediction error E is white noise:

$$\sum_{p=0}^{P} a_p r_{t-p} = 0 \tag{2.26}$$

With $a_0 = 1$ since the autocorrelation with itself is always 1, we end up with an expression allowing for the computation of any order of autocorrelation based on the autocorrelation coefficients of lower order and on the associated AR model coefficients, which can be called reflection coefficients:

$$r_t = -\sum_{p=1}^{P} a_p r_{t-p} \tag{2.27}$$

Using Eq. 2.27 to compute each autocorrelation coefficient up to the order P of the AR model, one finally ends up with the Yule-Walker equation which makes the autocorrelation matrix of Eq. (2.23) reappear:

$$\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_P \end{pmatrix} = - \begin{pmatrix} r_0 & \overline{r}_1 & \overline{r}_2 & \dots & \overline{r}_{P-1} \\ r_1 & r_0 & \overline{r}_1 & \dots & \overline{r}_{P-2} \\ r_2 & r_1 & r_0 & \dots & \overline{r}_{P-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{P-1} & r_{P-2} & r_{P-3} & \dots & r_0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{pmatrix} = -R_P A_P \tag{2.28}$$

The complete Yule-Walker expression of Eq. (2.28) now makes the bijection between the two families of coefficients clear: if the matrix R_P is invertible the AR model coefficients can be computed from the autocorrelation coefficients, while the reversed dependency is directly ensured by Eq. (2.28). Furthermore, the Levinson algorithm can be used to recursively compute the AR model coefficients, or reflection coefficients, of AR models of successive orders. Autoregressive models of different orders are thus iteratively linked, and the knowledge of the AR coefficients of lower order models enable the computation of the same coefficients for a higher order model. The bijection between the reflection and autocorrelation coefficients, as well as the recursive relationship among reflection coefficients associated with AR models of successive order enable to bypass the drawbacks of estimating the autocorrelation matrix through the empirical mean (see Eq. (2.24)). Indeed, algorithms such as the Burg algorithm [42] can be used to directly estimate reflection coefficients using the minimization of forward and backward prediction errors. Details regarding these developments, and adaptations of the previous expressions to multivariate time-series and signals can be found in [43]. One of the challenges of using AR models is to find a relevant AR model order P. The search of P translates into finding up to what order the autocorrelation coefficients have a significant value.

The representation of I/Q radar signals with an autocorrelation matrix will not be experimented with since our focus was set on complementary approaches to the works of [43] and [32]. Harnessing the autocorrelation coefficients as a single range cell representation could have still provided us with a relevant input representation baseline with respect to our raw I/Q signal input representation. The computation of autocorrelation matrices can yield representations constrained to the Riemannian manifolds of real-valued symmetric positive definite matrices, or to the Riemannian manifold of hermitian positive definite matrices. Processing radar data on such manifolds and on other manifolds whose points are built thanks to the autocorrelation and AR model coefficients has been continuously explored in [32, 34, 108, 203] during the last decade. To retrieve a single range cells embeddings belonging to $\mathbb{R}^Q$ that can be combined to form a single $\mathbb{R}^Q$ embedding as required by the pipeline of Fig. 1.10, one could use manifold-aware neural networks taking the representation belonging to the manifold as input and producing the fixed-size vector as output. Manifold-aware and specifically SPD manifold-aware deep learning are further discussed in chapter 3 in 3.2. We can additionally note that the Burg algorithm mentioned to determine reflection coefficients is based on the minimization of prediction errors, a task now commonly achieved with deep learning approaches.

2.3.2 Sequence-to-sequence models encoding

In the input $\mathbb{C}^{N\times H}$ matrix gathering H signals of N samples, we can consider each range cell to be a complex-valued time-series. This leads us to consider the recurrent neural networks usually applied to time-series in the deep learning community. Let us remind what such neural networks entail. The most simple recurrent neural network recursively combines an input x and a hidden state h at time step t:

$$h_t = \texttt{act}\left(W_x x + b_x + W_h h_{t-1} + b_h\right) \tag{2.29}$$

where W_x controls what information to extract from the current time step input and W_h controls what to keep from the previous hidden state. In this subsection notations otherwise usual in the remainder of this manuscript can be set aside due to the specific nature of recurrent architectures. For instance, samples are indexed by a t for time step instead of an index i. The previous hidden state ideally keeps the relevant information from the previous states. This output can then be further transformed by an affine

transformation and an non-linear activation function, or simply fed to the next time step to compute h_{t+1} within the same recurrent cell. In Eq. (2.29), typical activation choices are the rectified linear unit (see Eq.(2.7)) or the hyperbolic tangent:

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{2.30}$$

These activation functions are applied element-wise on vector or tensor representations within neural networks. An illustration of the recurrent nature of the RNN defined by Eq. (2.29) is proposed on Fig. 2.2.

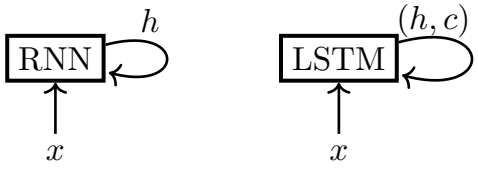


Figure 2.2: RNN principle for a simple RNN cell (left) and for a more complex LSTM cell (right). The RNN cell maintains one hidden state h_t that is passed to the next time step, while the LSTM cell passes two hidden states h_t and c_t.

Recurrent neural networks based on Eq. (2.29), which we will call vanilla RNN, are challenging due to the vanishing or exploding gradients appearing in training and the difficulty of learning long-term dependencies [54]. To tackle the challenges posed by the training of vanilla RNNs, a more complex recurrent unit called long short-term memory (LSTM) was proposed in [87]. This recurrent unit combines operations acting as information and gradient gates to control the flow of information through successive time steps. As for the vanilla RNN described by Eq. (2.29), a hidden state h_t is updated through successive time steps, along with an additional cell state c_t and other complementary operations. These operations define the following gates: an input gate i, a forget gate f, a cell gate g and an output gate o. The output of each gate is computed according to the following equations[4]:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{2.31}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{2.32}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{2.33}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho} \tag{2.34}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.35}$$
$$h_t = o_t \odot \tanh(c_t) \tag{2.36}$$

In these equations, the subscript letters indicate to which gate and input and intermediate representation weights are applied. Input and intermediate representations are associated with a time step subscript. The symbol σ stands for the sigmoid function:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2.37}$$

The distinction between the four gates put forward can seem confusing at first due to their output being systematically based on the same input representations: the current time step input x_t and the previous time step hidden state h_{t-1}. Each of the gates

[4]Here we pick the notation of PyTorch documentation. The literature sometimes combines the double bias terms in the affine inputs in a single bias to produce a shorter and equivalent expression.

however has its own role and its own trainable weights, in addition to a distinct non-linear activation for the cell gate g. The LSTM cell passes the two hidden states h_t and c_t to the next time iteration. An illustration of the recurrent nature of the LSTM defined by Eq. (2.31) is proposed on Fig. 2.2. A prominent gated recurrent unit variant is the gated recurrent unit (GRU) [51]. The GRU combines three gates instead of four like in the LSTM; a reset gate r, an update gate z and a new gate n. Unlike the LSTM, a single hidden state h_t is passed to the next iteration of the gates. The gates output are determined as follows[5]:

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \tag{2.38}$$
$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \tag{2.39}$$
$$n_t = \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{t-1} + b_{hn})) \tag{2.40}$$
$$h_t = (1 - z_t) \odot n_t + z_t \odot h_{t-1} \tag{2.41}$$

Achieving unsupervised representation learning with the recurrent cells described is enabled by the definition of sequence-to-sequence (seq2seq) generative architectures where one RNN encodes the input signal and another RNN decodes the last hidden states of the encoder to reproduce the input. This is the sequence equivalent of an autoencoder discussed in chapter 1. The encoder and decoder RNN are separate architectures with their own trainable weights. Several recurrent cells can be stacked in layers, and the size of the hidden states provides us with a way to control their expressive power and learning potential. To generate a real-valued fixed-size representation in $\mathbb{R}^{\mathbb{Q}}$ for the input signal $(z_1 \dots z_N)^T \in \mathbb{C}^{N\times 1}$ describing a single column of the complex-valued matrix $Z_{I/Q}$ defined in Eq. (1.2), the last hidden state of a seq2seq encoder is reduced to its real part. The imaginary part is discarded and only real-valued coefficients are passed on to the decoder. This seq2seq architecture can be implemented with any of the diverse recurrent neural networks cells available in the literature, and in particular with the three recurrent setups described. The generative seq2seq approach with RNNs and its application to range cell signal encoding is illustrated on Fig. 2.3. Encoding signals with seq2seq architectures built with RNNs is a well established application of deep learning in the literature, for example to process electrocardiogram (ECG) data [88]. An LSTM-based seq2seq architecture is proposed in the appendix of unsuccessful models, in table C.1.

A different kind of generative sequence-to-sequence neural network can be adapted to our input data. One of the difficulties of our inputs is its variable size. To tackle it, one can harness a global pooling [117] in a generative fully convolutional neural network [149, 120]. The generative FCNN uses convolutions, normalization, activation and pooling layers without the intervention of dense or fully-connected layers to accept inputs of varying size and produce a fixed-size representation in $\mathbb{R}^{\mathbb{Q}}$ in its bottleneck. One of the key advantages of applying convolutions to signal inputs is the interpretability potential of the learned weights. If the convolution kernel is large enough the parallel with FIR filter becomes practical. In such a case, one should also keep in mind that having numerous independent convolutional kernel, *i.e.* numerous channels for the convolutional layer, can be relevant to tackle the difficulty of handling input signals with varying sampling frequency. Indeed, a given FIR filter will define a certain frequency response depending on its weight and the sampling frequency of the signal to which the weights are applied. Thus, to ensure similar frequency responses can be extracted from filters applied to input signals with different sampling frequencies, one should stay generous with the number of channels available in convolutional layers. The generative FCNN can be thought as closer to the autoencoder architecture than to the recurrent

[5] Again, we pick the notation of the PyTorch documentation.

seq2seq one, and will thus be called fully convolutional autoencoder (FCAE). The bottleneck is the intermediate representation of lower dimensionality, assuming one defines an undercomplete [77, p.494] generative architecture, which separates the encoder from the decoder.

Global max or average pooling allows an FCAE to reconstruct output of different sizes while producing a bottleneck representation of fixed-size. In order to do so, the global pooling operation is applied to the bottleneck features dimension and only happens to retrieve an $\mathbb{R}^Q$ embedding. In a forward pass on the other hand, the features dimension, whose size varies along with the input size, is left unreduced. This implies the Q dimensions of the latent representation defining the range embedding are equal to the number of convolutional channels in the bottleneck representation. The FCAE proposed here combines an encoder and a decoder with complex-valued parameters. However, as for the previously detailed RNN-based seq2seq architecture, the bottleneck representation will be constrained to real values in order to learn real-valued latent representations. One of the advantages of the FCAE with respect to the previously introduced RNNs is that the processing of all time steps is done in parallel instead of sequentially. As for RNN-based seq2seq architectures, convolution-based seq2seq architectures define a popular option to encode signals, for example to process ECG data [206]. In our case as well as in the case of ECG encoding, one could arguably talk about signal2signal architecture. An FCAE architecture adapted to our I/Q signals is proposed in table C.2.

Both seq2seq approaches, either recurrent or fully convolutional, train their neural networks parameters by minimizing the reconstruction error computed over the signal samples $(z_1 \dots z_N)^T \in \mathbb{C}^{N\times 1}$ describing a single column of the complex-valued matrix $Z_{I/Q}$ defined in Eq. (1.2):

$$\min_W \left[\frac{1}{N} \sum_{i=1}^{N} (\Phi(z_i; W) - z_i)^2 \right] \tag{2.42}$$

where Φ is the generative neural network of trainable weights W producing the complex-valued reconstruction of the input signal samples. In the computation of the previous error, the real and complex parts of the complex representation are taken into account like two components of a real-valued representation. This leads to a real-valued loss despite the complex-valued nature of the generative neural network, as announced in 2.2.

The loss here is defined at the scale of a single range cell I/Q signal, sampled over N pulses. This loss is computed over batches of multiple such signals during a neural network training. One can note that the reconstruction task can be coupled with a denoising or a missing input interpolation task. The latter is made particularly easy by the availability of dropout [171][6] in deep learning libraries, although such dropout should not apply any scaling as the common dropout layer does. Now that we have discussed the possibility to encoder single range cell IQ signal with AR models, RNN and FCNN-based generative neural networks, what is there to say to differentiate them ? A simple way to put it could be to say that whereas the AR models reduces the representation of the signal to an autocorrelation Toeplitz matrix of a carefully chosen order, the RNN-based seq2seq approach is more about choosing the right recurrent neural network complexity. Similarly, the FCAE can also be summarized to a neural network architecture choice, however here the model offers the interpretability potential of the convolutions weights as FIR filters. All three approaches benefit from the application cases and successes offered by the literature. As reminded in 2.3.1 and in the current section, processing radar signal with AR models has been put forward in [43] and ECG signal processing

[6] Dropout randomly discards elements of representations in a neural network to prevent overfitting and excessive co-adaptation of trained weights. It defines a form of regularization very popular in deep learning architectures.

can be done with recurrent neural networks [88]. The application of successive layers of convolutions to raw signal has also had its successes on audio [19, 163], ECG [206] and radio signal [119, 138], making the case for the FCAE method.

2.3.3 Taking advantage of limited supervision to avoid generative encoding

A prominent advantage of encoding a range cell signal using either the Toeplitz autocorrelation matrix of an AR process as described in 2.3.1 is that the range cell encoded representation is produced without supervision. This advantage also applies to the sequence-to-sequence recurrent or fully convolutional architectures described in 2.3.2: since the training task is reconstruction, possibly coupled with a denoising or missing input completion task, the model training does not require class labels. Since the encoding task is difficult, it appears relevant to propose neural network architectures and training setups able to take advantage of labels when the latter are available. We will therefore discuss two setups adapted to two distinct level of supervision. The first one will remain on a generative architecture but will take advantage of a limited amount of labels to penalize distances relative to class centroids within its latent space. The second one will assume sufficient supervision is available to eliminate the reconstruction training task altogether, in order to only harness the encoder part of the generative networks discussed in 2.3.2.

The first setup corresponds to the following loss:

$$\min_{W} \left[\frac{1}{n} \sum_{i=1}^{n} ||\Phi(z_i; W) - z_i||^2 + \frac{1}{m} \sum_{j=1}^{m} ||\Phi_E(z_j; W_E) - c_j||^2 \right] \tag{2.43}$$

Here, unlike in Eq. 2.42, z_i is the $i-th$ input range cell signal described by a vector, and not a single sample of one input signal, and $\Phi(z_i; W)$ is a vector containing its reconstruction by the generative neural network Φ of weights W. In the second term, $\Phi_E(z_j; W_E)$ is a vector containing the embedding of the labeled sample z_j, and c_j is the reference point or centroid associated with its label in the embedding or encoding space. Since the encoding network is part of the generative architecture, the trainable weights W_E are part of W, hence the sole presence of W under the minimization operator. As the embeddings of our framework live in $\mathbb{R}^Q$ (see Fig. 2.4) both $\Phi_E(z_j; W_E)$ and c_j necessarily belong to $\mathbb{R}^Q$ as well. The sums over n and m correspond to a loss computed for a set of n unlabeled and m labeled samples. This loss thus indicates a semi-supervised learning setup, in accordance with the semi-supervised learning appreciation of [152].

The two terms defined thus respectively relate to a reconstruction error and a latent clustering constraint. The association of a reconstruction term and a latent space clustering term is similar to what the literature of simultaneous clustering and representation learning already proposed. In [170] for instance, an autoencoder is trained with a less where one term minimizes the reconstruction error and another term minimizes the distance to assigned cluster centers. A key difference with respect to our proposal however remains: [170] additionally suggests an alternate optimization of the mapping neural network and of the cluster centers. Here, we simply propose c_j as targets classes mean representation or arbitrary coordinates towards which embeddings should be mapped. On a side note, in deep learning a regularization term is usually added to the training loss functions. For instance, the L2 norm of the neural network weights is added to the loss with a dedicated fixed weight [77]. Such a term will not appear in the loss functions discussed here.

Now that the first setup has been presented, let us turn to the second one. The concentration of latent representations of specific labels around centroids proposed as a

Layers in forward order	Layer parameters
$\mathbb{C}$-conv 1D	kernel size 10, in channels 1, out channels 8
$\mathbb{C}$-BN 1D	channels 8
$\mathbb{C}$-ReLU	channels 8
$\mathbb{C}$-conv 1D	kernel size 2, in channels 8, out channels 16
$\mathbb{C}$-BN 1D	channels 16
$\mathbb{C}$-ReLU	channels 16
Mean pooling	mean over features dimension
$\mathbb{C}$-Linear	input dimension 16, output dimension 16
$\mathbb{R}$-Output	input dimension 16, output dimension 16

Table 2.2: Overview of the FCN architecture used in our experiments, which begins by applying a large convolutional kernel to the input signal. Using a convolution with a large kernel size on the input signal is particularly interesting since it makes the potential interpretation of the learned weights as FIR filter coefficients more expressive. The mean pooling layer is applied over the features dimension, so that the remaining representation is a vector whose size equals the number of channels of the final convolutional layer. The final layer discards the imaginary part to produce the real-valued $\mathbb{R}^Q$ vector representation of the range cell. Since only the real part is taken into account in the training loss, the network learns to concentrate the information in the real part only.

complementary loss term in Eq. (2.43) could actually suffice to train an encoder neural network alone directly, assuming enough labeled data is available. This would lead to a setup close to supervised classification, although it would not necessary be the case for our encoding task. For instance, there could be known classes samples with yet unseen signal sampling parameters in the test set, resulting in a not completely supervised machine learning setup despite the existence of the right centroid for the test samples label. In this higher supervision setup, one can harness the following loss:

$$\min_{W} \left[\frac{1}{m} \sum_{j=1}^{m} ||\Phi_E(z_j; W_E) - c_j||^2 \right] \tag{2.44}$$

where the loss terms and indices correspond to the definitions provided for Eq. (2.43). The possibility to use such a loss to directly train the encoder would allow to discard the decoder in the architectures presented in section 2.3.2. This could simplify the trained architecture but it does not make the right centroid selection heuristic obvious. In the case of the recurrent seq2seq generative architecture, one ends up with a recurrent encoder whose last hidden state is reduced to its real part, the latter now defining the final representation in a forward pass of the neural network trained. In the case of the FCAE autoencoder, one ends up with a fully convolution network (FCN) that retains the potentially interpretable convolutional layers. An example architecture for such an FCN trained with the loss provided by Eq. (2.44) is proposed in table 2.2.

2.4 Encoding a neighborhood of range cells

Now that we have investigated the encoding of single range cells each described by an I/Q signal, let us go back to the more elaborate problem of encoding our actual input, *i.e.* the neighborhood of ranges cells (see Fig. 1.4 and Fig. 2.1). The task at hand consists in encoding the input matrix defined by Eq. (1.2) belonging to $\mathbb{C}^{N \times H}$, where N is variable among the input data points due to the varying number of pulses in the

transmitted bursts. In the latter, each column describes a range cell. Since all single range cells are first encoded into a fixed-size real-valued vector in $\mathbb{R}^Q$, the remaining transformation to produce a single real-valued vector in $\mathbb{R}^Q$ for the entire neighborhood should define a function $\Phi : \mathbb{R}^{Q \times H} \to \mathbb{R}^Q$. The definition of such a function Φ is the matter of upcoming sections and is illustrated on Fig. 2.4.

We will first mention naive approaches to combine the single range cells respective embedding and discuss the closely related subject of combining independent range cells to produce an artificial dataset respecting our input format in 2.4.1. We will then present the favored approach based on a graph neural network in 2.4.2. One can note that the inquired function Φ correspond to the mean of the Fourier features after resampling in the baseline free of machine learning proposed for perspective in Fig. 1.12. Assuming a resampling scheme could successively provide fixed-size $\mathbb{R}^Q$ single range cells embeddings, any embeddings combination approach presented here could also be integrated to this resampling baseline for the neighborhood embeddings integration subtask. The methods presented here could also extend to the combination of embeddings based on an individual range cell information different then the one carried by the I/Q signal, as suggested by the well-identified sensors networks data processing application of graph neural networks [137].

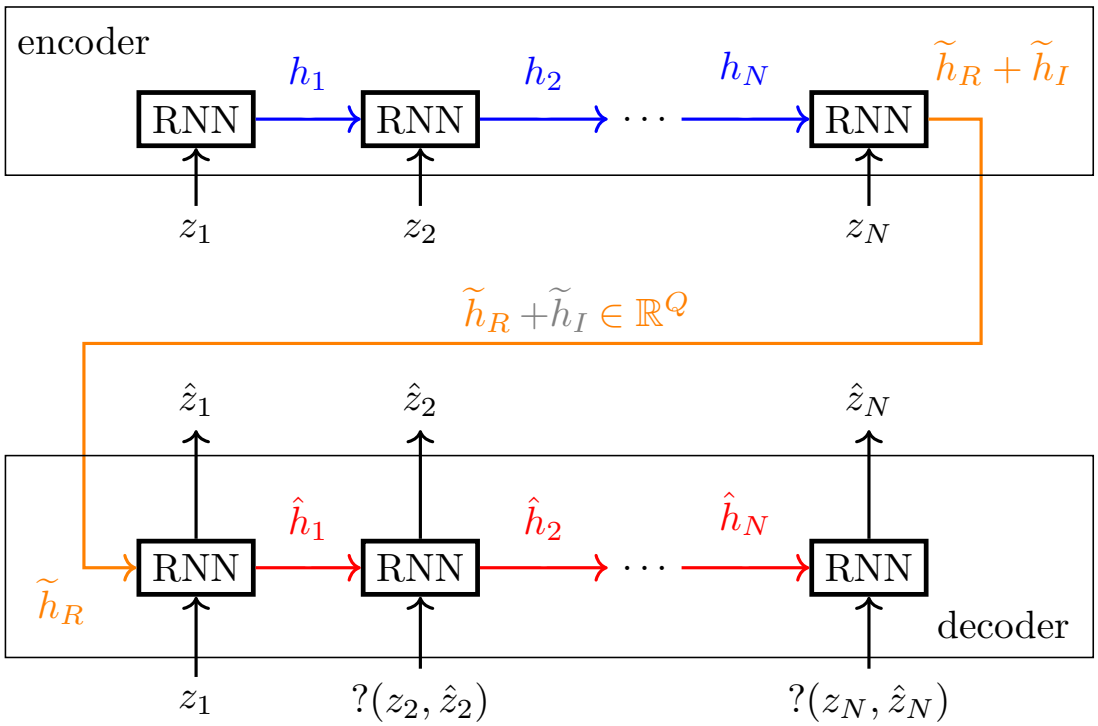


Figure 2.3: Unsupervised seq2seq learning with recurrent neural networks to retrieve the fixed-size vector of $\mathbb{R}^Q$ encoding each range cell signal. The upper part describes the encoder, the lower part describes the decoder. The blue and red arrows illustrate the recursive calls of the recursive architectures to their previous hidden states in the encoder and the decoder respectively. The last hidden state of the encoder $\widetilde{h_R} + \widetilde{h_I}$ is passed on to the decoder with its imaginary part zeroed out, thus creating the $\mathbb{R}^Q$ single cell embedding we are looking for. Indeed, since the only way to provide information to the decoder is to put it in the real part of the hidden state, the seq2seq trainable weights are forced to learn the complex to real conversion. This does not forbid the decoder to adopt complex-valued weights in order to output the reconstructed input signal $\hat{z}$. The question mark among the inputs of the decoder $?(z_N, \hat{z}_N)$ denotes teacher forcing [77, p.372], the first input of the decoder being the ground truth and no end-of-sequence (EOS) token being used. We allowed ourselves to work without start-of-sequence (SOS) and EOS tokens [174] since unlike in a sentence, the positioning towards the beginning of the end of a signal does not translate into semantic information. This is true due to the fact that we are interested in the discriminative frequency content spanning over the whole sequence.

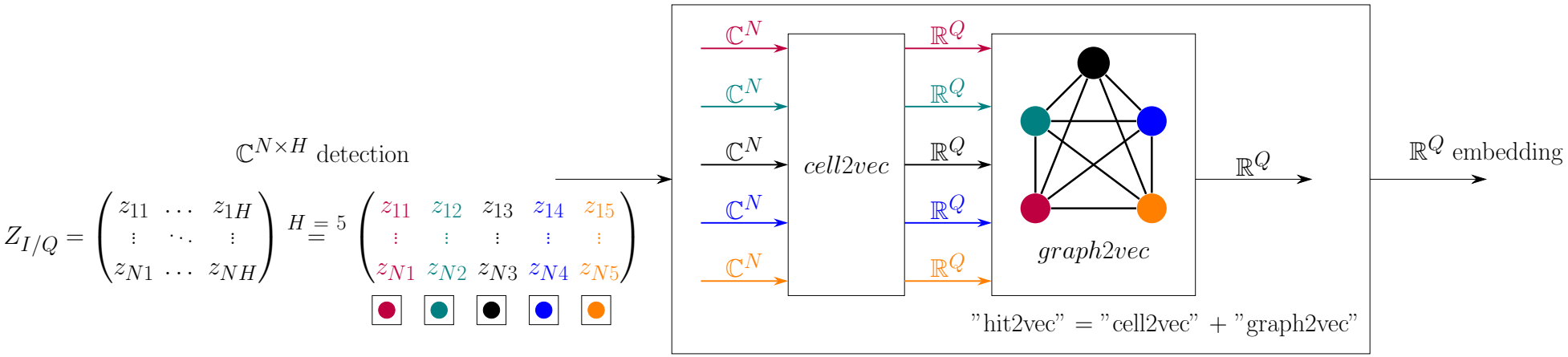


Figure 2.4: Proposed hit2vec step combining a cell2vec step and a graph2vec step. The cell2vec step, discussed in 2.3, corresponds to encoding a single range cell and transiting from a variable size complex-valued representation to a fixed-size real-valued one. The graph2vec step, discussed in 2.4.2, corresponds to encoding the graph of the range cells neighborhood where each range cell defines a node whose features is the $\mathbb{R}^Q$ embedding provided by cell2vec. The graph2vec relies on an adjacency matrix defining the range cells neighborhood connectivity pattern and allowing for the definition of a graph convolution through Eq. (2.48). Example neighborhood graphs are proposed on Fig. 2.6, Fig. 2.8 and Fig. 2.5. The input matrix is the enriched hit input format defined by Eq. (1.2), and the output graph embedding is passed on to a one-class classification method for low supervision discrimination.

2.4.1 Naive approaches and the possibility of data augmentation

A naive way of combining the encoded representations of each range cell to implement Φ is to use a simple mean computed over each one of the Q components provided by the single cells:

$$\Phi(R_{single}) = \frac{1}{H} R_{single} I_Q \tag{2.45}$$

where R_{single} is the $\mathbb{R}^{Q \times H}$ matrix containing the single cells embeddings of the neighborhood of H range cells, and I_Q is a $\mathbb{R}^{H \times 1}$ vector of ones. This mean can be adjusted with fixed or trainable weights over the H cells, which leads to the following alternative for Φ:

$$\Phi(R_{single}) = \frac{1}{H} R_{single} A_Q \tag{2.46}$$

where A_Q is a $\mathbb{R}^{H \times 1}$ vector:

$$A_Q = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_Q \end{pmatrix} \tag{2.47}$$

These weights can be interpreted as a naive form of attention spanning over the neighborhood of range cells, in reference to the deep learning concept of attention [185] already used in generative [75] and graph neural networks architectures [188]. An important difference remains with the deep learning attention, since the latter produces a weight based both on a vector representation key and a vector representation value. Assuming a large neighborhood is handled, a sparsity constraint could also be enforced to combine single cell embeddings on as was done on encoded representations in the context of a generative task in [75]. Such a sparsity constraint over a neighborhood of range cells could be inspired by or interpreted as an extension of the guard cells radar concept already regulating the contribution of neighboring cells for radar constant-false-alarm rate (CFAR) detection [18, 15]. The matrices R and A here have no relation with the autocorrelation matrix R and the autoregressive coefficients of 2.3.1.

In terms of deep learning experimentation, one can note that producing classes of neighborhood of range cells with diverse levels of local relative correlations can be done through the combination of single range cells even when they are not identified as neighbors. This can substantially ease the generation of a sizeable dataset to train and evaluate the proposed approaches with the enriched input format on already existing and not modifiable platforms. Indeed, such a dataset generation possibility allows to avoid a very inconvenient modification of the hits format produced by the signal processing part of the radar processing pipeline (see Fig. 1.1) by unlocking the artificial construction of neighborhoods of range cells of arbitrary size H from the already existing single range cell format. This range cells recombination is equally applicable when the hit format deployed on a system already produces a neighborhood of range cells but this neighborhood is not of the desired size H. Here are a few examples of possible range cells combinations to produce different local correlation configurations:

- Replicate a single range cell to create a perfectly correlated neighborhood. This translates into the pattern $A \dots A$, where one letter is one class or type of pattern in terms of Doppler content. An example of this pattern for a neighborhood of size $H = 5$ is illustrated on Fig. 2.1, on the left part.

- Replicate one range cell in the central part of the neighborhood, then another one on the border cells. This creates a symmetric neighborhood with two group of cells perfectly correlated, and translates into the pattern $B \dots BA \dots AB \dots B$.

An example of this pattern for a neighborhood of size $H = 5$ is illustrated on Fig. 2.1, on the bottom part.

- If the number of types of range cells available exceeds the number of range cells in the neighborhoods, one can define an uncorrelated, asymmetric neighborhood. This translates into the pattern $ABC\ldots$. An example of this pattern for a neighborhood of size $H = 5$ is illustrated on Fig. 2.1, on the right part.

With these example classes, one can ensure the classes of neighborhoods separated already represent some level of diversity of targets in the hit central range cell responsible for the detection. Evidently, the choice of the types of neighborhood artificially generated, when they are needed, should depend on the downstream discrimination task addressed. Furthermore, when building neighborhoods as well as for the development of neighborhood encoding methods, one should keep in mind that the relative position with respect to the range cell matters, but not the absolute range. On the other hand, the absolute position of the range cells with respect to the ranges axis should not matter. While creating a dataset of labeled neighborhoods of range cells, one should also not lose track of the actual objective which is the discrimination of the target described by the central range cell of the neighborhood. This should translate into a systematic greater relative contribution of the central range cell to the final neighborhood representation.

2.4.2 A deep learning encoding with graph neural networks

The solution proposed by this thesis to encode a neighborhood of encoded range cells I/Q sweeps relies on a neighborhood graph which will be processed by a graph neural network [161, 78]. As for the FCNN and the seq2seq architectures, let us first remind how a graph neural network works. A GNN takes a set of nodes $\mathcal{N}$ and a set of edges $\mathcal{E}$ as input and transforms these elements, *i.e.* their features, with both linear and non-linear operations constrained by the connectivity pattern of an adjacency matrix. Classification and regression tasks can thus be conducted on graphs at node scale as well as on a whole graph. This notably translates into the classification of individual nodes whose features have been processed by graph neural networks, and into the classification of graphs whose representation stems from an aggregation of processed nodes features. Generative architectures adapted to data distributed on graphs have recently been proposed, mirroring the Euclidean deep learning community findings with graph autoencoders [154] and graph UNets [70]. Here, GNNs are considered for training with a form of supervision, but the existence of generative GNNs suggest that unsupervised neighborhood of range cells encoding is already accessible. Key deep learning operations, such as convolutions and attention, have been adapted to graphs [188, 101, 83, 65, 41]. Among these adaptations, some directly relate to the graph Laplacian eigendecomposition and are not spatially localized over the graph. Less common edge-focused processing is also explored by the recent literature [27, 76]. The association of a convolutional neural network to produce signals embeddings with a GNN over which embeddings are placed was explored for modulation recognition in [119]. On another note, the combination of a spatial graph with a temporal convolutional processing of signal was applied to electroencephalogram (EEG) signals in [115].

The ability to process information organized according to a graph is critical to enable deep learning to tackle problems based on non-Euclidean data. Graph neural networks are part of the modern deep learning tools gathered under the name of geometric deep learning [31]. Geometric deep learning (GDL) achieves deep learning using representations and parameters constrained to graphs and differentiable manifolds. Here only the graphs aspect of GDL will be discussed, but a different application belonging to GDL involving the SPD matrices manifold is discussed in 3.2. Geometric deep learning

follows the refinement trend of deep learning approaches to adapt the vanilla framework brought by backpropagation, activation functions, dense and convolutional layers to data different from standard images and features vectors.

The point of using a graph in our application case is that the latter enables to enforce a constraint over how the information distributed over the neighborhood of range cells will be taken into account in a final encoding. The connectivity pattern enshrined in the adjacency matrix rules the relative flow of already encoded range cell information, while the importance weights among these information flows are determined in training. The graph convolutional layer (GCL) used in our experiments is the one proposed by [101] which transforms H input nodes defined by features vectors in $\mathbb{R}^M$ and organized in a graph as follows:

$$H = \widehat{D}^{-\frac{1}{2}}\widehat{A}\widehat{D}^{-\frac{1}{2}}XW \tag{2.48}$$

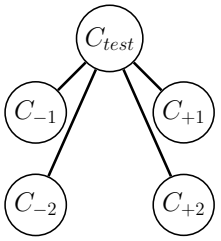
where:

- $X \in \mathbb{R}^{H\times M}$ is the nodes features matrix in which each of the H rows is described by a vector of M real-valued coefficients;
- $\widehat{A} \in \mathbb{R}^{H\times H}$ is the graph adjacency matrix with inserted self-loops (see Figs. 2.7, 2.9 and 2.5);
- $\widehat{D} \in \mathbb{R}^{H\times H}$ is the diagonal degree matrix of the graph with inserted self-loops (see Figs. 2.7, 2.9 and 2.5);
- $W \in \mathbb{R}^{M\times M'}$ is the trainable weights matrix enforcing the output nodes features dimensions M';
- $H \in \mathbb{R}^{H\times M'}$ is the output nodes features matrix, and the input of the following graph convolutional layer if several layers are stacked to form the GNN.

This GCL is spatially localized in opposition to, for instance, the convolution of [41]. It is followed by the application of an activation function like the ReLU defined in Eq. (2.7). To completely define a neighborhood graph one needs both the nodes features matrix X and the adjacency matrix A. The degree matrix D and $\widehat{D}$ are inferred from the adjacency matrix A and $\widehat{A} = A + I_H$ respectively, where I_H is the $H \times H$ identity matrix. The self-loops added in the graph thanks to the addition of I_H to A allows the convolution to take into account a node itself when computing its next representation according to its one step neighbors. The two occurrences $\widehat{D}^{-\frac{1}{2}}$ act as a normalization critical for gradient stability, and are part of the *renormalization trick* proposed by [101]. If several layers are chained one after the other, the input nodes features matrix X is replaced by the output nodes features representation H of the previous layers. Here, real-valued GNNs are considered since the transition from complex-valued representations to real-valued representations is already achieved during the single range cells encoding described in 2.3 (see Fig. 2.4). To produce the fixed size real-valued vector in $\mathbb{R}^Q$ with graph convolutions, one can either process the neighborhood according to a node2vec or a graph2vec framework[7].

Opting for a node2vec approach amounts to taking the central range cell vector representation, which belongs to $\mathbb{R}^Q$, as the output neighborhood representation. This node features vector takes into account local information thanks to the graph convolutional layer. However, since the GCL of Eq. (2.48) operates over a one-step neighborhood, its receptive field depends on the number of GCLs stacked before it. For instance, the

[7] "2vec" is a popular suffix in the deep learning community to indicate an architecture that outputs a vector representation of a form of data. For instance, among the works we have cite there are two wav2vec [19, 163] architectures encoding raw audio.



$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

A $\qquad$ $\widehat{D}_{ii} = \sum_{j=1}^{H} \widehat{A}_{ij}$

Figure 2.5: **Left:** Example neighborhood graphs without weights on edges. If all weights are equal to one as it is the case here, the relative ranges of the neighborhood are lost, *i.e.* information is lost using this graph. Since every cell is a single step away from the one carrying the actual detection, one convolutional layer as defined by Eq. (2.48) is enough for the whole neighborhood to impact the output representation of the central node. **Center:** Adjacency matrix A of the neighborhood graphs, without the inserted self-loops necessary to compute the graph convolutional layer proposed in [101] and defined in Eq. (2.48). **Right:** Degree matrix $\widehat{D}$ of the neighborhood graph with the inserted self-loops necessary to compute the graph convolutional layer.

neighborhood graph *(a)* of Fig. 2.6 requires two GCL stacked one after the other in order for the whole neighborhood information to be taken into account in the C_{test} range cell features vector. On the contrary, the neighborhood graph proposed on Fig. 2.5 would only require one GCL to achieve the same goal. Now that the stacking of GCLs is mentioned, it is necessary to mention the over-smoothing problem of GNNs. Stacking too many layers in GNNs may lead to nodes becoming hardly separable, eventually rendering an GNN architecture void of discriminatory power [45, 136]. Here, node2vec amounts to a so-called cell2vec since one node defines one radar range cell.

Opting for a graph2vec approach can be achieved through a similar application of graph convolutions, although this time stacking a minimal number of GCL is not essential for the final representation vector to capture data stemming from every node. A GNN can indeed be made into a graph2vec architecture by means of a final global pooling over the graph nodes, for instance through the computation of a *mean*. Such a global pooling discards the semantic information associated with the graph structure, although the latter could have been extracted by the preceding layers [48]. The intervention of the graph2vec encoding mechanism is illustrated on Fig. 2.4, where it is interchangeable with the previously detailed node2vec technique. The problem of range cells neighborhood graph encoding can seem very similar to the task of encoding molecules, which also happen to define small graphs where nodes are connected through bonds of varying nature. The molecules representation learning success [48, 202, 65] in the literature remarkably emphasizes how small graphs of varying size can be effectively transformed in to fixed-size vectors, suggesting augmented relevance for radar range cells neighborhoods. The varying nature of the molecular bonds can provide an interesting perspective with respect to the sampling diversity separating the graphs of our own application case. In [48], the embedding space is populated with graphs prototypes which reminds of the dictionary learning literature and the latent memory developed for an AE in [75]. This could inspire future orientations of the proposed hit2vec processing.

The neighborhood of range cells graphs presented on 2.6 are limited by the fact that all edges share the same weight e. This can seem surprising in the context of radar range cells since the latter can already span over large areas individually, as a direct consequence of the constrained bandwidth, as implied by Eq. (1.9). This suggests that it should be possible to attach relative importance to range cells within a neighborhood.

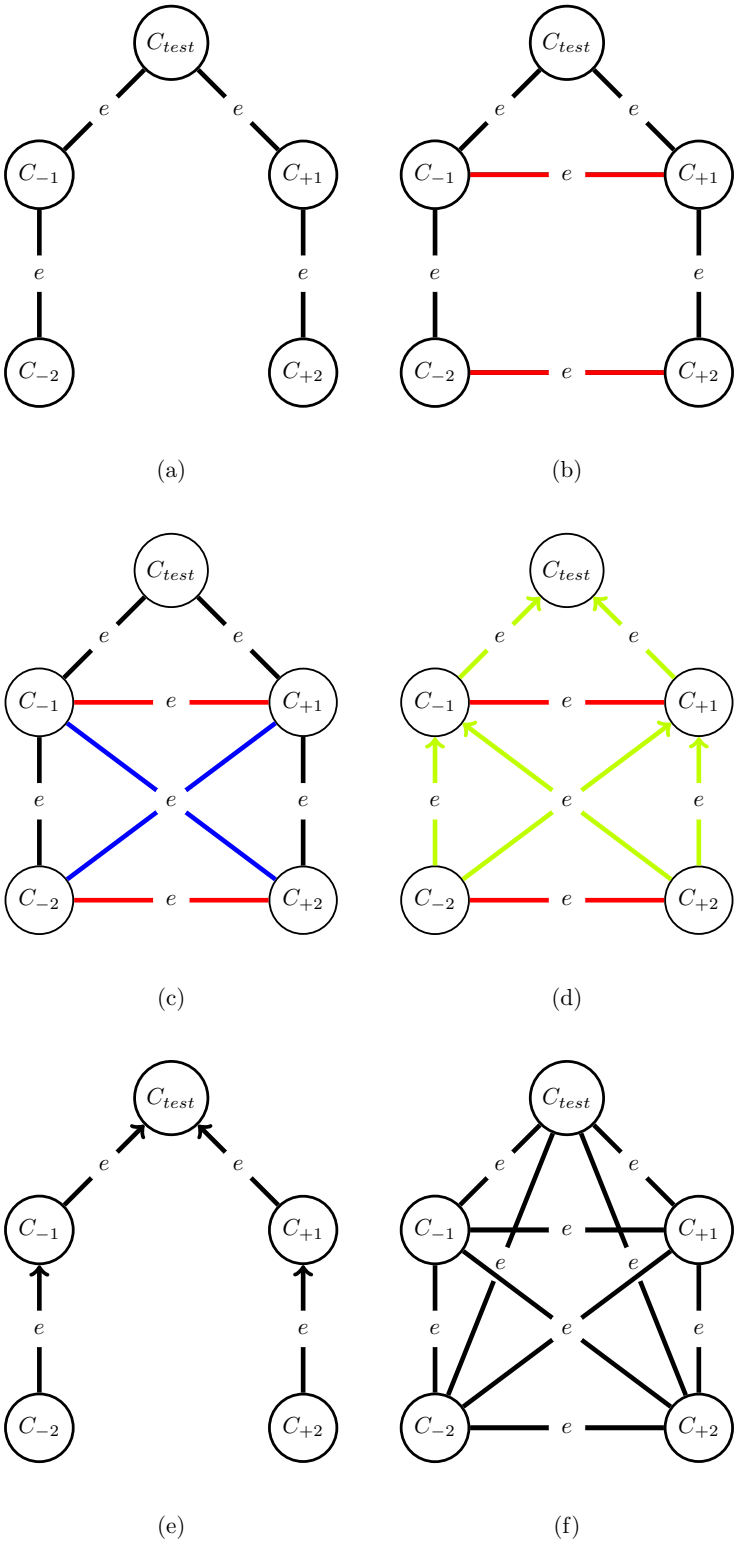


Figure 2.6: Neighborhood of $H = 5$ range cells proposed graphs: a diversity of edges configurations are considered to generate a relevant embedding for the range cell under test. Among the configurations illustrated, one can notice *(a)* that is close to time-series processing, the directed graphs {*(d),(e)* } which reveal what cell we are actually interested in learning a representation for, and the complete graph *(e)* that lets the learning phase chooses how to take into account the neighboring cells. The corresponding adjacency and degree matrices are shown on Fig. 2.7.

$$\begin{pmatrix} 0 & e & 0 & 0 & 0 \\ e & 0 & e & 0 & 0 \\ 0 & e & 0 & e & 0 \\ 0 & 0 & e & 0 & e \\ 0 & 0 & 0 & e & 0 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

(a)

$$\begin{pmatrix} 0 & e & 0 & 0 & e \\ e & 0 & e & e & 0 \\ 0 & e & 0 & e & 0 \\ 0 & e & e & 0 & e \\ e & 0 & 0 & e & 0 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

(b)

$$\begin{pmatrix} 0 & e & 0 & e & e \\ e & 0 & e & e & e \\ 0 & e & 0 & e & 0 \\ e & e & e & 0 & e \\ e & e & 0 & e & 0 \end{pmatrix} \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{pmatrix}$$

(c)

$$\begin{pmatrix} 0 & e & 0 & e & e \\ 0 & 0 & e & e & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & e & e & 0 & 0 \\ e & e & 0 & e & 0 \end{pmatrix} \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{pmatrix}$$

(d)

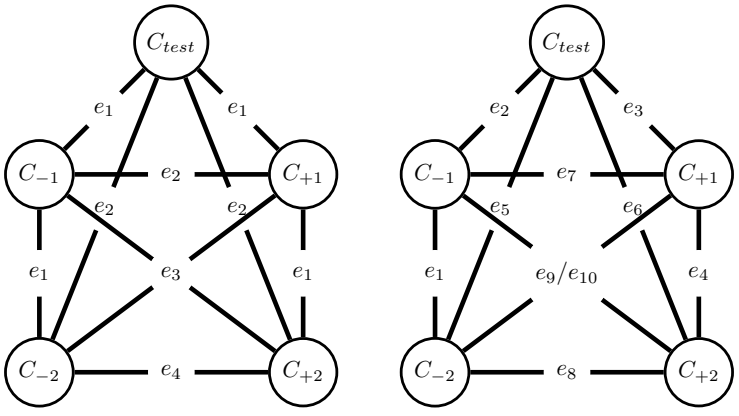
$$\begin{pmatrix} 0 & e & 0 & 0 & 0 \\ 0 & 0 & e & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e & 0 & 0 \\ 0 & 0 & 0 & e & 0 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

(e)

$$\begin{pmatrix} 0 & e & e & e & e \\ e & 0 & e & e & e \\ e & e & 0 & e & e \\ e & e & e & 0 & e \\ e & e & e & e & 0 \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

(f)

Figure 2.7: Neighborhood of $H = 5$ range cells proposed adjacency and degree matrices: these are the adjacency and degree matrices associated with the neighborhood graphs proposed on Fig. 2.6. As for Fig. 2.5, the degree matrix takes into account the inserted self-loops necessary to compute the graph convolutional layer. One can notice that only the undirected graphs translate into symmetric adjacency matrices.



(a) (b)

Figure 2.8: Alternative fully-connected neighborhood of $H = 5$ range cells proposed graphs, this time with different possible edge weights: For *(a)* we propose to associate each link between two range cells with a weight representing the distance in amount of range cells, while for *(b)* we propose a unique edge weight for each edge in the graph. The corresponding adjacency and degree matrices are shown on Fig. 2.9. The edge weights could for instance integrate the input signals sampling parameters creating the input diversity and one of the main difficulties of the representation learning task addressed.

$$\begin{pmatrix} 0 & e_1 & e_2 & e_3 & e_4 \\ e_1 & 0 & e_1 & e_2 & e_3 \\ e_2 & e_1 & 0 & e_1 & e_2 \\ e_3 & e_2 & e_1 & 0 & e_1 \\ e_4 & e_3 & e_2 & e_1 & 0 \end{pmatrix} \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{pmatrix} \quad \begin{pmatrix} 0 & e_1 & e_5 & e_{10} & e_8 \\ e_1 & 0 & e_2 & e_7 & e_9 \\ e_5 & e_2 & 0 & e_3 & e_6 \\ e_{10} & e_7 & e_3 & 0 & e_4 \\ e_8 & e_9 & e_6 & e_4 & 0 \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

(a) (b)

Figure 2.9: Alternative fully-connected neighborhood of range cells proposed adjacency and degree matrices: these are the adjacency and degree matrices associated with the alternative neighborhood graphs proposed on Fig. 2.8. As for Fig. 2.5, the degree matrix takes into account the inserted self-loops necessary to compute the graph convolutional layer. For *(a)* we end up with an adjacency matrix that associates each link between two range cells with a weight representing the distance in amount of range cells, while for *(b)* we get a unique edge weight for each edge in the graph, each weight appearing twice in the adjacency matrix since the graph is undirected. One can notice that both undirected graphs translate into symmetric adjacency matrices.

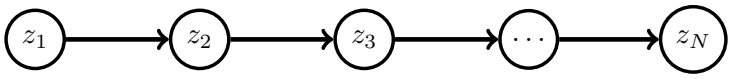


Figure 2.10: Illustration of the close relationship between processing data with a graph neural network and a recurrent neural network. A signal z sampled over N points can be seen as a directed graph with N nodes and $N-1$ edges.

The GNN neighborhood encoding is interestingly versatile since it could be combined to SPD manifold-aware processing to take advantage of the AR representation of the input signals put forward in section 2.3.1. For instance, each autocorrelation matrix computed over one range cell signal could be fed to a manifold-aware neural network to produce a fixed-sized real-valued vector. These vectors could then be similarly distributed over the neighborhood graphs proposed here. Graph neural networks are closely related to the recurrent neural networks put forward to encode a single range cell signal since a recurrent neural network simply follows a directed graph translating the chronological order of signal samples. For a signal z sampled over N points, the directed graph is illustrated on Fig. 2.10. The corresponding adjacency matrix is:

$$A_{RNN} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \tag{2.49}$$

However, GNNs are much more generic and versatile than RNNs since they allow to enforce arbitrary symmetry and weighting schemes over the range cells defining the neighborhood graph. Such weights can be applied to nodes but also to edges features. While the upstream encoding methods proposed in 2.3 do not explicitly take into account the varying input signals sampling frequency (the PRF), this information could be added to the edge features to intervene in the features flow of the graph during training. Taking the PRF explicitly into account seems relevant since this information is leveraged in the resampling baseline previously mentioned. Taking into account so-called features of

features, or input data hyperparameters, within the architecture was already done in the literature. For instance, [70] recorded the nodes locations in a graph pooling layer to subsequently place nodes back to their position in the input graph while unpooling. This was done in order to define a generative graph UNet architecture. That being said, the varying input signal length is already available in our input signals as a data dimension, but not necessarily explicitly valued by the proposed range cell encoding of 2.3. As is, the example GCL defined by Eq. 2.48 could only take into account scalar edge weights integrated within the adjacency matrix, as the examples of Fig. 2.5, Fig. 2.7 and Fig. 2.9 show. Other GNNs architectures however address this limitation [76]. Graph neural networks can be said to be a generalization of RNNs to deal with cyclic, directed and undirected graphs [188, 161]. Constraints over a range cells neighborhood graph could for instance contribute to define and enforce a concept similar to the one of guard cells used for radar CFAR detection [18, 15]. In such a case, a graph could maintain constrained edge weights in order to take into account guard, test and reference cells in framework smoother than simply choosing to take into account a cell features or not to determine a local clutter or targets map.

2.5 Single range cell encoding experiments

Since the experiments evaluate the separability of targets and number of pulses classes using one-class classification approaches and metrics, the experiments encoding neighborhoods of range cells are presented in chapter 4. Here, we will only report preliminary results regarding the encoding of single range cells. The experimental results presented here stem from only one of the single range cell encoding approaches detailed in this chapter, as it is the only approach that showed encouraging results.

2.5.1 Experiments protocol and data

The dataset used to evaluate the representation learning over range cell IQ signals is a slightly modified version of the publicly available [8] simulated PDR dataset used in [26]. Whereas in [26] the discrimination task was conducted at the scale of several bursts of pulses to define a Doppler signature containing the evolution of a spectrum over time, here we work at the scale of a radar hit, *i.e.* at the scale of a single burst. This is equivalent to a single row of the Doppler signatures depict on Fig. 1.6 and Fig. 3.8. Such a row is defined by the DFT computed over the burst pulses backscatter. Looking at the previous figures, one can realize that the reduced information available in a single burst is can translate into an unlucky combo where the burst describes an uncharacteristic node in the modulation pattern of a target, assuming the target creates one such pattern in the first place. This should be kept in mind as it could explain a small amount of encoding or discrimination failures.

The dataset defines four classes of helicopter-like targets, each of the latter defining a Doppler signature modulation of specific complexity as illustrated on Fig. 3.8. This varying modulation pattern is due to the different numbers of blades characterizing each class. The main difference between the targets representation here and the one used in [26], other than the single burst scale of the data points, is the varying number of pulses available in the radar bursts. The effect of such a variation on the resolution of the targets I/Q response has already been illustrated on Fig. 1.6. The robustness of our single range cell encoding with respect to such an input resolution variation is one of the pursued invariances as summarized in table 2.1. The four kinds of modulation patterns define the so-called *targets classes* in our experiments, in opposition to the

[8] `https://github.com/Blupblupblup/Doppler-Signatures-Generation` Accessed: 28/10/2022

so-called *pulses classes* whose labels refer to the number of pulses describing the target in the input representation. Again, this number of pulses translates into the size of the complex-valued input matrix $Z_{I/Q}$ of Eq. (1.2) and the resolution at which the target can be described.

The model used to encode the data is the FCN whose architecture is described in table 2.2. The aim of the learning phase is to separate the targets classes and to verify how strongly the neural network output representations are distributed according to the number of pulses defining the input representations. In order to create the dataset, the MATLAB [125] simulation creates numerous series of bursts of pulses to describe targets individually. Several targets are generated to populate each class. The diversity among targets is generated through the variation of the the rotor rotation speed, the initial position, the blades length, and the target velocity. The bursts themselves are then individually distributed in the training, validation and test sets for our experiments. The creation of bursts in series to describe a moving target helps avoid the unlikely generation of bursts batckscattering only or mostly modulation patterns at their modulation node, which hides the discriminative modulation pattern we are relying on.

This approach aims at exploring the ability of the considered architectures to learn the sought after invariances and is flawed in the sense that it suffers from a data leakage potential. Data leakage here amounts to the presence of bursts backscatter describing the same target in more than one of the three divisions (training, validation, testing) of the complete dataset. This deficiency may allow the encoding neural network to learn based on targets features irrelevant to the actual objective and create misleading performances. A typical example case of the danger of data leakage was illustrated by the successive versions of an X-rays discrimination study [145][9] where each patient produced several input representations, and patient overlap corrupted the (training, validation, testing) data division. This led to deceptive performances where the neural network could cheat with the features identifying patients instead of their pathology. The risk of data leakage is set aside to enable a more accessible preliminary study of the proposed encoding scheme, and should be reduced by the lack of features identifying individual targets across bursts anyway. Indeed, there is nothing else than the modulation pattern identifying the class in the inputs. In a more realistic setup the danger of data leakage remains, since individual targets could be identified across the three data divisions by a situational clutter spilling over the single burst signatures.

2.5.2 Preliminary results with supervised representation learning

In our preliminary experiments, the only promising performances were stemming from the FCN architecture for which supervision was available, *i.e.* where the training setup described by Eq. 2.44 was used. The unsuccessful RNN, seq2seq and FCAE architectures are still briefly described in Appendix C. To evaluate the evolution of the separability of targets and pulses classes in the single range cell embedding space, we record the AUCs of the one-class classification (OCC) of each of these classes during training, the minority monitored class defining the positive class for the AUC computation. The AUCs are computed over the test data made of separate bursts. An example run yields the metrics indicated on Fig. 2.11, where the loss per batch during training complements the AUCs evolutions. On these metrics, one can observe the encouraging rise of the targets classes AUCs while the pulses classes AUCs remain aroung the randomness performance of 0.5. The OCC methods selected to produce the successive AUC scores are the widespread, shallow learning isolation forest (IF) [118] and one-class support vector machine (OC-

[9] First version with data leakage: `https://arxiv.org/abs/1711.05225v1` Third version without data leakage: `https://arxiv.org/abs/1711.05225v3` Accessed: 25/11/2022

SVM) [164], both being defined in chapter 3. All targets classes seem to benefit similarly from the encoding network training, and the performances of both OCC methods remain close during training.

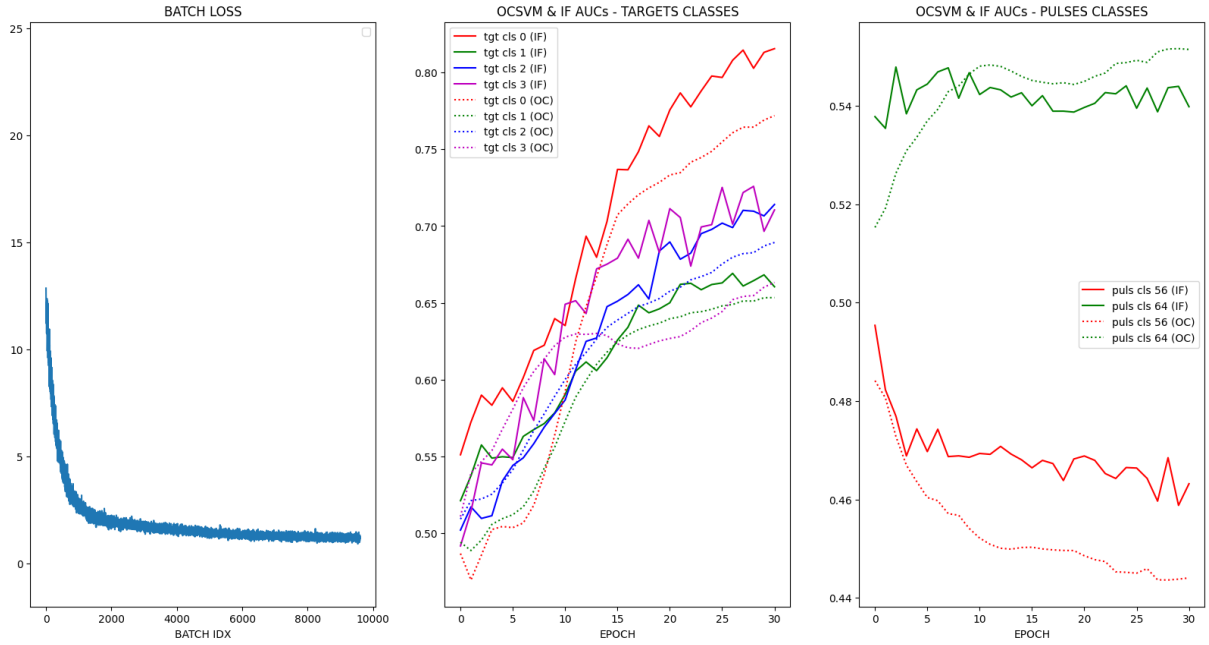


Figure 2.11: Learning metrics of the cell2vec FCN architecture training. The batches loss rapidly decrease and converge, outlining a common loss trajectory in deep learning experiments. The targets classes AUCs rise during training, indicating a continuously improving separability of the targets classes in the embedding space. On the contrary and as hoped, the pulses classes separability is not favored since the associated AUCs remain around 0.5, which is the random discrimination performance. The improving separability of targets classes appears when using specialized machine learning methods like IF and OC-SVM, and is not visible when replacing the latter with a simple Euclidean distance to a class reference point, or with a Silhouette clustering score [150] computed with a Euclidean metric.

To complement the evolution of the losses and test dataset AUCs during training, one can compare the latent distribution of the classes before and after training. An example of one such comparison if proposed with Fig. 2.12 and Fig. 2.13, where one can observe how the targets classes appear partially disentangled after training on the TSNE visualization (see top left image on both figures). The 2D visualization of such distribution is made possible by the dimensionality reduction methods TSNE [182] and PCA. The OCC methods IF and OC-SVM, as well as the dimensionality reduction and the AUC scores computation are all implemented using Scikit-learn [139].

2.5.3 Necessary follow-up experiments

The previous experimental results can only be taken as a proof-of-concept that aims at demonstrating the feasibility of encoding a semantic diversity of single range cells in order to separate them based on their individual Doppler content. On the one hand, in terms of deep learning experiments the results presented here lack statistical significance due to the fact that a single experimental run, *i.e.* a single random seed, is put forward. Ideally, the following improvements should be made to ensure a fair and relevant evaluation of the proposed encoding methods:

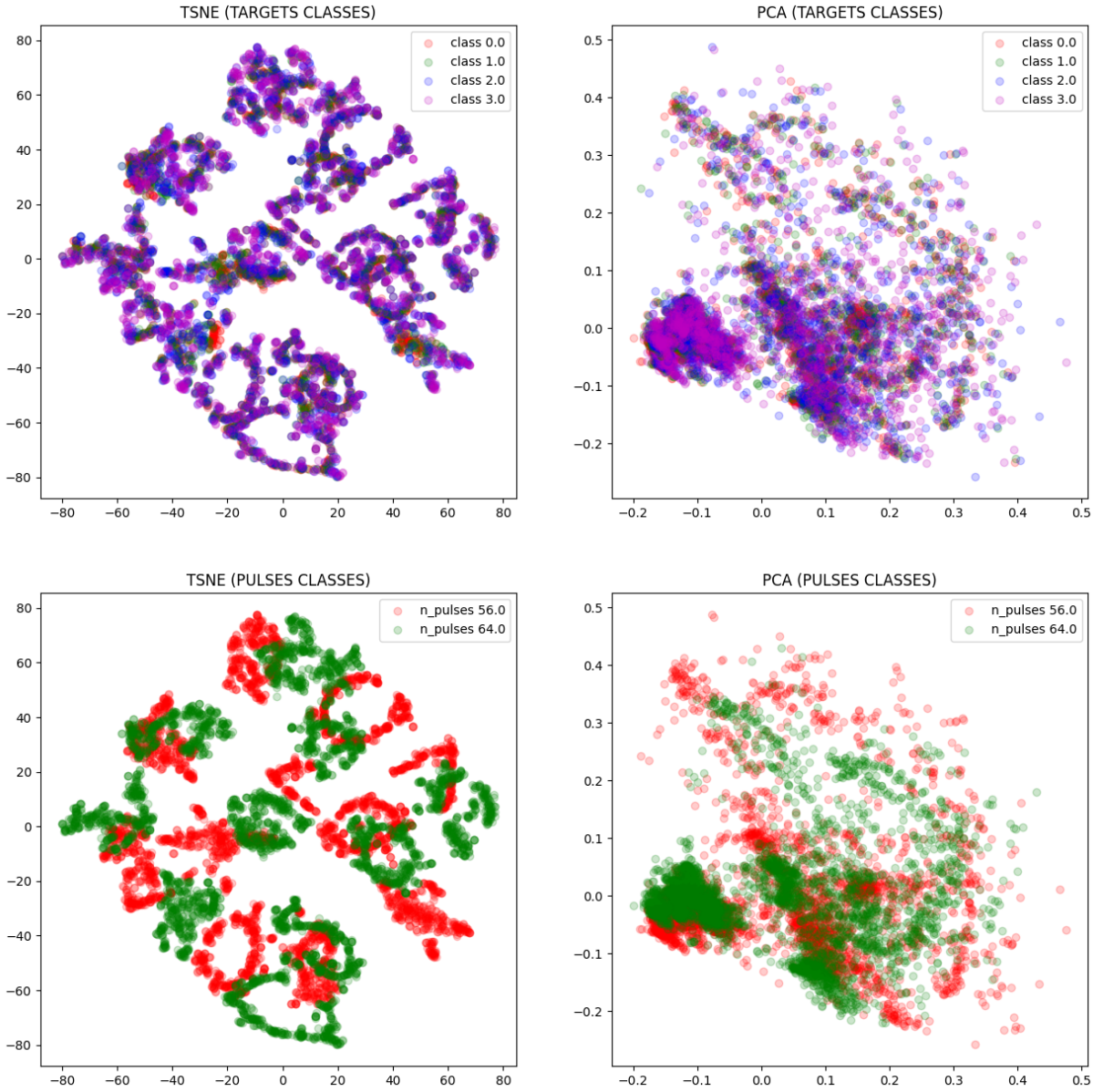


Figure 2.12: 2D visualization of the individual range cell embeddings distribution produced by an FCN as cell2vec, before training. **Top** - each color depicts one target class, i.e. one Doppler pattern. **Bottom** - each color depicts one pulse class, i.e. one Doppler resolution class.

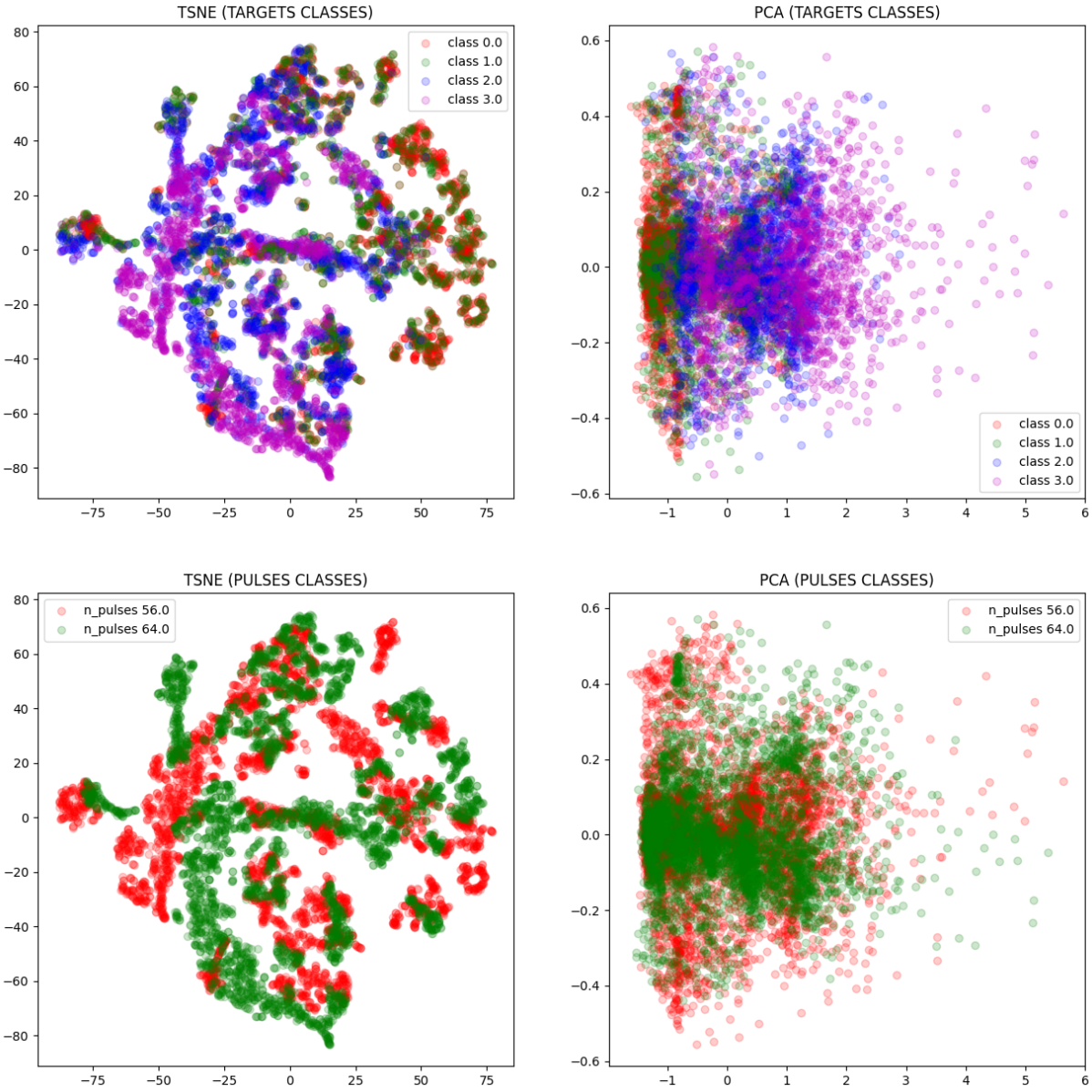


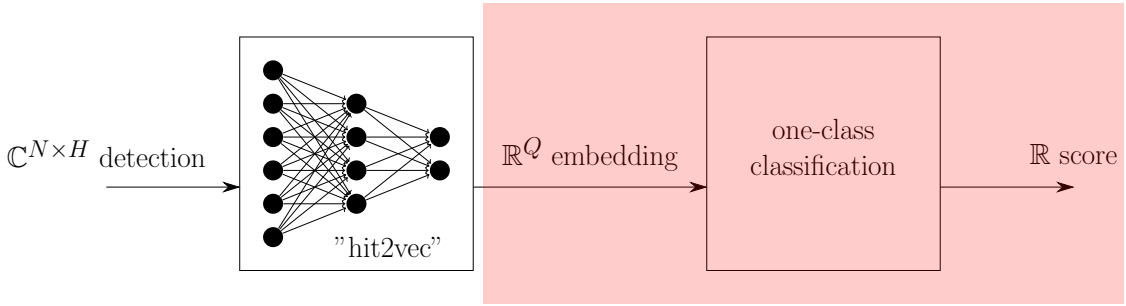
Figure 2.13: 2D visualization of the individual range cell embeddings distribution produced by an FCN as cell2vec, after training. **Top** - each color depicts one target class, i.e. one Doppler pattern. **Bottom** - each color depicts one pulse class, i.e. one Doppler resolution class.

- Evaluate mean and variance of the AUC metric over a dozen of random seeds to take into account the benefits and drawbacks of random initialization.
- Compare the proposed approaches with a simple baseline. Here the latter should be fabricated since the use case is very specific. In a very permissive way, one could argue that the unsuccessful approaches we proposed but which did not yield encouraging performances and thus do not appear in the results here constitute a form of baseline.
- Remove the targets overlap among the three (training, validation, testing) dataset divisions to suppress the data leakage risk.
- Enrich the targets dataset with single burst targets backscatter of more variable SNR and including progressively confusing clutter.
- Add targets from an unknown target class in the testing set to see if the encoding framework remains somewhat discriminative for this unseen class, since this is the semi-supervised encoding we are actually seeking.
- Add targets of known and unknown targets classes (cf. previous point) represented by input signals defined by a number of pulses different from the ones seen in training to see if the encoding remains somewhat discriminative for this unseen input signal format, since this is the semi-supervised encoding we are actually seeking.
- Add targets from known and unknown classes (cf. previous point) represented by input signals with a sampling PRF unseen in the training set to see if the encoding remains somewhat discriminative for this unseen input signal format, since this is the semi-supervised encoding we are actually seeking.

As a reference, rigorously conducted comparison of machine learning methods can be seen in chapter 3. The two last improvement points proposed here are motivated by the ideal invariances the encoding neural network should manifest. These invariances are summarized in table 2.1.

Chapter 3

One-class classification for radar targets discrimination



Chapter 3

This chapter contains contributions of this thesis in sections 3.1.3, 3.1.5 and 3.2.3. These contributions are associated with our three publications [26, 25, 24]. The previous chapter introduced approaches to encode the enriched I/Q sweep feature of radar hits, *i.e.* to go from $\mathbb{C}^{N\times H}$ to $\mathbb{R}^Q$. Once detections are encoded, one can implement a discrimination method to help the radar operator isolate relevant targets from clutter and low-priority objects. Since encoding leads to the projection of hits into the shared representation space $\mathbb{R}^Q$, any discrimination method applicable to vector representations could be used. This opens the door to radar hits discrimination with any of the numerous deep and non-deep, unsupervised, semi-supervised and supervised classification approaches available in the literature. Ideally, such discrimination would be handled by a supervised classification or an open-set recognition (OSR) pipeline with specific enough targets classes with respect to military and civilian activities.

Such a supervised classification approach would require excessive supervision, *i.e.* would call for large, diverse and completely labeled datasets to be available for training. Such dataset is unthinkable in military radar applications, since not many labeled samples are accessible for friendly targets, while unfriendly targets can remain completely unknown to the sensor. An OSR pipeline supposedly identifies a closed set of classes seen during training just like supervised classification, but also whether a test sample belongs to one of the known classes at all. This entails OSR demands as much supervision as supervised classification regarding the closed set of classes encountered during training, making it equally unsuitable for the radar targets discrimination pursued. This emphasizes the similarity between OSR and classification with rejection [124, 23, 176].

One can still note that a high-performing open-set recognition, identifying well referenced classes during training, coupled with a clustering of samples belonging to poten-

tially unindentified but consistent data modes could perfectly answer the challenges of a radar deployment. Indeed, targets perception varies a lot due to radar target-aspect sensitivity, and both the geography surrounding the sensor and the weather influence the values being processed. Thus, setting up a radar amounts to update the clutter and relevant targets appearances and how to separate the latter, making the adaptive nature of open-set recognition approaches and clustering notably adapted when they are combined.

To tackle the challenge of low supervision, we chose to discriminate between encoded radar hits using OCC. The intuition of the favored OCC method is to use the few labeled data points available during training to gather the output representations of a neural network around one or several latent reference points. The distance to the reference points in the output space can then be used to generate an outlyingness score $\mathcal{O}(x) : \mathbb{R}^d \to \mathbb{R}$ for a test sample $x \in \mathbb{R}^d$. If we then consider the latent reference points as capturing the distribution of a set of radar targets classes, one can conclude that this score directly translates into a mean to discriminate between targets belonging to the latter set of targets classes and targets that do not. Details regarding this favored OCC technique and other OCC methods will be provided in the upcoming sections.

Assuming a performing enough OCC, one could separate a set of radar targets described by a limited quantity of labeled samples from other detected objects and phenomena. The advantage of this approach is that it does not require labeled samples for all the classes being separated, which gives an answer to the lack of supervision in the task at hand. The concentration of latent outputs belonging to the set of targets classes offers an intuitive way of including labeled samples outside of the one-class during training by repelling their output representation from the reference points. In the proposed one-class classification setup, it is worth noting that the one-class could potentially contain a certain diversity of targets, *i.e.* not be limited to a single kind of radar targets. This last point is critical with respect to the targets discrimination task considered, since an operator would likely want to set up an alarm [1] for arbitrary sets of targets, however diverse.

This arbitrary one-class diversity is yet another key challenge of the ideal radar targets discrimination. Semantically close data modes, *i.e.* samples so close that their separation can be difficult, can be found both inside and outside of the one-class. This question of semantic proximity leads to the definition of near and far OODD. Near out-of-distribution detection (OODD) aims at distinguishing one or several data classes from semantically similar data points. For instance, identifying samples from one class of CIFAR10[2] among samples of the other classes of the same dataset solves a near OODD task. On the other hand, separating CIFAR10 samples from MNIST[3] samples is a far OODD task: there is no strong semantic proximity between the data points being separated [146]. The concept of OCC for radar targets discrimination is illustrated on Fig. 3.1, which depicts how diverse targets can be gathered within a one-class boundary, and how labeled out-of-distribution samples can contribute to the boundary. This additional supervision translates into semi-supervised AD. In some cases and with expert knowledge involved, it is possible to generate artificial labeled anomalies to contribute to the training phase, defining a form of self-supervised learning[4]. This idea could also be interpreted as data augmentation[5], although it may involve creating a label initially

[1] A threshold applied to the OCC score.

[2] CIFAR10 is a popular baseline dataset of tiny images proposed in [105].

[3] MNIST is a popular baseline dataset of handwritten digits proposed in [109].

[4] Self-supervised learning (SSL) is any learning enabled by artificially generated supervision. This can typically be achieved by transforming unlabeled data and associating one label to each transformation, thus unlocking supervision.

[5] Data augmentation is the process of enriching a dataset thanks to transformations of existing data

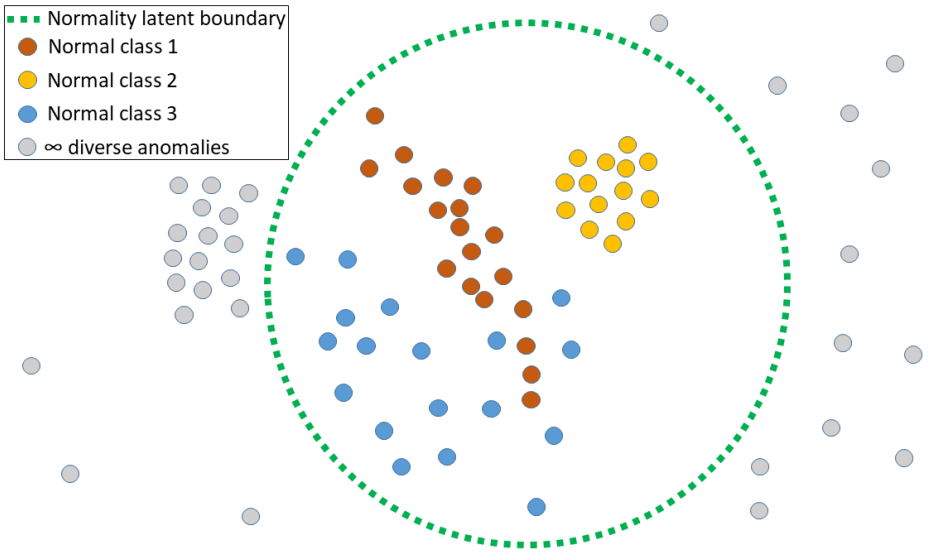


Figure 3.1: One-class classification intuition diagram, which can also be understood as anomaly detection: the "normal" one-class distribution has to be captured in order for the detection of out-of-distribution samples to be possible. The one-class ideally can be composed of several classes, and the anomalies or out-of-distribution samples are of infinite diversity. The latter can be other data classes, and noisy samples. In the case of radar targets discrimination, the one-class may for instance gather a diversity of small and slow targets for which labeled reference samples are available for training, and for which an alarm would be raised to warn the radar operator. In such a context, negative labeled samples available during training to refine the one-class boundaries could stem from weather phenomena which are known to appear close to relevant small and slow targets. The use of a minority of unrepresentative labeled anomalies during training for additional supervision will be addressed by some of the OCC methods presented in this chapter.

absent from the learning setup. Our application of OCC to radar targets discrimination can be considered as a near OODD task, since the OCC is meant to separate valid radar targets stemming from a unique sensor.

In addition to the presentation of OCC methods, this chapter will present OCC experiments conducted independently from any hit encoding in section 3.3.

3.1 One-class classification methods considered

This section will put forward several anomaly detection methods we have considered for encoded hits discrimination in this work. One of these methods, deep random projection outlyingness (RPO), is one of our original contributions in this thesis. This contribution is detailed in 3.1.3 and is an evolution of the non-deep RPO presented in 3.1.1 beforehand. Experiments were first conducted on generic datasets, *i.e* MNIST, Fashion-MNIST and CIFAR10, and on simulated and real radar data, all different from encoded hits. These experiments on data types of a different nature than the hits we are considering remain relevant for our AD experiments, since encoded hits are vectors of features, and given a comparable level of supervision discriminating encoded hits should not be different from discriminating the images or 1D range profiles we will consider. In

points. For instance, one can add noise to existing data points to make the model training more robust.

all cases however, the difficulty of the task relates to the relative proximity of the data modes separated. This emphasizes the relevance of working on a filter with two independent steps: one can improve and plug various discrimination approaches on the output of the radar targets encoding, and vice versa. An important feature of the methods presented is that they all produce a continuous scalar decision score that would allow the radar operator to have a refined appreciation of the anomalous nature of a target, in contrast with a less expressive binary result.

3.1.1 Shallow and deep one-class classification baselines in the literature

The widespread use of deep learning for data discrimination, including OCC, being recent and calling for specific data and computational needs, it seemed necessary to include a diversity of non-deep learning discrimination methods in our study to end up with a useful perspective on which discrimination to prefer to sort out encoded hits. We begin by considering a classic outlier detection measure, the Mahalanobis distance (MD) [123], which computes a distance between a multivariate distribution sampled over n samples in d dimensions $X \in \mathbb{R}^{d \times n}$ and a data point x in the same representation space. This distance is defined by the following expression:

$$\mathcal{O}_{MD}(x; X) = \sqrt{(x - \mu_X)^T \Sigma_X^{-1} (x - \mu_X)} \tag{3.1}$$

where Σ_X denotes the sample covariance matrix $\Sigma_X = \frac{1}{n} X_c X_c^T$ with X_c the centered data matrix, *i.e.* the sample mean $\mu_X = \frac{1}{n} \sum_{i=1}^{n} x_i$ was subtracted from every data point in X to compute X_c. We can make two remarks regarding the MD: it is found in the exponential of the probability density function of a multivariate Gaussian distribution, and it uses dedicated spread and location estimators for everyone of the d dimensions. One of the downsides of the MD is the computation of a covariance matrix and its inverse. The estimation of a covariance matrix is problematic because for a sampled version to be relevant and well conditioned, it is necessary to estimate it with numerous samples and while respecting a relatively small $\frac{d}{n}$ ratio. In other words, the number of samples must be much larger than the number of dimensions in the representation space where the covariance matrix is estimated. Not respecting this and computing a Mahalanobis distance based on the ill-conditioned covariance matrix is ill-advised since the bad conditioning prevents the accurate computation of the covariance inverse required by Eq. (3.1) [187]. This leads us to another old non-deep outlyingness measure, called random projection outlyingness (RPO), that does not require a covariance matrix estimate and which we will be using in our experiments. RPO combines numerous normalized outlyingness measures over 1D projections with a *max* estimator in order to produce a unique and robust multivariate outlyingness measure, which translates into Eq. (3.2):

$$\mathcal{O}_{RPO}(x; p, X) = \max_{u \in \mathbb{U}} \frac{|u^T x - MED(u^T X)|}{MAD(u^T X)} \tag{3.2}$$

where x is again the data point we want to compute the outlyingness for, p the number of random projections (RP) u of unit norm gathered in the set $\mathbb{U}$, and X the training data matrix. MED stands for median, a location estimator, and MAD for median absolute deviation, a spread estimator. The *max* implies retaining only the worst outlyingness measure available among all the 1D projections, i.e. the worst normalized deviation from the projected median. In [186], the asymptotic equivalence between Eq. (3.2) for a large number of RPs and the MD of Eq. (3.1) (up to a constant factor) is established, with the motivation of obtaining an equivalent of the latter without computing a covariance

matrix[6]. This equivalence with the Mahalanobis distance indicates that RPO with enough RPs, after the *max* integration over RPs, describes a normality ellipsoid in the input space, i.e. the representation space of x. The RPO outlyingness actually leads to the definition of a statistical *depth*[7] approximation [62, 90], another quantity that orders data points from a given set from most to least normal. This stochastic approximation of a statistical depth is called random projection depth (RPD) and is defined by:

$$RPD(x; p, X) = \frac{1}{1 + \mathcal{O}_{RPO}(x; p, X)} \tag{3.3}$$

The exact statistical depth is computed with a *sup* replacing the *max* in Eq. 3.2, the computation of the *sup* implying the consideration of all possible random projections, *i.e.* an infinity of random projections. Both Eq. (3.2) and Eq. (3.3) can be said to combine 1D views of a multivariate distribution to produce a multivariate outlyingness measure based on normalized univariate distances. This intuition behind the use of 1D RPs to generate a multivariate outlyingness measure is illustrated on Fig. 3.2. A deep adaptation of RPO is proposed in section 3.1.3, and RPs with multiple output dimensions were also considered in our experiments. Harnessing RPs with multiple output dimensions however reintroduces a covariance matrix as a spread estimator in order to produce normalized distances. One can notice that the use of RPs to project data points is related to the generation of intermediate representations with neural networks layers whose weights are frozen right after their random initialization, something quite recently mentioned in the literature [69, 198, 74, 160].

The other common shallow AD methods chosen are one-class Support Vector Machine (OC-SVM) [164], Isolation Forest (IF) [118] and Local Outlier Factor (LOF) [30]. The first method is an extension to one-class classification of the now classic SVM classifiers [29]. OC-SVM projects data in a feature space where it will try to find a maximum margin hyperplane to separate data points from the feature space origin. This is achieved thanks to the following objective function:

$$\min_{w,\rho,\xi} \quad \frac{1}{2}||w||_F^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \tag{3.4}$$

ρ is the distance separating the origin from the hyperplane w, ξ_i are slack variables allowing boundary violation with penalization. $||w||_F$ regularizes the definition of the hyperplane w using the norm of the feature space F in which data points are projected by a kernel. The integer n is the number of data samples available for training and $\nu \in (0, 1]$ is an upper bound on the fraction of outliers during training and a lower bound on the fraction of support vectors for the hyperplane boundary. IF [118] uses recursive partitioning on subsets of data points in the feature space, and produces an anomaly score based on the ease with which each point is isolated from the rest in each subset. It works based on the assumption that anomalous samples are more susceptible to isolation in the feature space. One recursive partitioning isolation binary tree is built for each subset of data points, making IF an ensemble method. In the end, IF computes an anomaly score $\mathcal{O}_{IF}$ for each instance x whose expression is:

$$\mathcal{O}_{IF}(x; n, X) = 2^{-\frac{E(h(x))}{c(n)}} \tag{3.5}$$

n is, as for OC-SVM, the number of data points available for training and $c(n)$ the average path length of an isolation tree. The average path length intuitively translates into the average number of recursive splits needed to isolate a data point in the feature

[6]The convergence is proved when the number p of projections tends to infinity.

[7]A statistical depth provides a center-outward ordering of data points with respect to a dataset.

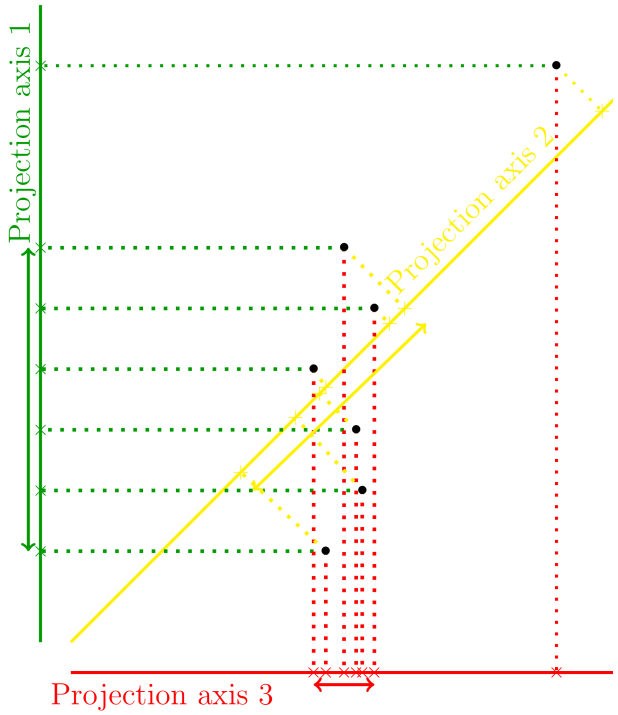


Figure 3.2: Illustration of the intuition behind the use of 1D random projections to compute a multivariate outlyingness measure. Once a set of 2D samples is projected, evaluating the normalized distance to the location estimator of each projection easily allows to detect the obvious outlier, the latter being positioned at greater distance from the location estimator on at least one random projection. One random projection is enough to raise the maximum seen in Eq. (3.2). This depicts that multivariate outlyingness can translate into multiple univariate outlyingnesses.

space. The vector x is the sample whose anomaly score we want to obtain, and $h(x)$ the associated path length. The previous equation uses $E(h(x))$, the average path length across the forest of isolation trees, normalized by $c(n)$, to obtain $\mathcal{O}_{IF}(x, n)$. IF is a particularly interesting shallow AD method since it is advertised as being able to provide good performances with a small subsample of data and few isolation trees.

The last common shallow AD method considered is LOF [30]. To compute LOF, we choose a certain number k of nearest neighbors to be considered for each data point. The local density attached to a point will be determined by how close its k nearest neighbors are. A point having a higher local density than its neighbors will be more likely to be an inlier, since this translates into belonging to a higher density part of the feature space. Thus, LOF assigns to each data point an outlier score based on the ratios of its own local density and the local densities of its k nearest neighbors.

Now that we proposed shallow baselines, let us describe our deep OCC baseline: the autoencoder (AE). The autoencoder is a generative neural network architecture which is commonly trained to reproduce its input and, thanks to constrained intermediate representations, provide lower dimensionality encodings. In order to do so one typically defines an undercomplete autoencoder [77, p.494], where the reduced representation capacity of the neural network compels it to select the most important information to encode the input before recreating it through some form of upsampling [77]. Assuming an autoencoder neural network Φ, the training loss is the reconstruction error $\|\Phi(x) - x\|$. A simple way to achieve AD with an AE is to use the reconstruction error of test samples as the AD score [199, 155]. Once the AE is trained to recreate samples belonging to the one-class of OCC, the reconstruction error of OOD samples can be expected to be higher. The outlyingness computed thanks to an AE is thus:

$$\mathcal{O}_{AE}(x; \Phi, X) = \|\Phi_X(x) - x\| \tag{3.6}$$

where $\Phi(x)$ is the reconstructed input, *i.e.* the output of the autoencoder. This AE anomaly detection intuition has already been applied to radar data in [40, 193]. The reconstruction score intuition also makes it easy to grasp the challenge of separating semantically similar inputs belonging to different OCC classes in the case of near OODD, since it is likely that an AE able to reconstruct one sample will also achieve a fair reconstruction on semantically similar ones.

Not only the AE is an interesting choice due to it being an OCC method based on a pretense reconstruction task unrelated to AD, but it also led to diverse variants designed for AD. A memory-augmented deep AE, called *MemAE*, was for instance proposed in [75], where the latent space is discretized through the definition of reference elements based on the training data. These latent prototypical representations are then combined with weights determined by an attention mechanism with a sparsity constraint to produce the latent representation provided to the decoder. This approach aims at limiting the generalization capacity of the AE, the latter potentially being able to reproduce some anomalies just as well as some normal samples after training. The sparsity constraint limiting the complexity of the latent dictionary entries combination ensures the reconstructed input can not escape the main patterns observed in the training data. Another extension of the initial intuition of using AE for AD worth mentioning is *ConAD*, which stands for *consistency-based anomaly detection*, which was put forward in [134]. This method utilizes a multi-headed decoder network to handle multi-modality in the input and latent spaces, and particularly to avoid the acceptance of reconstructions based on an artificial global mean mode in the latent space as belonging to the "normal" one-class. The shallow methods previously mentioned can also be associated with an AE to produce a hybrid one-class classification method where the AE embeds the inputs in the latent representation space of the generative architecture, *i.e.* the neural network

bottleneck, the embeddings being then used for AD through a shallow AD [159]. In such a case, the reconstruction error does not contribute to the outlyingness measure, and is only harnessed to train the generative neural network whose encoding part produces embeddings. The reconstruction-oriented parameters optimization amounts to a pretense task requiring no supervision since the input serves as output, and is thus close to the fundamental idea of self-supervision setups. This closeness to self-supervision also suggests transformations could be used to enrich the training set and reinforce the generality and relevance of the bottleneck representation in the AE.

3.1.2 Deep one-class classification with latent hyperspheres

Most of the OCC experiments and developments proposed by this thesis are inspired by [153] which proposed a deep AD method called Deep Support Vector Data Description inspired by the now decades old non-deep Support Vector Data Description (SVDD) method [175]. The SVDD approach estimates a hypersphere boundary around a training set made of samples x_i to allow for one-class classification based on whether the tested point lies within or outside of the boundary learned. To find this boundary the hypersphere volume is minimized while keeping as many training points within the volume as possible. This can be understood as the minimization of R^2 where R is the radius of the one-class hypersphere under the constraint:

$$\|x_i - c\|^2 \leq R^2, \forall i \tag{3.7}$$

During this boundary optimization the training points are all considered to belong to the so-called one-class. Slack variables $\xi_i \geq 0$ are introduced to make the boundary soft and lead to the minimization of the following quantity:

$$\min_{R,c,\xi} R^2 + C \sum_i \xi_i \tag{3.8}$$

while respecting the soft boundary constraint $\|x_i - c\|^2 \leq R^2 + \xi_i, \xi_i \geq 0$ retaining the training data points within the one-class boundary, where c is the hypersphere centroid and C is a weight to balance the volume with the boundary trespassing [175]. The use of kernels, as for Support Vector Machines, makes the boundary more adaptive through the usual implicit mapping trick. Since SVDD defines a hypersphere, the ideal mapping would enclose the data points within a spherical volume. Using SVDD can be equivalent to using the OC-SVM method described in 3.1.1 if one uses a Gaussian kernel [153, 175].

With a very close intuition, Deep SVDD [153] uses a neural network to learn representations of training samples in an output space where their Euclidean distance to a one-class reference point or centroid c defines the loss to be minimized during training. The training loss of Deep SVDD for a sample of size n (*i.e.* n data points) with a neural network Φ with weights W distributed over L layers is as follows:

$$\min_W \left[\frac{1}{n} \sum_{i=1}^{n} ||\Phi(x_i; W) - c||^2 + \frac{\lambda}{2} \sum_{l=1}^{L} ||W^l||^2 \right] \tag{3.9}$$

The second term is a weights regularization controlled by λ, and will also appear in Eq. (3.10), (3.13), (3.14) and (3.17). This l2-norm parameter regularization is called *weight decay*, ridge regression or Tikhonov regularization in the scientific literature [77]. In our experiments, the heuristic of defining c as the mean output representation of the training samples x_i before training any parameter in the encoding neural network sometimes appeared as relevant as the coordinates of one of the training samples in the output space also taken before training. It also seemed that a reasonable yet arbitrary

output coordinates choice could replace the heuristic, reasonable coordinates depending on the transformations implemented by the neural network in use.

A natural extension of Deep SVDD is the replacement of the single hypersphere boundary in the output representation space with a multitude of hyperspheres, which could be understood as one-class *atoms*[8]. This approach was proposed in [73]. Such a composite boundary could be better tailored to the output representations distribution. For this multisphere alternative the loss, regularization put aside, combines the sum of the radii r_k of the K still active hyperspheres with a penalty term greater than zero as soon as training samples, with each sample assumed to be part of the one-class, are further away from the nearest centroid than the radius of the latter:

$$\underset{W,r_1 \ldots r_K}{min} \left[\frac{1}{K} \sum_{k=1}^{K} r_k^2 + \frac{1}{\nu n} \sum_{i=1}^{n} max(0, ||\Phi(x_i; W) - c_j||^2 - r_j^2) + \frac{\lambda}{2} \sum_{l=1}^{L} ||W^l||^2 \right] \tag{3.10}$$

The second, penalty term is controlled by $\nu \in [0, 1]$, and training samples are assigned to the nearest hypersphere of center c_j. Numerous spheres centers are initialized using the k-means clustering algorithm and progressively merged during training, while each radius r_k is updated with a preset quantile parameter of the distances separating its centroid from its assigned data points. This quantile parameter is a hyperparameter of the method. The relevance of latent hyperspheres is determined thanks to the cardinality of the latent cluster they encompass. This seemed particularly relevant to handle a one-class actually containing a variety of data modes, since it could potentially capture disjoint clusters in the representation space without engulfing artificial in-between data modes within the one-class boundaries.

This work presents anomaly detection or one-class classification methods as a mean to discriminate between a diversity of points belonging to different classes using machine learning with limited supervision. While Deep MSVDD and Eq. (3.10) provided a natural extension to the intuition of SVDD and Deep SVDD by proposing the use of several hypervolumes instead of one to make the one-class boundaries more flexible and better tailored, one can wonder how to integrate additional supervision coming from labeled negative examples[9] $\widetilde{x_j}$ when the latter are available for training. This is handled by SVDD thanks to additional slack variables ξ_j:

$$\|\widetilde{x_j} - c\|^2 \geq R^2 - \xi_j, \xi_j \geq 0, \forall j \tag{3.11}$$

in the minimization of Eq. (3.8), which thus becomes:

$$\min_{R,c,\xi} R^2 + C_1 \sum_i \xi_i + C_2 \sum_j \xi_j \tag{3.12}$$

The new constraint described in Eq. (3.11) has the opposite effect of the constraint of Eq. (3.7) on negative j-indexed data points. It repels them out of the minimized hypervolume. Here again the optimization minimizes the hypervolume that contains as many one-class data points as possible while simultaneously pushing the negative samples out. Like C in Eq.(3.8), C_1 and C_2 balance the optimization objectives with the hypervolume minimization. The same repulsion intuition was proposed for Deep SVDD [153] under the name of Deep Semi-supervised Anomaly Detection (Deep SAD) [152], where in addition to the minimization of the distance to the reference centroid, the inverse of the distance to the reference centroid is added with a weight η to the training loss for the negative training samples $\widetilde{x_j}$. This translates into a similar mathematical trick where a

[8] Each of the atoms here would be defined by a hypersphere partially capturing the one-class distribution.

[9] Negative samples are out-of-distribution samples, *i.e.* data points not belonging to the one-class.

similar yet inverted term is added to the minimized quantity in order for the negative samples to be pushed away from the reference centroid, transforming Eq. (3.9) into:

$$\min_{W} \left[\frac{1}{n+m} \sum_{i=1}^{n} ||\Phi(x_i; W) - c||^2 + \frac{\eta}{n+m} \sum_{j=1}^{m} (||\Phi(\widetilde{x_j}; W) - c||^2)^{-1} + \frac{\lambda}{2} \sum_{l=1}^{L} ||W^l||^2 \right] \quad (3.13)$$

In Eq. (3.13), in addition to the n in-distribution x_i data points, the loss harnesses m labeled anomalies $\widetilde{x_j}$ during training. Here, SAD can be said to be achieved through a form of outlier exposure[10], although the latter does not necessarily rely on a semantically different auxiliary dataset [85]. The contribution of labeled anomalies to a refined one-class classification intuitively depends on the actual proximity of the labeled anomalies with the one-class boundary. Thus, in addition to the specification of a near OODD task at the beginning of this chapter, near out-of-distribution samples can be here seen as key to the success of semi-supervised AD. Labeled anomalies in the training set need to be distinguished from potential unlabeled anomalies that are considered to be normal samples, which confuse the AD by contaminating the training set instead of providing supervision. Robustness to such contamination is critical since experts providing the labels with which neural networks are trained are not exempt from mistakes. Experiments evaluating such robustness are proposed in sections 3.3.1 and 3.3.3. This semi-supervision adaptation can be repeated for Deep MSVDD, although in that case the multiplicity of normality centers calls for an additional consideration on how to choose from which centroid the labeled anomalies should be repelled as long as several centroids are kept active. One could also think of weighted inverse distances to the active centroids, the weights possibly implementing latent data modes attention. The experiments implementing Deep MSVDD adapted to SAD with an additional loss term for labeled anomalies were inconclusive, such an adaptation will therefore not be further discussed. The additional loss term evaluated to train Deep MSVDD in a SAD context either minimized the latent distance between anomalies and dedicated centroids, or maximized the latent distance between anomalies and normality, one-class, centroids. The labeled negative samples bringing additional supervision during training for SAD can be created through the transformation of in-distribution samples. Thus, a form of self-supervision can be associated to SAD. This possibility is explored in 3.3.3.

Arbitrary sets of outliers could not be completely gathered around a reference point since they do not necessarily belong to a common mode [152, 172]. However, this does not forbid the concentration of identified modes among labeled anomalies close to dedicated centroids to provide additional supervision during training, a case which is part of the experiments presented in section 3.3.3. The possibly arbitrary distribution of normal and anomalous centroids and the relative distance between the centroids adds a way to use prior information regarding the proximity between the training samples. Such a setup can seem close to classification with rejection [85, 23], since the concentration of data points around dedicated normal and anomalous centroids can be interpreted as classification while the data points attached to no centroid and thus supposedly repelled from all centroids by the trained network constitutes a rejection. This parallel with classification with rejection is not necessarily relevant since the availability of labeled anomalies to train machine learning-based AD methods is usually very limited if not nonexistent. In contrast, supervised classification of identified data modes would imply rich, representative and relatively balanced datasets for each latent mode. The limited availability of labeled anomalies applies to actual anomalies and not to artificial anomalies provided by the transformation of existing training samples i.e. through self-supervision. With proper transformations self-supervision can produce as many

[10] Outlier exposure consists in taking advantage of labeled anomalies.

labeled anomalies for training as there are normal samples, or even more if each normal sample is transformed multiple times. However this does not overcome the lack of representativeness of labeled anomalies. This is also made difficult since the choice of transformations requires expert knowledge to control the semantic implications of the transformations. Such transformations should also be selected according to the properties of the neural network when doing deep AD, for instance to take into account the invariances implemented within the architecture.

3.1.3 Deep random projection outlyingness

In addition to Deep MSVDD, a second Deep SVDD variant considered here is Deep RPO [25], which replaces the latent Euclidean distance to the normality centroid with a RPs-based outlyingness measure adapted from Eq. (3.2) in the latent space. This modification leads to the following minimization problem:

$$\min_{W} \left[\frac{1}{n} \sum_{i=1}^{n} \left(\underset{u \in \mathbb{U}}{mean} \frac{|u^T \Phi(x_i; W) - MED(u^T \Phi(X; W))|}{MAD(u^T \Phi(X; W))} \right) + \frac{\lambda}{2} \sum_{l=1}^{L} ||W^l||^2 \right] \tag{3.14}$$

This training loss uses the outlyingness defined in Eq. 3.2 after the neural network encoding, with a *max* estimator transformed into a *mean* as suggested in [25] for better integration with the deep learning setup. The *mean* estimator computes a mean over the set of RPs available to compute the latent outlyingness, while the $\frac{1}{n}$ computes a mean over the sample of size n. The use of a *mean* instead of a *max* removes the convergence to the Mahalanobis distance-inferred ellipsoid (up to a constant factor) already mentioned in the RPO definition (see 3.1.1) for a large set of RPs. This RPO variant however remains affine invariant (see Appendix B). The loss nonetheless still combines 1D outlyingness measures individually centered by their median and normalized by their median absolute deviation, but with no ellipsoid-like score distribution guarantee in the input space of the RPO once integrated (see Appendix A). Note that the input space mentioned here is the output space of the encoding neural network in the case of Deep RPO. No square was applied to the first loss term, in accordance with the quantity put forward in [62]. An SVDD adaptation where the latent distances are computed using a Mahalanobis distance has been proposed in [178], but the latter does not encode data with a neural network. Combining the deep version of SVDD with a Mahalanobis score would be another way to achieve a trainable latent normality representation based on an ellipsoid of minimal volume [181]. Since Eq. (3.14) relies on an encoding deep neural network, one way to see the difference between Deep RPO with *mean* and Deep RPO with *max* is to consider that while with *max* the gradient will be based on a single projection at a time for each sample, the *mean* systematically produces a gradient stemming from all projections simultaneously.

We can also define an evolution of (3.2) and (3.14) with multidimensional random projections, as we proposed in [25]. In the case of random projections leading to a single output dimension, we have the following setting: if d is the data samples dimensionality, and m the random projection output dimensionality, a random projection u with $m = 1$ will lead to a single projected coordinate $u^T x$ for any individual sample x with d dimensions. This projected coordinate can then be compared to a location estimator computed with the application of u on all the available samples, for instance $u^T x - MED(u^T X)$ where the location estimator chosen is the median. On the other hand, for multidimensional random projections, i.e. $m > 1$, a covariance matrix $\Sigma_{m \times m}$ can be harnessed in the reduced space to obtain a more subtle normalized distance to location estimators. Each sample x can then be associated with a robust Mahalanobis distance,

transforming Eq. (3.2) into:

$$\mathcal{O}_{RPO_{multidim}}(x; p, X) = \max_{u \in \mathbb{U}} \sqrt{(u^T x - MED)^T \Sigma^{-1}_{m \times m} (u^T x - MED)} \tag{3.15}$$

where MED stands for the median projected vector $MED(u^T X)$. In this configuration, each sample has an outlyingness based on m projected coordinates per RP. Each of the projected coordinates is compared to a location estimator determined on each random projection dimension. One projected location estimator is thus computed over all data samples, for each output dimension of the random projections in use. This transforms the training objective described by Eq. (3.14), the normalized distance to the median in Eq. (3.2) before the integration by a max operator becoming:

$$\sqrt{(u^T \Phi(x) - MED)^T \ \Sigma^{-1}_{m \times m} \ (u^T \Phi(x) - MED)} \tag{3.16}$$

where MED stands for the median latent projected vector $MED(u^T \Phi(X; W))$. Thanks to Eq. (3.16), the Deep RPO training loss now has the possibility to incorporate multi-dimensional projected representations for data samples, enabling additional latent representation flexibility. As for Deep SVDD, this new RP-based setup can incorporate labeled anomalies $\widetilde{x_j}$ during training using an inverse distance to repel negative samples from the in-distribution data centroids:

$$\begin{aligned} \min_W \Bigg[& \frac{1}{n+m} \sum_{i=1}^{n} \left(\underset{u \in \mathbb{U}}{mean} \frac{|u^T \Phi(x_i; W) - MED(u^T \Phi(X; W))|}{MAD(u^T \Phi(X; W))} \right) \\ & + \frac{\eta}{n+m} \sum_{j=1}^{m} \left(\underset{u \in \mathbb{U}}{mean} \frac{|u^T \Phi(\widetilde{x_j}; W) - MED(u^T \Phi(X; W))|}{MAD(u^T \Phi(X; W))} \right)^{-1} \\ & + \frac{\lambda}{2} \sum_{l=1}^{L} ||W^l||^2 \Bigg] \end{aligned} \tag{3.17}$$

Experiments evaluating the performances associated with Eq. (3.14) are proposed in 3.3.2 and in 3.3.3. Experiments implementing Eq. (3.15) are presented in 3.3.2. The latter include the proposal of a dropout [171] over entire random projections, and over random projection components.

3.1.4 Density and boundary one-class characterization

We have discussed OCC driven by SVDD and its deep learning variants Deep SVDD, Deep SAD and Deep MSVDD. One of the prominent features of all the aforementioned methods is that they achieve OCC through the explicit or implicit definition of a one-class boundary. More importantly, this opposes these approaches to density estimation-based outlier detection, where an actual density distribution is estimated. This key difference between boundary estimation and density estimation is illustrated on Fig. 3.3. The point of opting for boundary one-class characterization is that it can require less data than a density estimation approach, the latter necessitating a training set very authentically distributed. This requirement is even more critical in high dimensional problems [175]. In Deep SVDD, Deep SAD and Deep MSVDD the boundary is made of one or several hyperspheres in the output space of the encoding neural network. One of the losses put forward by the original Deep SVDD paper [153] actually has the hypersphere radius as a term in the sum to be minimized, as in the minimization of Eq. (3.8) defining the original non-deep SVDD. In Eq. (3.10), the sum of radii similarly defines a term in the training loss. These radii, in addition to the hypersphere centroids, completely define a one-class boundary without density description. In Eq. (3.9), no

radius is taken into account in the training loss which only contains the centroid as the in-distribution samples location estimate, making the boundary only implicit this time. One could call Deep SVDD, associated with this objective function, a distance-based [118] OCC since the distance to the centroid defines both the minimized loss and the AD score at test time, making this score independent of an actual boundary.

Among the most common density estimation approaches are the normal density estimation and the Gaussian Mixture Model mentioned in section 1.3.2. The normal density estimate relies on the computation of a covariance matrix and mean estimate, leading back to the Mahalanobis distance (cf. section 3.1.1). The challenge of high dimensional data for density estimation is revealed in such a case since it can make the covariance matrix estimate singular and may therefore dictate proper regularization. Such normal density estimation brings Deep RPO nearer to density-based OCC since RPO and Deep RPO also comparably rely on estimates of the training data location and spread, respectively provided by the median and the median absolute deviation. In terms of performances, boundary characterization should be used for OCC when the data availability is too weak for a density-based approach, and is especially relevant when labeled outliers are available to refine the one-class boundary. On the other hand, a density estimation approach should be used when the training data availability allows it, the latter likely leading to better performances [175]. One should keep in mind that other kinds of OCC exist besides density, distance and boundary-based OCC, for instance the isolation method that is IF [118].

3.1.5 Latent space regularization and specialization

It is important to notice that when talking about AD using the representation from an AE's bottleneck representation or the output of Deep SVDD's [153] neural network, one considers nothing more than using an encoded version of the input data. The relevance of the information saved in the data representation depends primarily of the task used to train the encoding neural network, which in the case of the AE is usually reconstruction, perhaps including denoising [191], and in the case of Deep SVDD is the normal training data latent isotropic concentration. One of the key challenges in the design of such an encoding neural architecture is not only to choose wisely the transformations contained in the successive layers, responsible for information retrieval, but also to select an appropriate output representation to encode the data. As Deep SVDD's latent normality hypersphere collapse risk showed [52, 153], neural network design choices or the association of limited training supervision and a specific loss can favor the collapse of the output representations distribution, *i.e.* to a naive and useless constant mapping. This raises the question of the evaluation of the quality of the distribution in the output encoding space, which the literature could call an embedding space [157], encoded data points actually defining embeddings. This evaluation should typically be sensible to the partial or complete collapse of the embeddings distribution. This matter was examined in [95], where the authors observe the complete or partial collapse in the embedding space using the curve described by the sorted singular values of a covariance matrix computed over a set of embeddings. On such a curve, a sharp collapse of these singular values on a logarithmic scale can highlight a partial collapse, or as they call it, a dimensional collapse. This led us to unsuccessfully experiment with regularization terms computed using the singular values of the minibatch output representations covariance matrix for Deep SVDD. We thus defined a singular values uniformity regularization term $R_{uniformity}$:

$$R_{uniformity} = log\left(\frac{\Lambda_{max} + \epsilon}{\Lambda_{min} + \epsilon}\right) \tag{3.18}$$

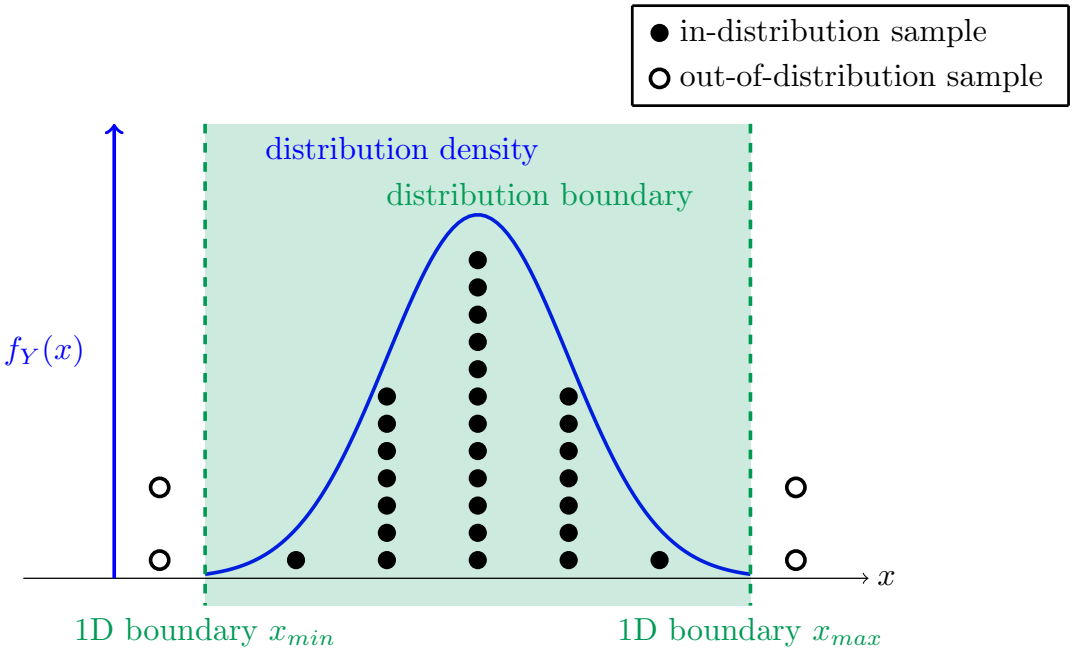


Figure 3.3: Illustration of the difference between boundary and density-based OCC. Here, the one-class boundary discriminating between in and out-of-distribution samples is defined by the domain $[x_{min}, x_{max}]$. This boundary can evidently benefit from the near OOD samples to refine its range, and holds no information describing the density of the one-class within the boundary. Characterizing this density intuitively requires more data, the density estimation being accurate only if the data distribution leading to the estimate is correctly distributed. Supposing a successful density estimation, the AD score proposed by a boundary will necessarily be less relevant than the one provided by the density estimate. Harnessing a SVDD-based OCC here could translate into the use of the distance between a test sample and the peak of the density estimate. This further depicts how SVDD-inspired approaches rely on a simple distance and not on a density information.

and a singular values spread regularization term R_{spread}:

$$R_{spread} = log\left(\frac{1}{\Lambda_{min} + \epsilon}\right) \tag{3.19}$$

In both equations Λ_{min} and Λ_{max} represent respectively the smallest and the largest eigenvalue of the minibatch output representations covariance matrix. The ϵ is a small constant parameter added to the denominator of fractions to ensure numerical stability. The $R_{uniformity}$ regularization term encourages the eigenvalues of the covariance matrix to remain close to each other, penalizing any partial collapse in the eigenvalues distribution. The R_{spread} punishes low eigenvalues, to help discard uniformly distributed but very small eigenvalues. This term is called *spread* since discarding small eigenvalues for the covariance matrix of the embeddings guarantees some variance among the embeddings. A very small variance along one or more embedding dimensions would indeed translate into at least one very small covariance matrix eigenvalue. Note that whereas [95] used the term singular values, we speak of eigenvalues because these quantities are equal in the case of real symmetric positive definite matrices, covariance matrices belonging to this manifold of matrices. We also experimented with another R_{spread} term which has the advantage of systematically taking into account all eigenvalues, and not just the one or two highest or lowest ones:

$$R_{spread} = log\left(\frac{1}{\sum_{e=1}^{d} \Lambda_e + \epsilon}\right) \tag{3.20}$$

The regularization provided by Eq. (3.20) thus ensures the gradient of the loss encompasses all dimensions of the output representations at each optimization step. Two intuitive comments can be made in such a case: the regularization impact is smoothed over all dimensions, and one optimization step can not freely focus on one output dimension at the expense of the distribution of representations along other dimensions.

Such a regularization setup is related to what [52] proposed, where they encourage the mean minibatch variance over all dimensions to remain above a threshold thanks to a dedicated regularization term which is nonzero only below the threshold. This was implemented along with an adaptive regularization weight defined with a momentum term and the ratio of the actual Deep SVDD loss with the regularization term. The proximity of this regularization mechanism with ours relies on the fact that both relate to the covariance matrix. The mean minibatch variance across dimensions considered in the regularization proposed by [52] even corresponds to the mean of the diagonal values of the covariance matrix from which we compute the eigenvalues used in Eq. (3.18) and Eq. (3.19). This mean minibatch variance even corresponds to the eigenvalues mean when the components of the output representations are uncorrelated, *i.e.* when the embeddings covariance matrix is diagonal. Regularizing in such a way the embeddings covariance matrix eigenvalues specializes further the output representation space. The final training loss, when including both of our supplementary regularization terms, would for instance be defined by the following equation in the case of the vanilla Deep SVDD setup, that is without taking into account labeled out-of-distribution samples during training:

$$\min_{W}\left[\frac{1}{n}\sum_{i=1}^{n}||\Phi(x_i;W)-c||^2 + \frac{\lambda}{2}\sum_{l=1}^{L}||W^l||^2 + \lambda_u R_{uniformity} + \lambda_s R_{spread}\right] \tag{3.21}$$

The two eigenvalues-based regularization terms are balanced through the fixed weights λ_u and λ_s. The ratio $\frac{\Lambda_{max}}{\Lambda_{min}}$ is the condition number of the covariance matrix and describes how well the inverse of the covariance matrix can be computed by classical numerical

methods. It is related to the ratio between the number of dimensions in the output representation space d and the number of samples n used to evaluate the covariance matrix, *i.e.* $\frac{d}{n}$. Here n amounts to the batch size. If this ratio is relatively large, *i.e* we estimate the covariance matrix with not that many samples when compared with the number of dimensions of the output space, the condition number tends to be large indicating an ill-conditioned estimation [187]. This is important in our regularization proposal since it makes it irrelevant for all neural network architectures and training batch sizes where $\frac{d}{n}$ ends up being quite large. One of the advantages of the minibatch variance regularization proposed in [52] is that it does not rely on a covariance matrix estimation and instead directly estimates the diagonal terms, the variance along each output dimension, of the latter to compute their mean value to finally define a penalty term. This does not go without reminding us one of the points of using the random projection outlyingness (see Eq. (3.2)) instead of the Mahalanobis distance (see Eq. (3.1)), which was precisely to avoid relying on a covariance matrix estimation to produce a robust outlyingness measure. Using the regularization terms defined by Eq. (3.18), Eq. (3.19) and Eq. (3.20) in semi-supervised OCC setups notably implies choosing whether to include negative labeled samples of the training set in the embeddings covariance matrix estimation preceding the eigenvalues decomposition.

Constraining a representation space to avoid dimensional collapse[11] can recall the discretization of the AE bottleneck of [75] mentioned in section 3.1.1. The definition of latent prototypical representations could also be constrained so that a certain metric of dimensional collapse remains below a predefined threshold. This would involve adapting the training loss of the proposed memory-augmented autoencoder since in the published setup the prototypical representations are trained with a backpropagation discarding the prototypical representations with zero as attention weight. The reunion of normal latent representations achieved through the Deep SVDD-based losses put forward is analogous to the alignment principle put forward in [196], which also argued for a latent uniformity. Whereas the alignment principle compels similar samples to be assigned similar representations, the uniformity principle demands the preservation of maximal information. One way to achieve that according to [196] is to push all features away from each other on the unit hypersphere to intuitively facilitate a uniform distribution. Here the unit hypersphere is the manifold on which representations lie, and not a minimal volume holding in-distribution samples. This uniformity principle is rarely mentioned in the literature even when the building of an embedding space is discussed.

The extension of the Deep SVDD loss to encourage a form of latent uniformity using the pairwise distance between normal samples during training was investigated without ever improving the baselines. Such latent uniformity can be interpreted as a latent space regularization as well. The experiments conducted to evaluate the contribution of a pairwise distance of normal samples latent representations loss term revolved around the following training loss format, where the term tasked with enforcing latent uniformity is weighted using $\lambda_{uniformity}$ and was expected to be judiciously balanced with the overall latent concentration:

$$\min_{W} \left[\frac{1}{n} \sum_{i}^{n} \|\Phi(x_i; W) - c\|^2 + \frac{\lambda_{uniformity}}{n} \sum_{i \neq j}^{n} \left(\|x_i - x_j\|^2 \right)^{-1} + \frac{\lambda}{2} \sum_{l=1}^{L} ||W^l||^2 \right] \quad (3.22)$$

The failure to make a loss term enforce a form of beneficial latent uniformity could signal the necessity of associating such a constraint with latent representations confined to a relevant manifold. In our experiments, the representation were free to lie anywhere in the Euclidean output representation space, without any constraint. Enforcing the

[11] Dimensional collapse is the collapse of the variance in some of the representation space dimensions. Intuitively, this amounts to the irrelevance of some of the components in the associated representations.

uniformity principle is intuitively appealing for Deep SVDD-like training objectives because it would simultaneously discourage latent hypersphere collapse, *i.e.* the learning of a fixed mapping to a constant output representation.

3.2 SPD-manifold specific processing

The processing of spectrums and time-series led us to consider methods stemming from the active research field devoted to geometric deep learning [31], this time not for the deep learning architectures adapted to data on graphs (see 2.4.2) but for learning parameters and representations constrained to Riemannian manifolds with a special interest for the SPD matrices manifold. Said in simple words and without being formally exhaustive a manifold is a smooth space where each point admits a Euclidean tangent space. As an intuitive example, one can think of a smooth surface embedded in a Euclidean space: each point admits a tangent space from which points can be mapped back to the surface. Riemannian manifolds are manifolds with a Riemannian metric which enables the computation of distances and angles. Paradoxically, the availability of a Euclidean tangent space at each point on the manifold is actually key to the computation of manifold-aware metrics and gradients. Indeed, a Riemannian metric can hide an inner product on the tangent space of a given point lying on the manifold, and the optimization of parameters can be defined through the repeated projection of a gradient descent in the tangent space back to the manifold. Regarding the Riemannian gradient, see 3.2.2.

Recent developments offer approaches specialized in processing data points represented by symmetric positive definite (SPD) matrices. These approaches typically constrain representation learning in the intermediate layers of a neural network to the SPD matrices Riemannian manifold, which defines a convex half-cone in the vector space of matrices [141]. Doing so implies redefining linear and non-linear operations commonly used in deep learning, and to replace the usual backpropagation with a Riemannian one. The use of manifold-aware operations and statistics for radar targets detection and discrimination was previously put forward in [35, 38, 18, 204]. A real-valued matrix $M \in \mathbb{R}^{d \times d}$ is symmetric positive definite if and only if:

$$x^T M x > 0 \ \forall x \in \mathbb{R}^d \setminus \{\mathbf{0}\} \tag{3.23}$$

Processing spectrums and time-series naturally lead to work with SPD representations. In the case of spectrums for example, a series of spectrums, each associated with a pulse Doppler radar burst, allows for the computation of a covariance matrix of frequency bins over time. Time-series of one or a handful of dimensions can be characterized using autoregressive models which in turn are associated with Toeplitz matrices of autocorrelation coefficients, the latter being a specific case of SPD matrix. The generation of such Toeplitz matrices is discussed in 2.3.1. In the case of the series of radar spectrums however, we move away from the radar hits encoding problem we are interested in since one hit is defined over a single burst, *i.e.* one hit generates a single spectrum. This would not prevent this spectrum from being processed with CNNs or the complex-valued samples in time to be processed as a time-series. A covariance matrix could be built over the output channels of a convolutional layer, which could be understood as a second-order pooling [13, 93]. Such covariance computation layers can notably be adapted to include first order information in the generated intermediate representation while maintaining the SPD nature of the output, thanks to the two following formulations available in the literature [207, 46]:

$$\begin{pmatrix} \Sigma + \beta^2 \mu\mu^T & \beta\mu \\ \beta\mu^T & 1 \end{pmatrix} \tag{3.24}$$

$$\begin{pmatrix} \Sigma + \beta\mu\mu^T & \beta\mu \\ \beta\mu^T & \beta \end{pmatrix} \tag{3.25}$$

In Eq. (3.24) and Eq. (3.25), Σ is a covariance matrix, μ a mean vector, and β a constant parameter. These SPD representations combining first and second order statistics were implemented for the *ICLR 2021 Computational Geometry & Topology Challenge* [12] [129]. They define the following mapping, where $\mathcal{S}_*^+$ is a manifold of SPD matrices of adequate dimensions:

$$(\mu, \Sigma_{d\times d}) \to \Sigma_{(d+1)\times(d+1)} \tag{3.26}$$

$$(\mathbb{R}^d, \mathcal{S}_*^+) \to \mathcal{S}_*^+ \tag{3.27}$$

As suggested by [34], the complex-valued nature of raw IQ data can benefit from the definition of a similar processing framework for Hermitian Positive Definite (HPD) matrices, which are the complex-valued equivalent of Symmetric Positive Definite real-valued matrices. Furthermore, whereas in the earlier mentioned Mahalanobis distance (see Eq. (3.1)) one used the covariance matrix to compute the distance between a test data point and a reference distribution, here the covariance matrix represents a test data point by capturing the distribution of its features generated by convolutions. Transforming SPD matrices with specialized neural networks open the way to manifold-constrained learning for one-class classification. As we will see in 3.2.3, SPD neural networks are not the only way of conducting one-class classification specific to SPD representations. In the remainder of this section, building blocks of SPD neural networks and SPD-specific one-class classification will be presented.

3.2.1 SPD neural network operations

Let us first define a linear operation for SPD neural networks. This linear operation is a bilinear mapping called BiMap [89] and relies on a trainable parameters matrix of full-rank $W \in \mathbb{R}_*^{d_k \times d_{k-1}}$ to transform the input X_{k-1} SPD matrix into the output X_k SPD matrix:

$$X_k = W_k X_{k-1} W_k^T \tag{3.28}$$

Here, d_k is the output square matrix dimension, while d_{k-1} is the input square matrix dimension, which indicates the BiMap layer also allows to reduce the intermediate SPD representations size. This reduction in the SPD representation size in turn implies that representations are going from one SPD matrices manifold to another of lower dimensionality. The trainable W_k matrix is kept orthogonal, and thus ends up belonging to a compact Stiefel manifold in order for the representation produced by Eq. (3.28) to remain SPD matrices and to enable the optimization of the parameters [89]. Now that SPD neural networks are equipped with a linear transformation, one needs to define a nonlinear one. In a similar fashion to the rectified linear unit (ReLU), [89] proposed the ReEig layer that rectifies the small eigenvalues of an SPD matrix, increasing the ones above an arbitrary threshold to the threshold value:

$$X_k = U_{k-1} max\left(\epsilon I, \Lambda_{k-1}\right) U_{k-1}^T \tag{3.29}$$

[12] Code produced during our participation available at `https://github.com/Blupblupblup/challenge-iclr-2021`. Accessed: 06-10-2022.

In both Eq. (3.29) and Eq. (3.30), the matrices U_{k-1} and Λ_{k-1} are respectively the square and diagonal matrices produced by the eigenvalue decomposition $X_{k-1} = U_{k-1}\Lambda_{k-1}U_{k-1}^T$. The matrix I is the identity matrix. After a linear transformation that can reduce the representation size and a nonlinear activation, [89] proposes another nonlinear layer called LogEig:

$$X_k = U_{k-1} log\left(\Lambda_{k-1}\right) U_{k-1}^T \tag{3.30}$$

The LogEig layer serves as a mapping from the manifold to a Euclidean representation. This mapping stems from the logarithmic and exponential mappings that allow to go respectively from a Riemannian manifold to the Euclidean tangent space and vice versa. The LogEig layer is actually a specific case of the logarithmic mapping in the SPD matrices space where the reference of the mapping is the identity matrix [32, p.101]. The transposition of the now defined BiMap, ReEig and LogEig layers for the real-valued case of SPD ($\in \mathcal{S}_*^+$) matrices to the equivalent complex-valued case of HPD matrices ($\in \mathcal{H}_*^+$) has been detailed in [32], which also puts forward a simple way to bring the HPD case to a real-valued SPD one through the following complex components mean:

$$\forall H \in \mathcal{H}_*^+, \frac{1}{2}(H_r + H_i) = S \in \mathcal{S}_*^+ \tag{3.31}$$

where H_r and H_i are the real and imaginary parts of the input HPD representation respectively. A manifold-aware Riemannian batch normalization [92] adapted to SPD representations was proposed in [35] and further discussed along with the proposal of variants in [103, 102, 121]. The mean used in the batch normalization of [35] is a Riemannian barycenter corresponding to the Fréchet mean. This geometric mean has no closed form when more than two SPD matrices are involved and is computed using an iterative method called the Karcher flow algorithm [97].

3.2.2 SPD neural network gradient and backpropagation

Transforming SPD representations into other SPD representations implies a set of constraints over the operations conducted within the SPD neural network. These constraints can be divided into two categories: the constraints over the parameters used to compute new intermediate representations, and the constraints over the gradient computation with respect to these parameters. The first category dictates that the W_k bilinear mapping matrix is required to be part of the Stiefel manifold as indicated in 3.2.1. The second category implies the definition of specific manifold-aware gradients for the backpropagation during training. For instance, computing the gradient to update parameters constrained to a Riemannian manifold such as the ones defining the bilinear transformation of Eq. (3.28) implies three steps: first determine the Euclidean gradient, then retrieve the normal component with respect to the tangent space of the parameters matrix on the Riemannian manifold, and then finally project the gradient descent-updated parameters in the tangent space on the manifold [35, 31, 89, 82]. To get back to the Riemannian manifold from the tangent space one can use a Riemannian exponential mapping, or a first-order approximation of the latter called a *retraction* [135, 12].

On the other hand, passing the gradient through the SPD neural networks nonlinearities such as Eq. 3.29 and Eq. 3.30 requires other specific operations adapted to the eigenvalues decomposition. In such cases backpropagation relies on the matrix generalization of backpropagation [89, 93] which will not be detailed here since it is quite elaborate and does not contain any original contribution from this thesis. One can note that [35] required both kinds of gradient adaptation to implement a Riemannian batch normalization for SPD neural networks. All experiments using the SPD neural

network operations and backpropagation were conducted using a third party implementation[13] publicly available [10]. One can note the proximity of using a Riemannian gradient in the context of an SPD neural network and with the intuition of the natural gradient put forward in [16] which takes into account the possible non-Euclidean structure of a parameter space.

3.2.3 One-class classification specific to SPD matrices

Using the previously defined SPD neural network operations and backpropagation, one can adapt the already presented Deep SVDD and Deep SAD one-class classification to an SPD setup. In such an approach, input SPD representations, *e.g.* covariance matrices, are transformed into other SPD representations of similar or lower dimensionality, and the Riemannian distance to the Riemannian mean output representation is either minimized or maximized during training. The lower dimensionality is controlled thanks to the BiMap constrained parameters matrix W shown in Eq. (3.28). An example architecture of such an SPD-manifold aware Deep SVDD adaptation is detailed in section C.2. The mean output representation serving as reference point during training is the mean training data output representation computed before training, in accordance with the heuristic put forward in [153]. Note that this Riemannian mean is a geometric mean of the SPD representations in the output space of the SPD neural network and corresponds to what was done in [35] to define a Riemannian batch normlization as was previously explained in 3.2.1.

The distance minimization in the output space equally constrained to the SPD matrices manifold implies the use of a Riemannian distance between two arbitrary SPD matrices S_1 and S_2. In our experiments, we used either the Log-Euclidean metric (LEM):

$$dist(S_1, S_2) = \|log\,(S_1) - log\,(S_2)\|_F \tag{3.32}$$

or the affine-invariant metric (AIM):

$$dist(S_1, S_2) = \left\| log\left(S_1^{-\frac{1}{2}} \cdot S_2 \cdot S_1^{-\frac{1}{2}} \right) \right\|_F \tag{3.33}$$

The LEM is actually a specific case of the AIM [32, p.101].

Since we consider input, output and intermediate representations belonging to the SPD matrices manifold, no LogEig (see Eq. (3.30)) layer is necessary to plug representations in Euclidean layers. Such a fully Riemannian approach can be opposed to [38, 207] which suggested to use SPD-specific operations in portions of a processing pipeline otherwise Euclidean, these setups actually requiring logarithmic mappings to get back to Euclidean representations. Two other non-deep manifold-aware OCC approaches are put forward in this work, both relying on tangent PCA (tPCA). The tPCA projects SPD points on the tangent space of the Fréchet mean, a Riemannian mean which allows to compute an SPD mean, keeping the computed centroid on the Riemannian manifold naturally occupied by the data. A common principal component analysis [77] is then performed in the tangent space at the Fréchet mean. Using tPCA offers the advantage of being sensible to the manifold on which the input samples lie, but implies that input data is centered around the Riemannian mean and not too scattered. This makes tPCA a questionable choice when the objective set is AD with multimodal normality [140]. In other words, although the linear approximation of the SPD inputs around its Riemannian mean provides us with a manifold-aware dimensionality reduction, the distribution of the inputs in the SPD matrices manifold may lead this approximation to be excessively inaccurate, leading to irrelevant reduced representations. This is due to the linear approximation not preserving the Riemannian distances between points [169].

[13] `https://gitlab.lip6.fr/schwander/torchspdnet` Accessed: 28-10-2022

The first non-deep manifold-aware OCC consists in replacing the principal component analysis-based dimensionality reduction usually part of machine learning pipelines preprocessing with tPCA dimensionality reduction. To achieve OCC following the dimensionality reduction one only need to plug the reduced representations provided by tPCA in a OCC method afterwards. The second manifold-aware OCC approach consists in using the norm of the last components of the tPCA as an AD score, an anomalous sample being considered as out of the one-class for OCC. This would be the Riemannian equivalent of taking the last components of a vanilla PCA as an AD score, a method called *negated PCA*. This negated PCA is motivated by the possibility that, in one-class classification where model fitting occurs on normal data only, the first principal components responsible for most of the variance in normal data are not the most discriminating ones when it comes to distinguishing normal samples from anomalies [129, 147, 148]. These two last manifold-aware approaches relying on tPCA are not fully Riemannian either, in reference to the purely Riemannian Deep SVDD SPD adaptation previously proposed. In the first case the OCC method is not Riemannian in any way, and in the second case the AD score is not manifold-aware, whereas for the Riemannian Deep SVDD SPD adaptation the final score is provided by a Riemannian metric in addition to all transformations being manifold-aware. A non-exhaustive map of the OCC approaches mentioned in this chapter is proposed on Fig. 3.4.

3.3 One-class classification experiments

As indicated in Chapter 1, we consider unsupervised AD as experiments where no labeled anomalies are available during training to refine the implemented discrimination, and SAD experiments as experiments where in addition to the samples belonging to the one-class, a small minority of unrepresentative labeled anomalies is accessible during training. The OCC experiments presented in this section are presented in their order of publication in conference and workshop papers, *i.e.* each upcoming section refers to specific datasets while combining experiments with different levels of supervision.

3.3.1 OCC on high-resolution range profiles

The experiments presented in this section stem from our 2020 IEEE Radar Conference paper [24] which applied OCC to HRRPs. The dataset is composed of real radar 1D range profiles (RP) generated by a very high performance radar for coastal surveillance, the Coast Watcher 100 Thales radar (see Fig. 1.8). Each range profile is composed of 200 cells with various intensities. The labelling of the data samples stems from the AIS[14] announced ship types. This constitutes a first potential difficulty in the creation of the dataset since the labels provided by AIS are functional (*i.e.* they describe the function of a ship, not its appearance and dimensions). Another possible interpretation of our training set pollution with unlabeled anomalies for unsupervised AD is the mislabeling of AIS data, or the intra-class variance of AIS labels, both interpretations affirming the point of experimenting with such pollution. Note that here pollution has the same meaning as contamination in 3.3.3. The dataset is balanced and composed of four classes of range profiles: the latter either belong to cargo ships, fishing ships, passenger ships or tankers. A characteristic example RP of each class can be seen on Fig. 3.5. A total of 12000 HRRPs are available for each class.

A possible operational use of anomaly detection with the previous classes could be the following: considering a busy but regulated waterway in terms of fishing, an operator could be helped by automatic AD alerts detecting fishing ships in the area. This implies

[14] Automatic Identification System

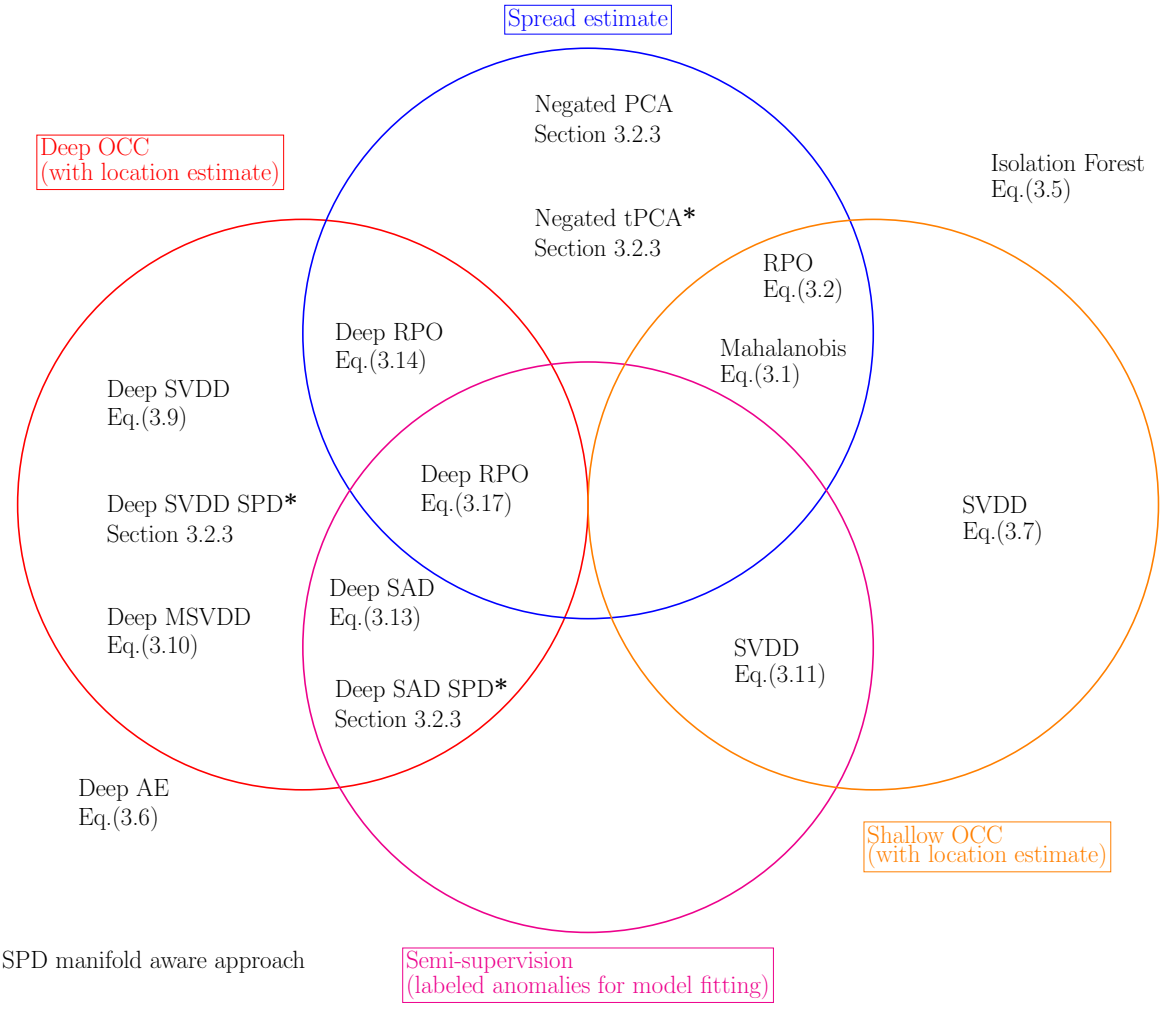


Figure 3.4: Non-exhaustive map of the OCC approaches mentioned in chapter 3. Each method name is associated with a reference to the equation or paragraph describing it. Not all methods are positioned on the illustration since the aim of the figure is to map either the proposed approaches based on Deep SVDD with respect to the latter, or the manifold-aware methods. Additionally, an isolation-based OCC such as IF (see Eq. (3.5)) has no place in the categorization proposed here. A deep OCC auto-encoder using a reconstruction error as training loss and OOD score is mapped outside of the red circle even though it learns to capture the training data distribution, *i.e.* the one-class distribution. This choice stems from the AE learning the distribution only implicitly. In such a case, no centroid or one-class reference coordinates are available in any representation space as in-distribution data location estimate, although one could harness a clustering algorithm on the representations generated by the trained AE. The part of the semi-supervision circle without intersection is empty since the circle is drawn to highlight the shallow and deep methods able the take advantage of a minority of labeled anomalies during model fitting. One can note that all the deep OCC approaches represented here should be within the semi-supervision circle had we opted for the alternative AD supervision categorization put forward in [47] as explained in 1.3.2.

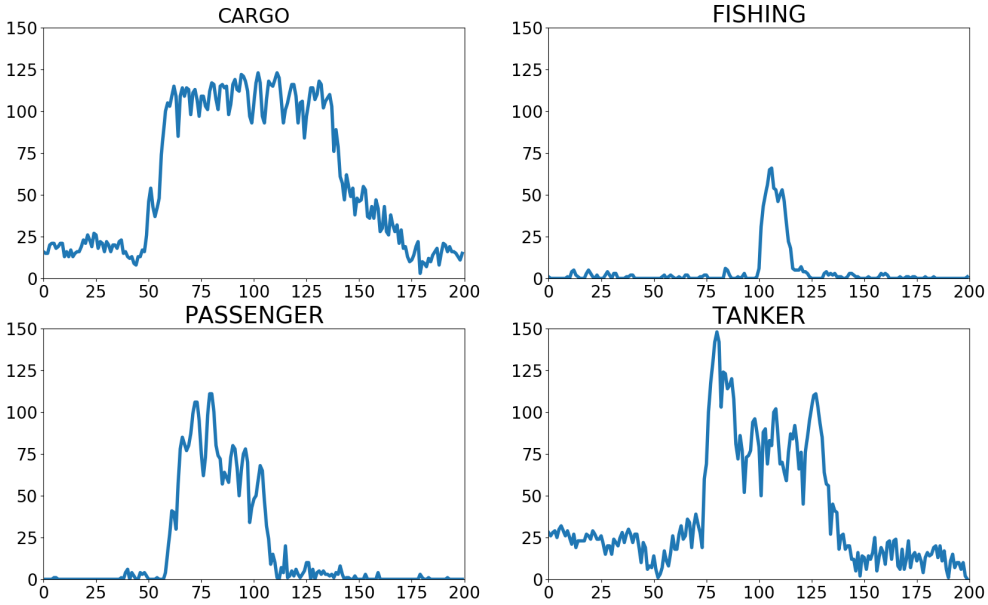


Figure 3.5: Example range profiles (one per class considered). Horizontal axis denotes the range cell index, vertical axis denotes the amplitude.

normality is defined by the three other classes (cargo ships, passenger ships, tankers), for which many samples should be available. Note that our experiments define normality based on a single class of ships, but the described operational use can still be achieved by combining the results of AD on each normal class. This scenario can be easily extended to a variety of realistic surveillance contexts: detection of specific ships despite the shutdown of AIS, detection of military ships without AIS and IFF[15].

In order to obtain range profiles expressive enough for relevant experiments, the data samples chosen for our experiments were beforehand selected according to their combined range cells energy. This selection aims at avoiding both too small and too high energies as a crude effort to avoid model bias imputable to the outliers of each class. This work deliberately avoids elaborate preprocessing to reveal the potential of AD on raw HRRPs. We will see in our results AD methods applied on data with and without normalisation. Outside this samples preselection and non systematic normalisation, no steps are taken in order to counter amplitude-scale, time-shift and target-aspect sensitivities, which seems uncommon when compared with other HRRP processing approaches [67, 63, 194]. Regarding time-shift sensitivity, one should note that most of our samples are nonetheless approximately aligned. About the diversity of targets, no exact count was made of the ships eventually retained by our preselection, but this diversity is above the one observed in other studies dedicated to HRRP targets such as [194].

Our unsupervised AD results aim at comparing between shallow and deep AD methods, but also at individually appreciating the sensitivity to training pollution of each unsupervised AD method, with fixed hyperparameters through the pollution changes. We maintain constant hyperparameters when polluting the training set in order to stay relevant regarding an actual implementation of one of the methods considered with imperfect radar data. The results of Deep SAD are here to demonstrate the ability of labeled anomalies to contribute to Deep AD through recent semi-supervised approaches adaptable to HRRP data. Each unsupervised and semi-supervised experiment includes detecting anomalies of at least two classes completely unseen during training at test stage. As was already mentioned, this is fundamental in order to respect the diversity

[15] Identification Friend or Foe

and unpredictability of anomalies. The metric defining all results is the area under the receiver operating characteristic curve (ROC AUC), as was used in other recent works on AD such as [153, 63, 194, 195]. For each experiment, over the 48000 HRRPs of our dataset a random fraction of 10% defines the test set.

Experimental setup

As was done in [153], we use a LeNet-type CNN with leaky ReLU activations for our Deep SVDD and Deep SAD neural network. The CAE architecture will be similar to allow the use of the trained weights of its encoder for the initialization of the Deep SVDD network (such initialization constitutes a pretraining). For these three neural network based AD, a batch size of 128 and a learning rate of 10^{-3} were used. The weight decay hyperparameter was set to 10^{-6}. The CAE used for AD and to provide Deep SVDD with pretrained parameters has been trained during ten epochs. For unsupervised AD, Deep SVDD was trained during 20 epochs whether it was initialized with the CAE encoder parameters or not. For SAD, the Deep SAD network adapted from Deep SVDD with a new objective was trained during 20 epochs, with a semi-supervised objective term balanced by η equal to one, as found in [152].

Regarding the hyperparameters of our shallow methods: LOF was set with a number of nearest neighbors of 48 to be considered in local densities estimations, and a contamination of 10%. RPD involves 1000 random projections to produce its statistical projection depth approximation, and OC-SVM had its ν set to 10^{-1}. Our IF was executed with 100 estimators, a contamination of 10% and a maximum number of samples per subsample of 1024, thus respecting the original spirit of IF, designed to work best with a substantially limited subsample [118]. The methods parameters were directly inspired by the available implementation of [153].

In the case of dimensionality reduction by PCA, the PCA is systematically preceded by min-max normalization and the number of components kept is chosen in order to retain 95% of the variance. This dimensionality reduction setup was inspired by the methodology proposed in [153]. It is to be noted that the PCA is always fitted using the training samples of the normal class only, this means that for each normal class change, there is a different PCA that is being used. The PCA used for dimensionality reduction of the test samples of a given class will, for instance, not be the same between an experiment where the same class is defined as normal, and another experiment for which another class will define normality and be responsible for the AD training. For Deep AD experiments, only a similar min-max normalization will be applied to the data.

Unsupervised AD with training set pollution

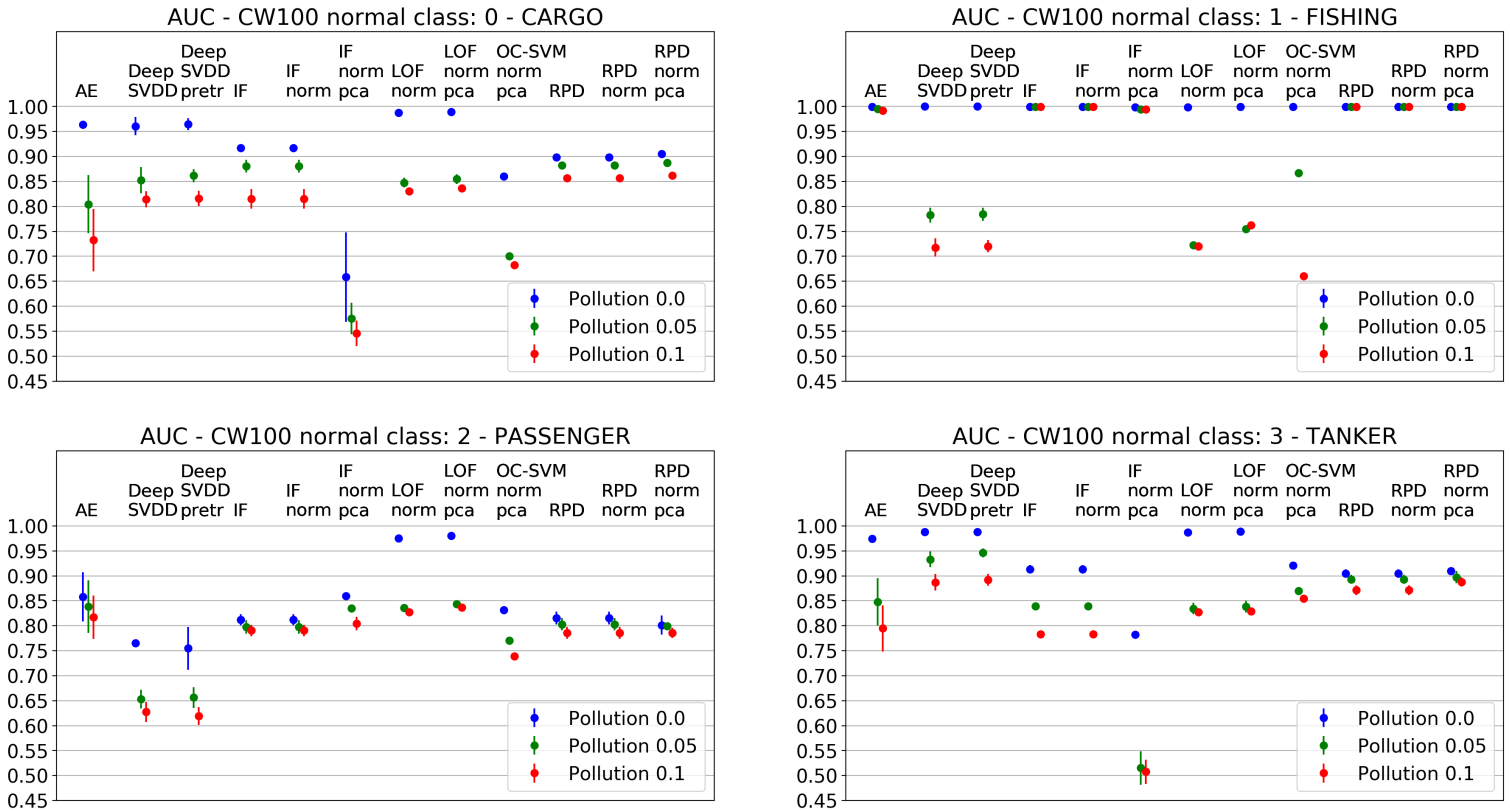


Figure 3.6: Unsupervised AD results. Each color point describes an AUC averaged over the experiments where a single pollution class is considered among the anomalous classes at a time. For example, a point associated with the normal cargos will be defined by the mean AUC of the experiments where pollution anomalies stem from the passenger ships, the fishing ships and the tankers respectively. Also, each single experiment setup is executed on three different seeds. This implies that each point actually depicts a mean of mean (over varying pollution and seeds respectively). The vertical bar associated with each point represents the mean standard deviation over all experiments and seeds (one standard deviation is computed over each set of seeds, then a mean is determined over all experiments). Each method suffers a loss of AUC for a higher pollution ratio of its training set supposedly pure of anomalous samples.

The results of unsupervised AD with training pollution are illustrated on Fig. 3.6. Results are globally good, with an expected impact of progressive pollution of the supposedly purely normal training set. Weaker detection is achieved when the normal class consists of passenger ships. This class likely makes it harder to guess a normality boundary since it is associated with an important features variance, and ships belonging to a wider range of lengths. This interpretation is compatible with the smaller drop of AUC for this normal class when pollution is introduced in the training set: the anomaly detection was already a little confused by the variety within the normality before any pollution. This reminds us how essential it is to wisely choose what the one-class can be made of when discriminating using OCC. The most sensible drops of performances can be seen for IF associated with normalization and PCA in preprocessing. A slight improvement can be observed when Deep SVDD benefits from a pretrained network thanks to an initialization that uses the weights of a CAE encoder trained on similar data. The most harmful pollution to normal classes cargo, passenger and tanker is the one made of fishing ships. Indeed, introducing anomalous fishing ships as pollution makes it harder to detect the most distinguishable class from normality in those cases.

IF and RPD stand out as the most stable AD through pollution (apart from IF with normalization and PCA), whereas the deep unsupervised AD methods and LOF indicate high sensibility to training set impurity, with an apparent plateau effect for LOF polluted experiments AUCs. LOF nonetheless obtains excellent results when the training set is not polluted. RPD also stands out as a shallow AD for which the normalization and PCA have no substantial impact on the AUC obtained. This stability illustrates the affine invariance properties of RPD. AUC is exceptionally high when the normal class is composed of fishing ships (class one) because of the easy distinction of this class among our three others: one could easily select most fishing ships in our dataset based on the active range-profile length only. The active range profile size is perhaps the most reliable feature of our HRRPs since the latter are subject to high variability within a single class, the dominant scatterers changing between ships with the same function (*i.e* the same AIS label), without even mentioning target-aspect sensitivity. Finally, regarding the resources needed to train the AD methods considered, this study revealed that outside the neural networks training that requires the biggest resources, LOF was the slowest, followed by OC-SVM (even though helped by a PCA-reduced dimensionality) and then IF. RPD was the quickest method to train in our study. Note that RPD training amounts to setting the location and spread estimators of RPO using the training data.

Semi-supervised AD

The results of semi-supervised AD with labeled samples during training are illustrated on Figure 3.7. For normal classes cargo and passenger, the introduction of labeled anomalies improves the mean AUC. One can further note that where unsupervised AD encountered difficulties for normal class passenger, SAD seems to tackle the latter. The mean AUC however drops with labeled anomalies for normal class tanker. Upon investigation, it emerged that a single SAD experiment (*i.e.* a single type a labeled anomaly) is responsible for this drop: once labeled anomalies from class cargo are added to the training data when the normal class is tankers with a non zero ratio, the AUC drops. It seems not to stem only from excessive proximity of the two classes in terms of ship size otherwise the same drop would impact the AUC when the roles are reversed (normal class defined by cargo and contaminated by tanker). The reason behind this asymmetrical decrease could be a combination of the untreated HRRPs sensitivities and the intra-class ships diversity. Going back to the successful SAD experiments, the rise of AUC due to labeled anomalies for normal class cargo and normal class passenger ships appears to reach a plateau: 1% and 5% of labeled anomalies help approximately as much

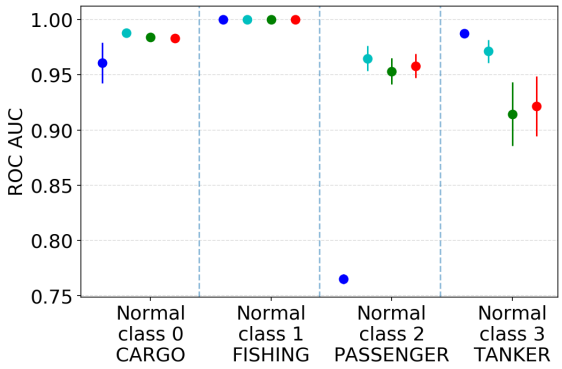


Figure 3.7: Semi-supervised AD results using Deep SAD, the semi-supervised adaptation of Deep SVDD. AUC averaged over three experiments, with three seeds per experiment as for unsupervised AD: in each experiment we change the anomaly class from which labeled anomalies originate. Vertical bars represent standard deviations computed over the various experiments and seeds. The colors describe the four different labeled anomalies ratios in the training data considered (blue: 0%, cyan: 1%, green: 5%, red: 10%).

as 10% labeled anomalies. A possible interpretation would be that few samples suffice to clarify the decision boundary of the AD method towards one specific kind of anomaly, with no further improvement in that so-called anomaly direction after a certain point. It is irrelevant to explore higher ratios of labeled anomalies within the training set, since doing so would switch our AD context with a supervised classification one.

Concluding remarks

This study shows that a variety of anomaly detection methods can be effective for unusual HRRP targets detection. Our results on semi-supervised detection demonstrate the possibility to improve such detection using a few labeled anomalies. Besides, since hyperparameters were not extensively fine-tuned the methods could yield additional improvements. This potential is also increased by the various advantages and drawbacks offered by each method considered.

3.3.2 OCC on images

The experiments presented in this section stem from our 2021 ICML UDL workshop paper [25] which applied OCC to the now classic image classification datasets of the deep learning community: MNIST [109], Fashion-MNIST [200], and CIFAR10 [105]. Additional experiments were conducted on the Statlog (Landsat Satellite) dataset from the UCI machine learning repository[16] [64]. The satellite data was considered to make our comparison of OCC methods more diverse for the data part, thus making the comparison more robust, and to make sure to consider a realistic discrimination task.

All experiments in 3.3.2 were conducted using PyTorch, on either of the following hardware configurations: AMD Ryzen 7 2700X with Nvidia RTX 2080, or Intel Xeon E5-2640 with Nvidia GTX Titan X. Table 3.1 reports the main results of this work. RPO stands for the original RPO, described in Eq. (3.2) with its location estimator and spread measure, respectively the median and MAD, defined on the training dataset completely made of normal samples. This means RPO is adapted to a machine learning data paradigm, whereas the original RPO was meant to directly be applied to a test set in which there would not be a significant proportion of anomalies. The direct application

[16] `https://archive.ics.uci.edu/ml/datasets/Statlog+\%28Landsat+Satellite\%29` Accessed: 01-09-2022

of RPO to our test sets without determining the medians and MADs on the training data leads to performances next to randomness. Such unsupervised and untrained RPOs are therefore not represented in the results tables. This poor performance is due both to the inadequate balance between samples considered as anomalies and the normal ones, and the potentially insufficient number of RPs with respect to the input space. Indeed, the more the input space to which RPs are applied to is of high dimensionality, the more RPs you need to obtain an informative projected estimator [80]. Most of the failure can however be attributed to the data balance of the test sets in this case. Apart from the results dedicated to the study of the influence of the number of RPs used in the latent RPO for deep RPO in table 3.3 and table 3.4, RPO is implemented using 1000 RPs.

In table 3.1, RPO-max is the closest AD to the original RPO but as previously stated it is beforehand adapted to take into account training data. RPO-mean is the shallow equivalent of the proposed method, deep RPO-mean, which adds an encoding neural network in front of RPO in the AD process. The same goes for RPO-max and deep RPO-max, which constitutes a more direct descendant of the original RPO. The random projections tensor is initialized by a random realization of a standard normal distribution. Random projections leading to a single projected dimension are normalized, so that they belong to the unit sphere in accordance with (3.2).

The input dimensionality for the shallow methods RPO-max and RPO-mean in table 3.1 is the dimensionality of the flattened input images, i.e. 784 for MNIST and Fashion-MNIST, and 3072 for CIFAR10. Deep SVDD and deep RPO encode the input images into latent representations of 32 and 128 dimensions, for MNIST, Fashion-MNIST and CIFAR10 respectively, before projecting using RPs when RPs are used. Hyperparameters were directly inspired by the ones used by deep SVDD authors since their method constitutes the baseline to which the proposed method is compared. In particular, the encoding networks architectures are the ones used for MNIST, Fashion-MNIST and CIFAR10 for the original deep SVDD [153] and deep SAD [152]. The weight decay hyperparameter was kept at 10^{-6}, even though for deep RPO it did not have a great impact in our experiments when compared with trials where the decay had been removed.

The metric used to evaluate the AD methods is the average AUC over several seeds, associated with a standard deviation, as can be found in the AD literature. One should keep in mind that when the number of classes defining normality increases, the datasets classes balance change. Before training, a validation set, made of 10% of the original training set, is created using scikit-learn common split function. For all deep experiments, the retained test AUC is the one associated with the best epoch observed for the validation AUC as was done in [52]. AUCs reach either a convergence plateau or a maximum before dropping in 50 epochs, this number of epochs was thus chosen for all the experiments. This represents a substantial difference with the experimental setup proposed in [153], where models were trained with many more epochs, benefited from pre-training accomplished using an auto-encoder, and a tailored preprocessing. The comparison between deep SVDD and deep RPO remains fair in this work since the network architecture is shared, along with the training hyperparameters. For each experiment, a new seed is set and a random pick of normal classes is performed. This means that, unlike many other papers in the literature, the nature of normality can change every time a new seed is adopted. This additional diversity behind the average AUCs presented explains the high standard deviations observed in the results.

# modes	RPO-max (1)	RPO-mean	Deep SVDD	Deep RPO-max	Deep RPO-mean (2)	(2)-(1)
MNIST - 1	84.64 ±6.73	84.12 ±6.74	88.60 ±4.62	87.96 ±5.31	**90.10** ±4.10	5.46
2	75.27 ±8.68	72.83 ±9.42	84.35 ±6.57	83.79 ±6.97	**85.36** ±6.48	10.09
3	69.67 ±9.65	66.92 ±10.25	81.23 ±6.76	80.16 ±7.12	**81.60** ±7.00	11.93
4	66.54 ±9.20	63.60 ±10.31	**78.89** ±6.56	77.35 ±6.92	78.65 ±7.05	12.11
F-MNIST - 1	89.19 ±5.81	89.73 ±5.79	90.45 ±5.76	90.17 ±6.09	**91.13** ±5.20	1.94
2	78.52 ±8.39	76.47 ±8.38	85.24 ±6.45	84.57 ±7.01	**85.81** ±6.36	7.29
3	71.06 ±7.38	69.37 ±7.64	80.30 ±6.99	80.64 ±6.69	**81.28** ±6.40	10.22
4	67.58 ±5.89	65.79 ±6.55	77.30 ±4.99	77.53 ±5.07	**77.82** ±5.34	10.24
CIFAR10 - 1	57.62 ±10.96	58.62 ±9.43	**64.15** ±7.38	60.22 ±7.00	63.14 ±7.30	5.52
2	53.85 ±9.49	53.81 ±7.61	56.37 ±9.25	55.66 ±8.54	**56.46** ±8.89	2.61
3	52.20 ±6.95	52.53 ±5.08	54.16 ±6.94	53.87 ±6.20	**54.30** ±6.80	2.10
4	51.88 ±5.91	52.32 ±4.97	53.64 ±5.97	53.71 ±5.78	**53.88** ±5.89	2.00

Table 3.1: The integrator estimator choice: mean versus maximum. RPO, deep RPO and deep SVDD test AUCs on MNIST, Fashion-MNIST and CIFAR10 for one to four modes considered as normal, for 30 seeds (truncated mean AUC ± std). The best AUC per number of modes and dataset is indicated in bold.

# MODES	1	3
MAX - 1D RPs	90.17 ±6.09	80.63 ±6.68
MAX - 2D RPs	89.40 ±6.43	79.63 ±7.08
MAX - 4D RPs	89.47 ±6.45	79.60 ±7.06
MEAN - 1D RPs	**91.13** ±5.20	**81.28** ±6.40
MEAN - 2D RPs	90.36 ±5.79	80.44 ±6.65
MEAN - 4D RPs	90.24 ±5.86	80.44 ±6.60

Table 3.2: Deep RPO test AUCs with varying RP latent dimensionality for the two estimators studied on Fashion-MNIST for 30 seeds (truncated mean AUC ± std). The best AUC per number of modes is indicated in bold.

The results of experiments over the three datasets considered, with 30 seeds per experimental setup, are gathered in table 3.1. The latter demonstrates the superiority of the *mean* over the *max* as an estimator for RPO when working with the deep RPO setup. The shallow RPO setup, on the other hand, suggests better performances can be obtained using a *max*. The neural network thus favors a loss balanced over all the single projected outlyingnesses. Moreover, the increasing AUC gap between deep RPO and shallow RPO ADs for MNIST and Fashion-MNIST supports the hypothesis that the encoding neural network allows RPO to face multimodal normality in AD. The growing gap in the last column is not observed for CIFAR10, however this failure is likely to stem from the excessive difficulty of the AD task rather than from an inability of deep RPO. The better performance of deep RPO-mean compared to deep SVDD placed the proposed method at the state-of-the-art level at the time of the workshop paper [25] publication.

The other experiments conducted mostly rely on two Fashion-MNIST setups, where the normality is defined by either one or three classes, as can be seen in tables 3.2, 3.3, 3.5, 3.6 and 3.7. CIFAR-10 was not selected, except to check whether the poor performances associated with this dataset stem from an inadequate number of RPs in table 3.4, because the multimodal AD remains an excessively complex task for the methods considered as table 3.1 points out, while MNIST does not carry multiscale structure, making it a less interesting example [151].

Dimensionality and number of random projections

The results of Table 3.2 indicate that no improvement was achieved through the increase of the RP dimensionality, the best score being associated with RPs projecting output representations stemming from the neural network on a single dimension. These scores also support the idea that Deep RPO works better with a *mean* estimator to integrate single RP outlyingness scores over all RPs.

Results in Table 3.3 indicate an adequate number of RPs was chosen to implement the RP outlyingness for the encoded data. A slight AUC increase has been achieved by decreasing the number of random projections shared by the rest of the experiments, i.e. 1000, possibly indicating the approximate minimum number of RPs necessary to handle the dimensionality of the neural network encoded latent space. One can think that using the minimum number of RPs to handle the RPO score input dimensionality in a deep setup constitutes a sensible strategy since it avoids superfluous parameters without hurting the outlyingness measure.

As announced, Table 3.4 suggests that the poor performances observed on CIFAR10 do not stem from an insufficient number of random projections, eventhough the greater latent dimensionality used for this dataset encoding could be expected to call for ad-

# MODES	1	3
100 RPs	90.25 ±5.18	81.70 ±6.73
500 RPs	**90.46** ±5.21	**81.96** ±6.67
1000 RPs	90.30 ±5.25	81.67 ±6.87
2000 RPs	90.42 ±5.19	81.83 ±6.83

Table 3.3: Deep RPO-mean test AUCs with varying number of RPs for the latent space RPO on Fashion-MNIST for 20 seeds (truncated mean AUC ± std).

# MODES	1000 RPs	3000 RPs
1	63.14 ±7.30	63.18 ±7.49
2	56.46 ±8.89	56.44 ±8.92
3	54.30 ±6.80	54.32 ±6.74
4	53.88 ±5.89	54.00 ±5.98

Table 3.4: Deep RPO-mean on CIFAR10 for 30 seeds, with either 1000 or 3000 RPs for RPO (truncated mean AUC ± std). The data being more complex, more RPs were used to verify whether a simple increase in the number of RPs could lead to better performances, without success.

ditional model complexity. These results emphasize the difficulty of the learning task considered when it comes to more realistic multimodal data.

Projections and components dropouts

Picking up the previously introduced notation in sections 3.1.1 and 3.1.3 regarding random projections, d is the data samples dimensionality, m the random projections output dimensionality, and p the number of random projections. Two types of dropouts [171] can be introduced on the random projections leading to the encoding network training loss: a dropout on the projections themselves, and a dropout on the components of the projections. In the first case, the dropout removes entire projections, implying a selection, in accordance with the dropout rate, over the p-dimensional channel of the projecting random tensor. Components dropout implies a selection, with its own dropout rate, along the d-dimensional channel. The indexes selected for this dropout will then cancel the corresponding dimensions in the random projections, thus ignoring as many components among the inputs. The RPs are normalized again after the components dropout, when $m = 1$, in accordance with Eq. (3.2). To respect the notation introduced, applications on images would flatten the input pixels array into a d-dimensional vector before their projection. For Deep RPO there is no need to flatten matrices as the output of the neural network has a unique dimension. As an intuitive example, in the specific case $m = 1$ which coincides with the original RPO, the random projections define a matrix $d \times p$: the projections dropout here would remove columns over the second dimension, whereas the components dropout would discard rows over the first dimension. No rescaling is operated under the proposed dropout mechanism, which differs from the original dropout implementation [171].

Table 3.5 indicates there is no actual AUC increase when harnessing either of the dropouts put forward for the random projections leading to the outlyingness measure. Since no substantial performances improvement was reached using the dropouts individually, their combined effects were not studied.

# MODES	1	3
NO DROPOUT	89.00 ±3.71	**78.71** ±4.80
C = 0.1	**89.19** ±3.57	78.64 ±4.85
C = 0.3	89.18 ±3.58	78.64 ±4.85
C = 0.5	**89.19** ±3.57	78.64 ±4.85
P = 0.1	89.05 ±3.68	78.51 ±4.93
P = 0.3	88.88 ±3.80	78.36 ±4.97
P = 0.5	88.67 ±3.97	78.43 ±4.76

Table 3.5: Deep RPO-mean test AUCs with and without components and projections dropouts on Fashion-MNIST for 10 seeds (truncated mean AUC ± std). C. is components dropout rate, P. is projections dropout rate. The best AUC per number of modes is indicated in bold.

SAD METHOD	SAD RATIO	1	3
DEEP SAD	0.00	87.70 ±5.30	78.30 ±5.02
DEEP SAD	0.01	88.08 ±5.03	83.49 ±4.71
DEEP SAD	0.10	90.37 ±4.00	84.54 ±4.87
DEEP RP-SAD	0.00	89.00 ±3.71	78.71 ±4.80
DEEP RP-SAD	0.01	89.19 ±3.60	78.76 ±4.90
DEEP RP-SAD	0.10	89.40 ±3.46	79.93 ±5.30

Table 3.6: Semi-supervised anomaly detection with distance inversion as in deep SAD for deep RPO to take into account rare labeled anomalies during training. The SAD ratio denotes the percentage of the training set composed of labeled anomalies. Two anomalous classes are randomly picked for each seed to provide the labeled anomalies. Experiments conducted with either one or three modes in the normality on Fashion-MNIST for 10 seeds (truncated mean AUC ± std).

Deep RPO for SAD

As introduced in 3.1.2 and in 3.1.3, both Deep SVDD and Deep RPO can integrate the additional supervision of labeled out-of-distribution samples during training to refine the boundaries of the latent normality. In Table 3.6 Deep SVDD, transformed into Deep SAD, appears to more significantly benefit from the additional information provided by a small minority of labeled anomalies during the training. Nevertheless Deep RPO also takes advantage of the latter to improve detection performances, confirming the generality of the distance inversion method to allow a location estimator based unsupervised AD to achieve SAD.

Stability against affine transformation

An affine transformation, defined as a constant multiplication of every component of the input representation of the samples, is applied to challenge the affine stability of the AD methods performances once the training is over. This affine transformation, defined by the constant α shown in the upper part of Table 3.7, breaks the normalization of the inputs features before their presentation to the neural network first layer. The experiments results suggest that deep RPO and deep SVDD are comparably stable with respect to the input transformation considered, and that such transformation does not trigger a drop in AUC. In addition, one can notice that the average test AUC slightly increases in some cases with the affine data disturbance. The lower part of Table 3.7 reports the results where instead of a constant diagonal matrix applying α to each

AD METHOD	α	1	AUC GAP	3	AUC GAP
DEEP SVDD	0.80	87.02 ±5.56	-0.70 ±1.05	76.49 ±5.67	-1.81 ±0.86
DEEP SVDD	0.90	87.77 ±5.24	+0.03 ±0.29	78.02 ±5.15	-0.29 ±0.29
DEEP SVDD	0.95	87.83 ±5.22	+0.09 ±0.14	78.31 ±5.02	-0.00 ±0.16
DEEP SVDD	1.00	87.73 ±5.24	±0.00 ±0.00	78.31 ±5.02	±0.00 ±0.00
DEEP SVDD	1.05	87.48 ±5.30	-0.25 ±0.20	78.05 ±5.18	-0.25 ±0.28
DEEP SVDD	1.10	87.09 ±5.40	-0.63 ±0.50	77.55 ±5.52	-0.76 ±0.75
DEEP SVDD	1.20	86.01 ±5.71	-1.71 ±1.32	75.98 ±6.65	-2.32 ±2.13
DEEP RPO	0.80	88.53 ±4.15	-0.72 ±1.48	76.85 ±5.28	-1.77 ±1.22
DEEP RPO	0.90	89.25 ±3.65	-0.00 ±0.59	78.33 ±4.85	-0.29 ±0.57
DEEP RPO	0.95	89.33 ±3.56	+0.07 ±0.31	78.61 ±4.79	-0.01 ±0.29
DEEP RPO	1.00	89.26 ±3.54	±0.00 ±0.00	78.63 ±4.85	±0.00 ±0.00
DEEP RPO	1.05	89.01 ±3.60	-0.24 ±0.38	78.38 ±5.05	-0.24 ±0.34
DEEP RPO	1.10	88.62 ±3.76	-0.63 ±0.84	77.91 ±5.36	-0.71 ±0.74
DEEP RPO	1.20	87.48 ±4.39	-1.77 ±1.95	76.49 ±6.25	-2.13 ±1.74
DEEP SVDD	$U_{[0.9;1.1]}$	87.47 ±5.10		78.15 ±4.76	
DEEP SVDD	$U_{[0.8;1.2]}$	86.70 ±5.52		77.52 ±5.01	
DEEP SVDD	$U_{[0.6;1.4]}$	83.86 ±7.21		75.67 ±5.94	
DEEP SVDD	$U_{[0.5;1.5]}$	81.28 ±8.74		73.64 ±7.10	
DEEP SVDD	$N(0,1)$	52.20 ±16.01		50.56 ±12.37	
DEEP RPO	$U_{[0.9;1.1]}$	88.76 ±3.61		78.54 ±4.70	
DEEP RPO	$U_{[0.8;1.2]}$	87.99 ±3.99		77.97 ±5.03	
DEEP RPO	$U_{[0.6;1.4]}$	85.12 ±5.73		76.32 ±6.05	
DEEP RPO	$U_{[0.5;1.5]}$	82.49 ±7.35		74.59 ±7.19	
DEEP RPO	$N(0,1)$	52.02 ±16.16		50.05 ±10.77	

Table 3.7: Deep RPO-mean test AUCs with varying affine transformation coefficient α on Fashion-MNIST for 10 seeds (truncated mean AUC ± std). The AUC gap is the mean AUC error, computed over all seeds, with respect to the AUC obtained when $\alpha = 1$, i.e. the baseline case. In the first part of the table, α denotes the constant value along the affine transformation diagonal matrix. In the second part, the diagonal elements are randomly generated according to either a uniform or a gaussian standard distribution. No AUC gap is computed since the seed by seed comparison with the baseline AUC would be unfair, a mean AUC test being computed over 20 random picks of the diagonal matrix for each seed.

input component, another diagonal matrix is used for which the diagonal coefficients are generated using either a random uniform or a standard gaussian distribution. Again, deep RPO and deep SVDD show comparable stability when confronted with the more distorting affine transformations. Looking at the standard deviations overall, deep RPO seems slightly more stable.

Additional experiments on tabular data

Since AD on MNIST, Fashion MNIST and CIFAR10 is very common and excellent performances have already been obtained on these datasets using self-supervised learning, we compare the highlighted shallow and deep methods of our main results in Table 3.1 on less common tabular data. As can be seen in Table 3.8, Deep SVDD remains our baseline. A satellite dataset is chosen [17]. The data stems from the original Statlog

[17] `http://odds.cs.stonybrook.edu/satellite-dataset/` Accessed: 28/10/2022

Method	mean test AUC ± std
Deep SVDD	68.23 ±5.53
RPO-Max	64.89 ±2.67
Deep RPO-Mean	**73.01** ±5.93

Table 3.8: Deep RPO-Mean, RPO-Max and the baseline deep SVDD on the satellite dataset for 20 seeds (truncated mean AUC ± std). A more complete max versus mean comparison for both RPO and Deep RPO can be found in Table 3.1.

(Landsat Satellite) dataset from UCI machine learning repository [18] [64], where the smallest three classes are combined to form the outlier class, while the other classes define the inlier class. As for the previous experiments, deep SVDD and deep RPO-Mean share the same neural network architecture and training hyperparameters, to produce a fair comparison. The improvement provided by Deep RPO-Mean is confirmed. The number of RPs used in the latent RPO was set to 500, since the output dimensionality of 8 of the neural network is significantly lower. This in turn is due to the low input data dimensionality for the neural network, the input samples being 1D vectors defined by 36 values. The neural networks were always trained for 80 epochs, and the test AUC retained as the model performance for each seed is the one associated with the best epoch with respect to the validation set AUC. The results also put forward the contribution of the trainable neural network projecting data samples, deep SVDD performing better then the shallow method RPO-Max. Finally, the standard deviation of the performances appears to be higher for deep methods.

Concluding remarks

These experiments emphasize the possibility of adapting simple abnormality measures to complex and realistic anomaly detection tasks in which normality is multimodal. The experiments conducted on MNIST, Fashion-MNIST, CIFAR10 and satellite data show a light improvement in performance with respect to Deep SVDD when using Deep RPO and suggest that the task of anomaly detection in a fully unsupervised framework, in the case of multimodal normality, remains a challenge. The relative success of the proposed approach highlights the relevance of random projections and more generally of untrained transformations in neural networks, when they are associated with a well chosen trainable architecture.

3.3.3 OCC on radar spectrums and covariance matrices

The experiments presented in this section stem from our 2022 ECML PKDD paper [26] which applied OCC to Doppler signatures. The latter details a comparison of deep and non-deep OODD methods on simulated low-resolution pulse radar micro-Doppler signatures, considering both a spectral and a covariance matrix input representation. The covariance representation aims at estimating whether dedicated second-order processing is appropriate to discriminate signatures. Pulse Doppler Radar (see Chapter 1) signatures are generated by a MATLAB [125] simulation [19]. The Doppler signatures are a series of periodograms, i.e. the evolution of spectral density over several bursts, one periodogram being computed per burst. Both periodograms and the series of periodograms can be called spectrum in the remainder of this paper. The samples on which

[18] `https://archive.ics.uci.edu/ml/datasets/Statlog+\%28Landsat+Satellite\%29` Accessed: 01/04/2021

[19] `https://github.com/Blupblupblup/Doppler-Signatures-Generation` Accessed: 28/10/2022

the discrete Fourier transform is computed are sampled at the PRF frequency, i.e. one sample is available per pulse return for each range bin.

This work compares deep and non-deep OODD methods, including second-order methods harnessing the SPD representations provided by the sampled covariance matrix of the signatures. The extension of the deep learning architectures discussed to SAD and self-supervised learning (SSL) is part of the comparison. The use of SSL here consists in the exploitation of a rotated version of every training signature belonging to the normal class in addition to its non-rotated version, whereas SAD amounts to the use of a small minority of actual anomalies taken in one of the other classes of the dataset. In the first case one creates artificial anomalous samples from the already available samples of a single normal class, whereas in the second case labeled anomalies stemming from real target classes are made available. To avoid confusion, one should note that this single normal class can be made of one or several target classes, which end up being considered as a unique normality. No SSL or SAD experiments were conducted on the SPD representations, since the SSL and SAD extensions of the deep methods are achieved through training loss modifications, and the SPD representations were confined to shallow baselines. In the previously described setup, SSL is nothing more than SAD with artificial data points provided by SSL transformations.

Doppler signatures generation

The main parameters of the simulation are close to realistic radar and target characteristics. A carrier frequency of 5 GHz was selected, with a PRF of 50 KHz. An input sample is a Doppler signature extracted from 64 bursts of 64 pulses, i.e. 64 spectrums of 64 points, ensuring the full rank of the covariance matrix computed over non-normalized Doppler, i.e. Fourier, bins. The only simulation parameter changing across the classes of helicopter-like targets is the number of rotating blades: Doppler signatures are associated with either one, two, four or six rotating blades, as can be found on drones and radio-controlled helicopters. The quality of the dataset is visually verified: a non-expert human is easily able to distinguish the four target classes, confirming the discrimination task is feasible. The classes intrinsic diversity is ensured by receiver noise, blade size and revolutions-per-minute (RPM) respectively uniformly sampled in $[4.5, 7]$ and $[450, 650]$, and a bulk speed uniformly sampled so that the signature central frequency changes while staying approximately centered. The possible bulk speeds and rotor speeds are chosen in order for the main Doppler shift and the associated modulations to remain in the unambiguous speeds covered by the Doppler signatures [114]. Example signatures and their covariance matrix representations are depicted for each class on Fig. 3.8. For each class, 3000 samples are simulated, thus creating a 12000-samples dataset. While small for the deep learning community, possessing thousands of relevant and labeled real radar detections would not be trivial in the radar industry, making larger simulated datasets less realistic for this use case.

Preprocessing This work is inspired by [153], which experimented on MNIST, a dataset in which samples are images of objects without background or irrelevant patterns. In order to guarantee a relevant neural architecture choice, this kind of input format is deliberately reproduced. The idea of creating MNIST-like benchmarks has been of interest in different scientific communities such as biomedical images [201] and 3D modeling [94]. The series of periodograms, i.e. non-SPD representations are therefore preprocessed such that only the columns with top 15% values in them are kept, this operation being done after a switch to logarithmic scale. This results in periodograms where only the active Doppler bins, portraying target bulk speed and micro-Doppler modulations, have non-zero value. Only a grayscale region of interest (ROI) remains in

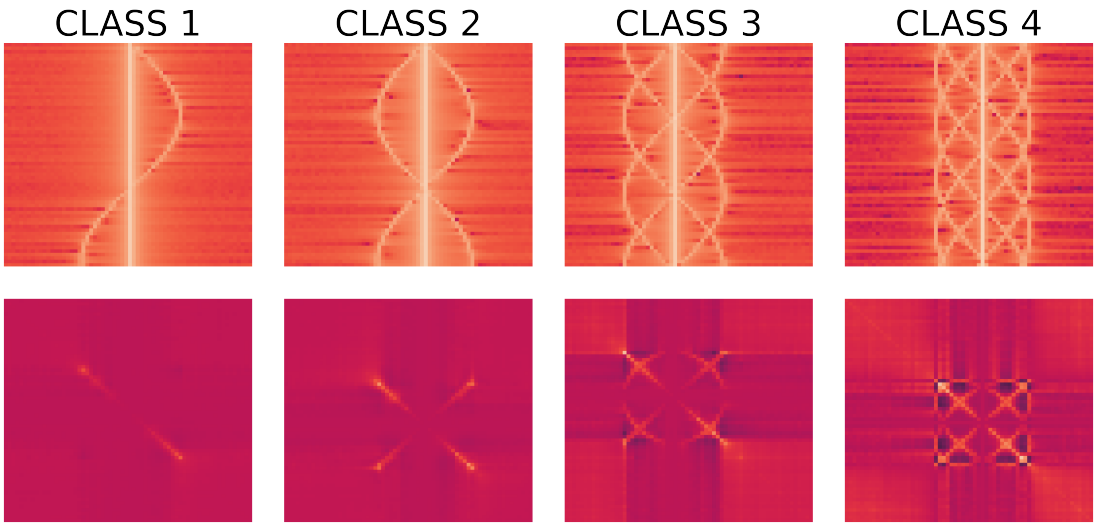


Figure 3.8: One sample of each target class: the varying number of rotating blades defines the classes, the modulation pattern being easily singled out. The first line of images shows Doppler signatures, i.e. the time-varying periodogram of targets over 64 bursts of 64 pulses. On those images, each row is the periodogram computed over one burst, and each column a Fourier i.e. a Doppler bin. The second line contains the covariance SPD representation of the first line samples. The width of the Doppler modulations around the bulk speed on the periodograms varies within each class, as well as the bulk speed, the latter being portrayed by the central vertical illumination of the signature.

the input matrix with various Doppler shifts and modulation widths, examples of which are shown on Fig. 3.9. This preprocessing leads to the "(SP)" input format as indicated in the results tables, and is complementary to the covariance representation. Covariance matrices are computed without such preprocessing, except for the switch to logarithmic scale which precedes the covariance computation. Comparing covariance-based OODD to OODD on spectral representations is fair since both representations stem from the same inputs, the covariance only implying an additional transformation of the input before training the AD. All input data is min-max normalized except for the covariance matrices used by tPCA.

Riemannian methods for covariance matrices

Two SPD-specific AD approaches were considered. The first approach consists in replacing the principal component analysis dimensionality reduction preceding shallow AD with an SPD manifold-aware tangent PCA (tPCA). As explained in 3.2.3, tPCA is a questionable choice when the objective set is AD with multimodal normality, something that is part of the experiments put forward in this work. Nonetheless, the Euclidean PCA being a common tool in the shallow AD literature, tPCA remains a relevant candidate for this study since it enables us to take a step back with respect to non-deep dimensionality reduction.

The second SPD-specific approach defines a Riemannian equivalent to Deep SVDD already mentioned in 3.2.3: inspired by recent work on SPD neural networks [121, 35, 89, 207], which learn intermediate representations while keeping them on the SPD matrices manifold, a Deep SVDD SPD would transform input covariance matrices and project the latter into a latent space comprised within the SPD manifold. Taking into account SAD

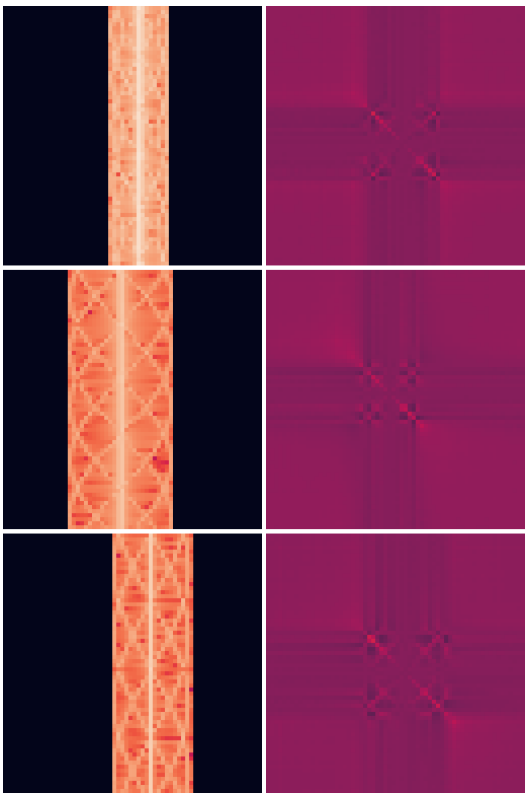


Figure 3.9: Random samples of the fourth class after the preprocessing erasing the irrelevant background, which makes the dataset closer to the MNIST data format. One can notice the varying modulation width of the target spectrum and central Doppler shift. The fourth class has the highest number of rotating blades on the helicopter-like target, hence the higher complexity of the pattern. These samples illustrate the input format of the various AD methods compared in this work, and are min-max normalized so that their values belong to $[0, 1]$. Only the inputs of the Riemannian setup where shallow learning AD is used on SPD inputs after tPCA uses a different input format, where the input covariance is not normalized.

and SSL labeled anomalies during training was expected to be done as for the semi and self-supervised adaptations of Deep SVDD described earlier, where labeled anomalies are pushed away from the latent normality centroid thanks to an inverse distance term in the loss. For Deep SVDD SPD, the distance would be a geometry-aware distance such as the Log-Euclidean distance. Despite diverse attempts to make such a Deep SVDD SPD model work, with and without geometry-aware non-linearities in the neural network architecture, no effective learning was achieved on our dataset. This second approach will therefore be missing from the reported experimental results. An example architecture of such an SPD-manifold aware Deep SVDD adaptation is still proposed in section C.2. Since this approach defined the ReEig [89] non-linearity rectifying small eigenvalues of SPD representations, the related shallow AD approach using the norm of the last PCA components as an anomaly score was also considered. This approach was applied to both spectral and covariance representations, with the PCA and tPCA last components respectively, but was discarded as well due to poor performances. The latter indicate that anomalous samples are close enough to the normal ones for their information to be carried in similar components, emphasizing the near OODD nature of the discrimination pursued.

Experiments

AD experiments are conducted for two setups: a first setup where normality is made of one target class, and a second setup where normality is made of two target classes. When a bimodal normality is experimented on, the normal classes are balanced. Moreover, the number of normal modes is not given in any way to the AD methods, making the experiments closer to the arbitrary and, to a certain extent, unspecified one-class classification useful to a radar operator. Within the simulated dataset, 90% of the samples are used to create the training set, while the rest is equally divided to create the validation and test sets. All non-deep AD methods are applied after a preliminary

dimensionality reduction, which is either PCA or tPCA. The number of RPs used to compute the outlyingness score with RPO and Deep RPO is the same and set to 1000, even though the estimator used differs between the shallow and the deep approach. All deep experiments were run on a single GPU, which was either a NVIDIA Tesla P100 or a NVIDIA RTX 2080. In both cases running one deep AD setup for ten seeds took approximately one hour. Non-deep experiments were CPU-intensive and also required around one hour for ten seeds on a high-end multi-core CPU.

Deep learning experiments The test AUC score of the best validation epoch in terms of AUC is retained, in line with [52]. All experiments were conducted with large 1000 samples batches, which stabilizes the evolution of the train, validation and test AUCs during training. The training is conducted during 300 epochs, the last 100 epochs being fine-tuning epochs with a reduced learning rate, a setup close to the one in [153]. As was suggested in [52] a relatively small learning rate of 10^{-4} is chosen to help avoid the latent normality hypersphere collapse, i.e. the convergence to a constant projection point in the latent space in the non-SAD and non-SSL cases, with $\lambda = 10^{-6}$. Such a latent normality collapse is made impossible when SAD or SSL samples are concentrated around dedicated centroids or scattered away from normality centroids, since the network is then trained to disperse representations. Loss terms integrating labeled anomalies for extra training supervision are balanced with $\eta = 1$, and for Deep MSVDD $\nu = 0.1$. Hyperparameters are kept constant across all experiments conducted, in order to ensure fair comparisons. In the results tables, the second and third columns indicate whether SAD and SSL samples were used for additional supervision during training, and describe how such samples affected the training loss if present. When the SAD or SSL loss term is defined by a centroid, it means that the distance to the mentioned centroid is minimized during training, whereas "away" implies the projection of the SAD or SSL samples are repelled from the normality centroid thanks to an inverse distance as described previously. For example, the first line of the second part of Table 3.10 describes an experiment where SAD samples are concentrated around the SAD samples latent centroid, and SSL samples concentrated around the SSL samples latent centroid. Centroids are computed, as for the normal training samples, with the averaging of an initial forward pass, therefore yielding the average latent representation.

Non-deep learning experiments Shallow AD conducted on the covariance representation after a common PCA uses the upper triangular part of the min-max normalized input as a starting point, avoiding redundant values. This contrasts with the Riemannian approach replacing PCA with the tPCA, the latter requiring the raw SPD representation. Furthermore, shallow approaches were also tried on the periodograms individually, where each row of an input signature, i.e. one vector of Doppler bins described for one burst, was given a score, the complete signature being then given the mean score of all its periodograms. This ensemble method did not yield relevant results and is therefore missing from our comparison. Such an approach ignores the order of periodograms in signatures.

Neural network architecture While the MNIST-like input format is thus replicated, the 2D features remain specific to radar signal processing and may therefore benefit from a different neural network architecture. Several neural networks architectures were considered, including architectures beginning with wider square and rectangular convolutions extended along the (vertical) bursts input axis, with none of the investigated architectures scoring systematically higher than the Fashion-MNIST architecture from the original Deep SAD work [152], which was only modified in order to handle

the larger input size. The latter was consequently selected to produce the presented results. This architecture projects data with two convolutional layers followed by two dense layers, each layer being separated from the next one by a batch normalization and a leaky ReLU activation. The outputs of the two convolutional layers are additionally passed through a 2D max-pooling layer.

Riemannian AD The tPCA was computed thanks to the dedicated Geomstats[20] [130] function, while experiments implementing a Riemannian equivalent of Deep SVDD were conducted using the SPD neural networks library torchspdnet[21] [36]. The AD experiments based on a SPD neural network ending up inconclusive, they are not part of the results tables.

[20] `https://geomstats.github.io/`
[21] `https://gitlab.lip6.fr/schwander/torchspdnet`

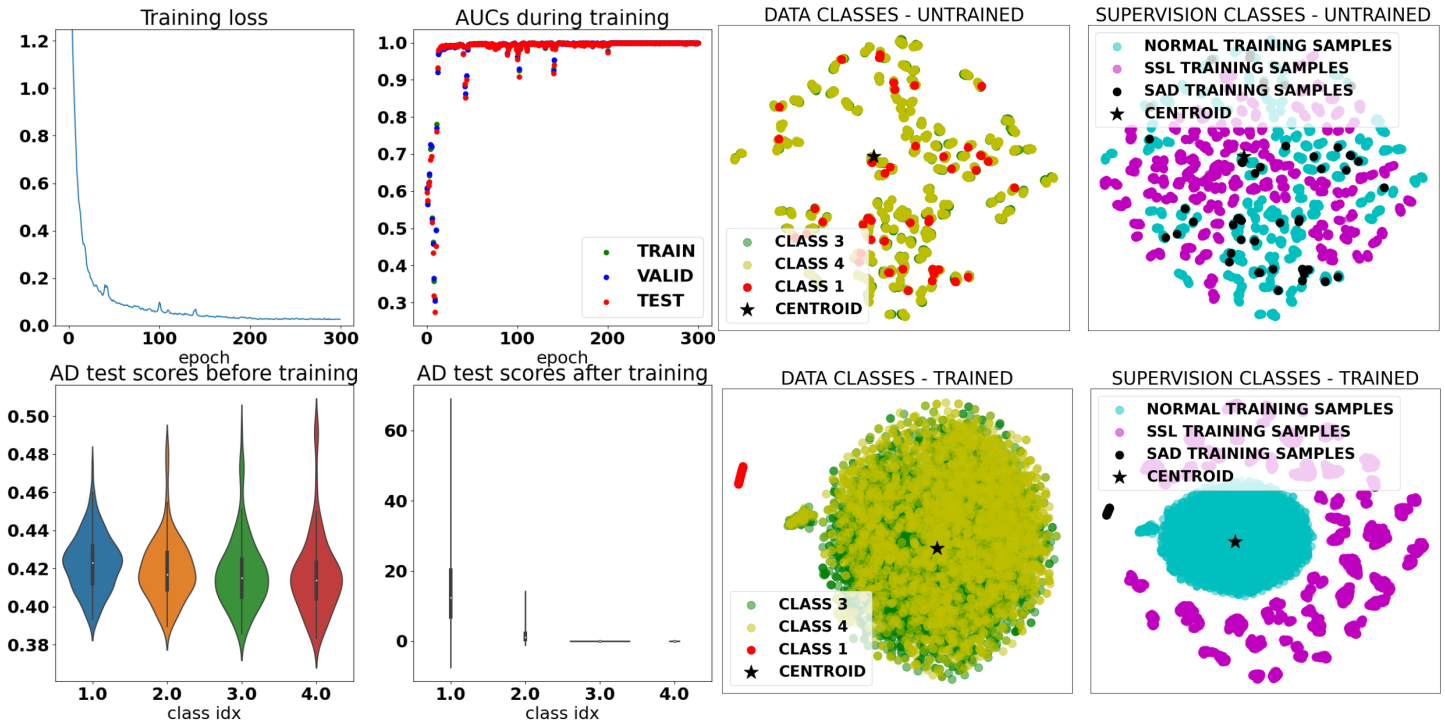


Figure 3.10: **Left** - Training metrics of a successful run where normal samples are concentrated around their average initial projection, and SAD and SSL samples are pushed away thanks to a loss term using the inverse of the distance with respect to the normality latent centroid. This is one of the most successful setups in Table 3.10, and one of the easiest AD experiments since the two classes defining normality here are class 3 (four blades are responsible for the modulation pattern around the central Doppler shift) and class 4 (six blades are responsible for the modulation pattern around the central Doppler shift), meaning the separation with the other classes deemed anomalous is actually a binary modulation complexity threshold. One of the contributions of the SAD and SSL supervisions can be observed on the evolution of AUCs during training: no AUC collapse can be seen during training, discarding the possibility of a latent distribution collapse during training. Experiments showed that large training batches contributed to stable AUCs growth. Spikes in the training loss match the drops in AUCs. **Right** - Latent distribution of the training samples visualized in 2D using t-SNE after projection by the untrained (top) and the trained neural network (bottom). One can notice that normal training samples from both normal classes are completely mixed up with the minority of SAD labeled anomalies from class 1 in red (one blade), semantically similar, whereas SSL samples which are rotated normal training samples are already gathered in their own latent subclusters. SAD labeled anomalies end up well separated after training.

Unsupervised AD results, for which the training is only supervised by normal training samples, are presented in Table 3.9. These results indicate the superiority of deep learning for the OODD task considered, while demonstrating the substantial contribution of geometry-aware dimensionality reduction through the use of tPCA for non-deep AD. RPO is kept in Table 3.9 even though it does not achieve useful discrimination because it is the shallow equivalent of Deep RPO, one of the highlighted deep AD methods, deprived of the neural network encoder and with a *max* estimator instead of a *mean*, as was previously justified. Deep MSVDD does not lead to the best performances, and is as effective as Deep SVDD and Deep RPO, which could seem surprising at least when normality is made of two target classes. Indeed, since Deep MSVDD has the possibility to use several disjointed hyperspheres to capture the latent normality distribution, one could expect it to better model more complex, e.g. multimodal, normality.

AD method (input format)	SAD loss	SSL loss	Mean test AUC (1 mode)	Mean test AUC (2 modes)	Equation
OC-SVM (SP-PCA)	/	/	49.16 ± 26.69	45.48 ± 27.53	(3.4)
OC-SVM (SPD-PCA)	/	/	64.68 ± 9.10	58.23 ± 15.12	(3.4)
OC-SVM (SPD-tPCA)	/	/	57.59 ± 3.91	55.33 ± 9.48	(3.4)
IF (SP-PCA)	/	/	50.96 ± 17.37	48.50 ± 18.76	(3.5)
IF (SPD-PCA)	/	/	52.36 ± 22.47	47.50 ± 20.32	(3.5)
IF (SPD-tPCA)	/	/	66.91 ± 9.65	61.23 ± 12.65	(3.5)
LOF (SP-PCA)	/	/	56.80 ± 2.38	61.55 ± 10.29	/
LOF (SPD-PCA)	/	/	66.44 ± 21.37	65.83 ± 19.52	/
LOF (SPD-tPCA)	/	/	78.38 ± 8.86	73.56 ± 10.09	/
RPO (SP-PCA)	/	/	49.61 ± 6.89	50.43 ± 7.13	(3.2)
RPO (SPD-PCA)	/	/	51.08 ± 19.66	54.95 ± 17.58	(3.2)
RPO (SPD-tPCA)	/	/	33.97 ± 7.36	38.08 ± 14.58	(3.2)
Deep SVDD (SP)	no SAD	no SSL	83.03 ± 6.83	**78.29 ± 6.68**	(3.9)
Deep MSVDD (SP)	no SAD	no SSL	82.27 ± 9.67	**78.30 ± 8.28**	(3.10)
Deep MSVDD "mean best" (SP)	no SAD	no SSL	82.29 ± 7.20	78.02 ± 6.80	(3.10)
Deep RPO (SP)	no SAD	no SSL	**83.60 ± 5.35**	78.13 ± 6.02	(3.14)

Table 3.9: Unsupervised AD experiments results (average test AUCs in % ± StdDevs over ten seeds). These machine learning methods are trained on fully normal training sets, without labeled anomalies for SAD or self-supervision transformations. The four last methods are our deep AD baselines, trained on normalized spectral representations only. Deep MSVDD "mean best" indicates the neural network was trained using a simpler loss, analogous to the Deep SVDD loss, where only the distance to the best latent normality centroid is minimized, thus discarding the radius loss term. PCA and tPCA indicate that the AD model is trained after an initial dimensionality reduction, which is either PCA or tangent PCA.

Potential contribution of SAD and SSL

The contribution of additional supervision during training through the introduction of SAD samples and SSL samples is examined in Table 3.10. Regarding SAD experiments, labeled anomalies will be taken from a single anomalous class for simplicity, and because only four classes are being separated, this avoids unrealistic experiments where labeled anomalies from every anomalous class are seen during training. When SAD samples are used during training, labeled anomalies represent one percent of the original training set size. This respects the spirit of SAD, for which labeled anomalies can only be a minority of training samples, which is not representative of anomalies. This is especially realistic in the radar processing setup initially described where labeled detections would rarely be available. SSL samples are generated thanks to a rotation of the spectral input format, rendering the latter absurd but encouraging better features extraction since the network is asked to separate similar patterns with different orientations. SSL samples are as numerous as normal training samples, implying they do not define a minority of labeled anomalies for training as SAD samples do, when they are taken into account.

Individually, SAD samples lead to better performances than SSL ones, but the best results are obtained when combining the two sets of samples for maximal training supervision. Deep SVDD appears to be substantially better at taking advantage of the additional supervision provided by SAD and SSL samples. Quite surprisingly for a radar operator, the best test AUC is obtained when SSL samples are concentrated around a specialized centroid while SAD samples are repelled from the normality centroid. Indeed, SSL samples being the only absurd samples considered in our experiments radarwise, it could seem more intuitive to project SAD samples, which remain valid targets, next to a dedicated centroid while repelling SSL samples. Likewise, on an ideal outlyingness scale, SSL samples should be further away from normality than SAD samples. This counter-intuitive performance could stem from the test set which only evaluates the separation of targets in a near OODD context. No invalid target representation, like the SSL samples are, is present in the test set, only valid representation from the four target classes make up the latter. This is consistent with the application put forward in this study: use OODD to discriminate between various kinds of radar detections.

AD method (input format)	SAD loss	SSL loss	Mean test AUC (1 mode)	Mean test AUC (2 modes)
Deep SVDD (SP)	no SAD	SSL c.	86.79 ± 6.54	83.91 ± 7.92
Deep RPO (SP)	no SAD	SSL c.	88.70 ± 5.10	84.59 ± 8.54
Deep SVDD (SP)	no SAD	away	81.43 ± 8.62	77.01 ± 8.20
Deep RPO (SP)	no SAD	away	80.21 ± 9.06	78.93 ± 9.39
Deep SVDD (SP)	SAD c.	no SSL	86.79 ± 8.94	87.65 ± 6.44
Deep RPO (SP)	SAD c.	no SSL	81.38 ± 6.09	76.45 ± 6.30
Deep SVDD (SP)	away	no SSL	93.93 ± 4.82	93.50 ± 7.61
Deep RPO (SP)	away	no SSL	84.19 ± 5.32	80.37 ± 7.22
Deep SVDD (SP)	SAD c.	SSL c.	91.00 ± 6.45	90.51 ± 7.38
Deep RPO (SP)	SAD c.	SSL c.	87.79 ± 5.81	82.69 ± 8.51
Deep SVDD (SP)	SAD c.	away	89.98 ± 7.79	91.03 ± 6.71
Deep RPO (SP)	SAD c.	away	78.86 ± 9.10	79.11 ± 9.64
Deep SVDD (SP)	away	SSL c.	**95.06 ± 4.20**	93.91 ± 7.31
Deep RPO (SP)	away	SSL c.	89.82 ± 5.21	87.17 ± 8.17
Deep SVDD (SP)	away	away	94.63 ± 4.31	**94.02 ± 7.30**
Deep RPO (SP)	away	away	90.91 ± 5.94	92.69 ± 7.98

Table 3.10: Experiments with additional supervision provided by SAD and/or SSL labeled samples during training (average test AUCs in % ± StdDevs over ten seeds). When available, SAD samples are the equivalent of one percent of the normal training samples in quantity. The first half of the Table reports performances where only one of the two kinds of additional supervision is leveraged, while the second half describes the performances for setups where both SAD and SSL labeled samples contribute to the model training. Each couple of lines compares Deep SVDD and Deep RPO in a shared AD supervision setup, thus allowing a direct comparison. *c.* stands for centroid.

Training with a contaminated training set

Unsupervised AD refers to the experiments of Table 3.9 where only training samples assumed to be normal supervise the training of the neural network. Real-life datasets, labeled by algorithms or experts, are unlikely to respect that assumption and will suffer from contamination of normal samples with unlabeled anomalies. The results in Table 3.11 depict how sensible the deep AD methods previously introduced are to training set contamination. The contamination is carried out using the one percent SAD samples already used for SAD experiments in Table 3.10. While in the SAD experiments SAD samples were repelled from the normality centroid or concentrated next to their dedicated latent reference point, here they will be processed as normal samples. SSL samples again appear to better contribute to improving AD when concentrated next to a specialized centroid, while the performance drop due to contamination does not seem to be particularly stronger for one of the approaches considered.

AD method (input format)	SAD loss	SSL loss	Mean test AUC (1 mode)	Mean test AUC (2 modes)
Deep SVDD (SP)	no SAD	no SSL	80.76 ± 7.11	76.02 ± 6.66
Deep MSVDD (SP)	no SAD	no SSL	78.31 ± 11.18	74.49 ± 9.13
Deep MSVDD "mean best" (SP)	no SAD	no SSL	79.84 ± 7.82	74.89 ± 7.01
Deep RPO (SP)	no SAD	no SSL	81.29 ± 5.92	74.82 ± 5.89
Deep SVDD (SP)	no SAD	SSL c.	85.34 ± 6.85	81.36 ± 7.47
Deep RPO (SP)	no SAD	SSL c.	**86.66 ± 6.41**	**82.78 ± 8.25**
Deep SVDD (SP)	no SAD	away	79.62 ± 9.02	75.38 ± 8.28
Deep RPO (SP)	no SAD	away	76.16 ± 9.87	76.56 ± 8.69

Table 3.11: Contamination experiments results (average test AUCs in % ± StdDevs over ten seeds): the SAD labeled anomalies are integrated within the training samples and taken into account as normal samples during training, thus no SAD loss term is used for SAD samples. The contamination rate is one percent, i.e. the equivalent of one percent of the normal training samples in labeled anomalies is added to confuse the AD. *c.* stands for centroid.

Mean vs max in Deep RPO training loss

In order to ensure the relevance of the replacement of the *max* estimator with a *mean* in Deep RPO for our application, which as previously explained in 3.1.3 removes the equivalence with a Mahalanobis distance when numerous RPs are used to estimate RPO, Deep RPO with *mean* and Deep RPO with *max* were compared for every deep experimental setup presented in the paper. The resulting performances confirmed the superiority of Deep RPO with *mean*. A few of these performances are presented in Table 3.12 and Table 3.13.

AD method (input format)	SAD loss	SSL loss	Mean test AUC (1 mode)	Mean test AUC (2 modes)
Deep RPO (SP) - mean	no SAD	no SSL	**83.60 ± 5.35**	**78.13 ± 6.02**
Deep RPO (SP) - max	no SAD	no SSL	74.85 ± 7.29	71.60 ± 6.57
Deep RPO (SP) - mean	away	away	**90.91 ± 5.94**	**92.69 ± 7.98**
Deep RPO (SP) - max	away	away	78.85 ± 9.40	75.53 ± 9.10

Table 3.12: Unsupervised and semi-supervised experiments where SAD and SSL samples provide additional supervision, with Deep RPO with *max* estimator to integrate over the RPs and Deep RPO with *mean* (average test AUCs in % ± StdDevs over ten seeds): the integration of 1D projected anomaly measures with a *mean* systematically leads to better performances. Note that in these Deep RPO experiments, RPO is applied to encoded inputs in the latent representation space of the neural network.

AD method (input format)	SAD loss	SSL loss	Mean test AUC (1 mode)	Mean test AUC (2 modes)
Deep RPO (SP) - mean	no SAD	no SSL	**81.29 ± 5.92**	**74.82 ± 5.89**
Deep RPO (SP) - max	no SAD	no SSL	74.21 ± 7.48	68.95 ± 5.09
Deep RPO (SP) - mean	no SAD	SSL c.	**86.66 ± 6.41**	**82.78 ± 8.25**
Deep RPO (SP) - max	no SAD	SSL c.	82.52 ± 7.43	77.29 ± 8.45
Deep RPO (SP) - mean	no SAD	away	76.16 ± 9.87	**76.56 ± 8.69**
Deep RPO (SP) - max	no SAD	away	**76.20 ± 9.37**	73.41 ± 9.61

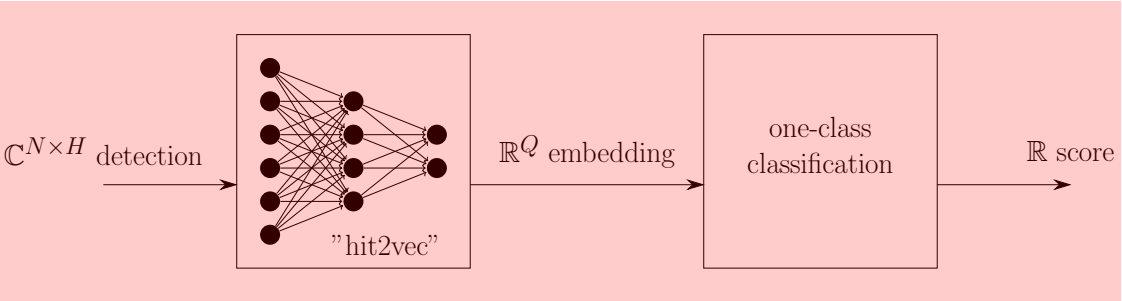
Table 3.13: Contamination experiments with Deep RPO with *max* estimator to integrate over the RPs and Deep RPO with *mean* (average test AUCs in % ± StdDevs over ten seeds): except in one case the integration of 1D projected anomaly measures with a *mean* leads to better performances. Note that in these Deep RPO experiments, RPO is applied to encoded inputs in the latent representation space of the neural network.

Concluding remarks

The near OODD performances of various deep and non-deep, unsupervised and semi-supervised AD methods were compared on a radar Doppler signatures simulated dataset. Deep AD approaches were evaluated in various supervision setups, which revealed the relevance of combining a minority of labeled anomalies with transformed normal training samples to improve semi-supervised near OODD performances, and avoid latent normality distribution collapse. The benefits of deep learning clearly showed, and while not leading to the best overall performances, geometry-aware processing with tangent PCA proved to be the source of a substantial improvement for non-deep AD.

Chapter 4

One-class classification for encoded hits



Chapter 4

This chapter will present preliminary results regarding the encoding and discrimination of radar hits enriched under the form of neighborhoods of range cells, a format introduced in chapter 1. It thus allies the hit format enrichment proposed in chapter 1, with the single and neighborhood of range cells encoding architectures of chapter 2 and the one-class classification methods presented in chapter 3. The combination of these processing steps create a filter answering the task motivating the works of this thesis and oriented towards the discrimination of small and slow targets by air surveillance radars. The pipeline presented here is therefore designed to face a varying input representation in terms of signal length and resolution, and a lack of supervision for the final discrimination.

4.1 Experiments protocol and data

The dataset used to evaluate the proposed representation learning over range cells neighborhoods is built with the single range cells embeddings provided by the FCN-based cell2vec approach presented in section 2.5. The interaction of the latter approach and the graph2vec method evaluated here is illustrated on Fig. 2.4. The dataset is created with the testing set embeddings of the single range cells encoding experiment, thus completely discarding the initial training set single cell embeddings. The single range cells embeddings of the target classes defined in section 2.5 are combined into four different classes of varying correlations to define a neighborhood of $H = 5$ range cells. The four neighborhood classes follow the following neighborhood correlation patterns $XXXXX$, $YXXXY$, $YYXZZ$ and $WYXZW$. In the latter, one letter stands for one kind of Doppler signature, *i.e.* one target class in the single range cells experiments of

section 2.5. The different correlation classes will here be named targets classes. The latter thus have no more direct connection to the four modulation patterns classes of the simulation generating the input signals. The ability to discriminate diversely correlated range cells neighborhoods is notably thought as a way to distinguish clutter spanning several range cells from actual targets surrounded by clutter. The creation of relevant neighborhoods of range cells with a set of individual range cells was discussed in section 2.4.1 and will therefore not be developed here, except for the next remark regarding the $XXXXX$ pattern.

The creation of a neighborhood class based on the replication of a single range cell embedding to create a perfectly correlated local neighborhood has the positive side effect of allowing us to partially keep track of the input signal resolution impact. Indeed, if a neighborhood stems from the replication of a unique range cell embedding, the number of pulses of the original range cell IQ signal is the unique signal resolution associated with the final neighborhood representation. This allows us to verify whether the clusters potentially appearing in the output representation space of the hit2vec encoding are essentially related to the burst resolution, and not to the range cells content. This is much trickier to follow as soon as the neighborhood of range cells is made of different cells, which often implies the neighborhood is derived from a mix of input signals resolutions. A refined range cells combination and labeling mechanism to produce the dataset of range cells neighborhoods could tackle this, but this was not done here in order to avoid the question of how exactly to label neighborhoods of range cells combining input resolutions with the objective of understanding the final influence of the latter. One might add that it is actually relevant and intuitive to build neighborhoods with ranges cells of variable input signal resolution since the single cell encoding is expected to be invariant with respect to the input sampling diversity. The difficulty to identify the exact relation between the final embeddings distribution and the input sampling parameters makes it even more important to control the distribution of Doppler classes and burst resolution in evaluation datasets. The reader is reminded that the burst resolution corresponds to the association of the number of pulses and the PRF. Another example of useful constant neighborhood in terms of Doppler content could be defined by a set of range cells filled with the same clutter. Such a neighborhood could help evaluate the false alarm risk.

The more or less correlated neighborhoods of range cells are distributed over a graph defined by two possible adjacency matrices: the matrix (a) and the matrix (f) as shown on Fig. 2.7 and illustrated on Fig. 2.6, where the shared edge weight e is set to one. The graph neural network used to encode the neighborhood of $H = 5$ range cells consists in the stacking of two graph convolutional layers as defined in section 2.4.2 according to the layer proposed by [101]. These graph convolutional layers are defined with a single channel, *i.e.* they maintain one trainable weights matrix $W \in \mathbb{R}^{Q \times Q}$ each. Since the neighborhood is of size $H = 5$ and the central range cell is the one whose features we use as output neighborhood representation when no graph-scale pooling is done, two layers are enough to cover the entire neighborhood with the output representation receptive field. The term central has meaning for the non-cyclic graphs such as the graphs (a) and (e) of Fig. 2.6, in other cases it refers to the range cell under test, *i.e.* the one carrying the actual target detection. The latter remains central in the initial input representation being encoded, since it is the central column of the input matrix $Z_{I/Q}$ defined in Eq. (1.2). One can note that the trainable weights of the GNN are real-valued since the complex-to-real representation transition is done during the single cell encoding (see Fig. 2.4).

4.2 Preliminary results with supervised representation learning

The preliminary results presented here evaluate the separability of the fabricated neighborhoods of range cells in the output representation space generated by the two-layers graph convolutional neural network. A single set of metrics is presented on Fig. 4.1, Fig. 4.2 and Fig. 4.3 since none of the few setups tested yielded performances substantially superior to the others. Regarding the generation of the figures and the supervision setup, the readers may refer to the description and the references indicated in section 2.5 as the same tools and training objective were used. Four setups were tested since we experimented with both of the graph adjacency configurations mentioned in section 4.1, and with both a global mean pooling and the node2vec encoding approach applied to the central neighborhood cell to retrieve a neighborhood embedding $\mathbb{R}^Q$. Here, the weights matrices defining the GNN are both of dimensions 16×16, and thus the GNN produces a neighborhood embedding of dimension 16.

The graph embeddings 2D distribution visualization using TSNE and PCA on Fig. 4.2 and Fig. 4.3 show a pulses class 0 which contains the mixed-resolution range cells neighborhoods, while the 56 and 64 pulses classes correspond to perfectly correlated neighborhoods stemming from bursts of 56 and 64 pulses respectively. While the two figures confirm no clear latent clusters exist for specific input signals resolutions before and after training, they also suggest not much of an improvement can be observed in the latent distribution of the four signatures correlation classes. The latter observation is confirmed by the metrics on Fig. 4.1: the AUCs evolutions suggest the training is beneficial for the OCC isolation of two of the targets classes only, the other two being quickly associated with a stable random classification performance. Among the pulses classes, the 0 class gathering the mixed-resolution neighborhoods seems easily identifiable, while the pulses classes associated with single signal resolution neighborhoods also quickly end up associated with random classification performance. This suggests our graph2vec encoding yields a weak discrimination power, and is favoring neither the targets classes separation, nor the pulses classes separation.

4.3 Necessary follow-up experiments

As said in section 2.5.3 regarding the single range cell encoding experiments, the previous experimental results can only be taken as a proof-of-concept that aims at demonstrating the feasibility of encoding neighborhoods of range cells with diverse levels of correlation. Here, the potential contribution of training a GNN to encode neighborhoods of range cells for subsequent discrimination remains unclear and calls for more experiments and comparisons. Besides the discouraging nature of the performances presented in this chapter, the experiments suffer from the same shortcomings as the ones discussed in section 2.5.3. The suggested extensions can be reformulated here in the context of neighborhood encoding. Finally, another observation may be suggested for this more challenging GNN-based encoding: the possibly greater discriminatory power of deep OCC in the embeddings space operated at the output of the GNN may complement the search for a better encoding architecture. Thus, before completely discarding a neighborhood encoding method, deep OCC discrimination could be harnessed to get a more thorough evaluation of the difficulty of the targets separation.

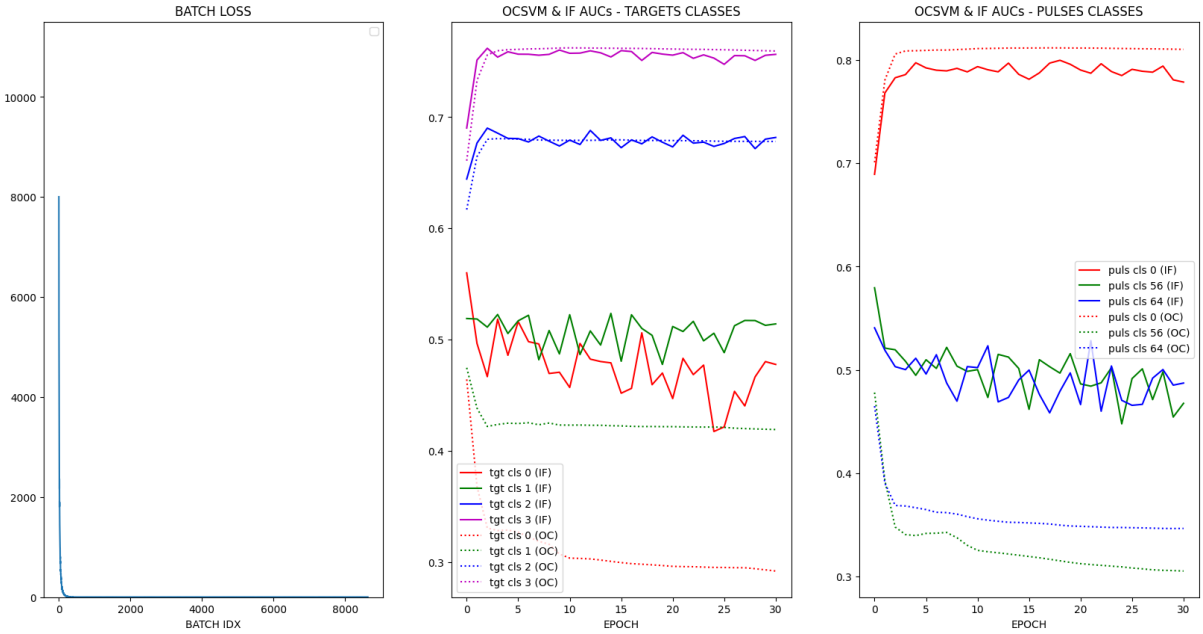


Figure 4.1: Learning metrics of the graph2vec architecture training. The GNN training impact on the AUCs describing the separability of the targets and pulses classes is disappointing. Neither the targets classes, nor the pulses classes benefit from an overall improvement of their AUC scores during training. The AUCs are computed for two OCC approaches presented in Chapter 3: IF and OC-SVM. These OCC methods are applied to the final representation produced by graph2vec.

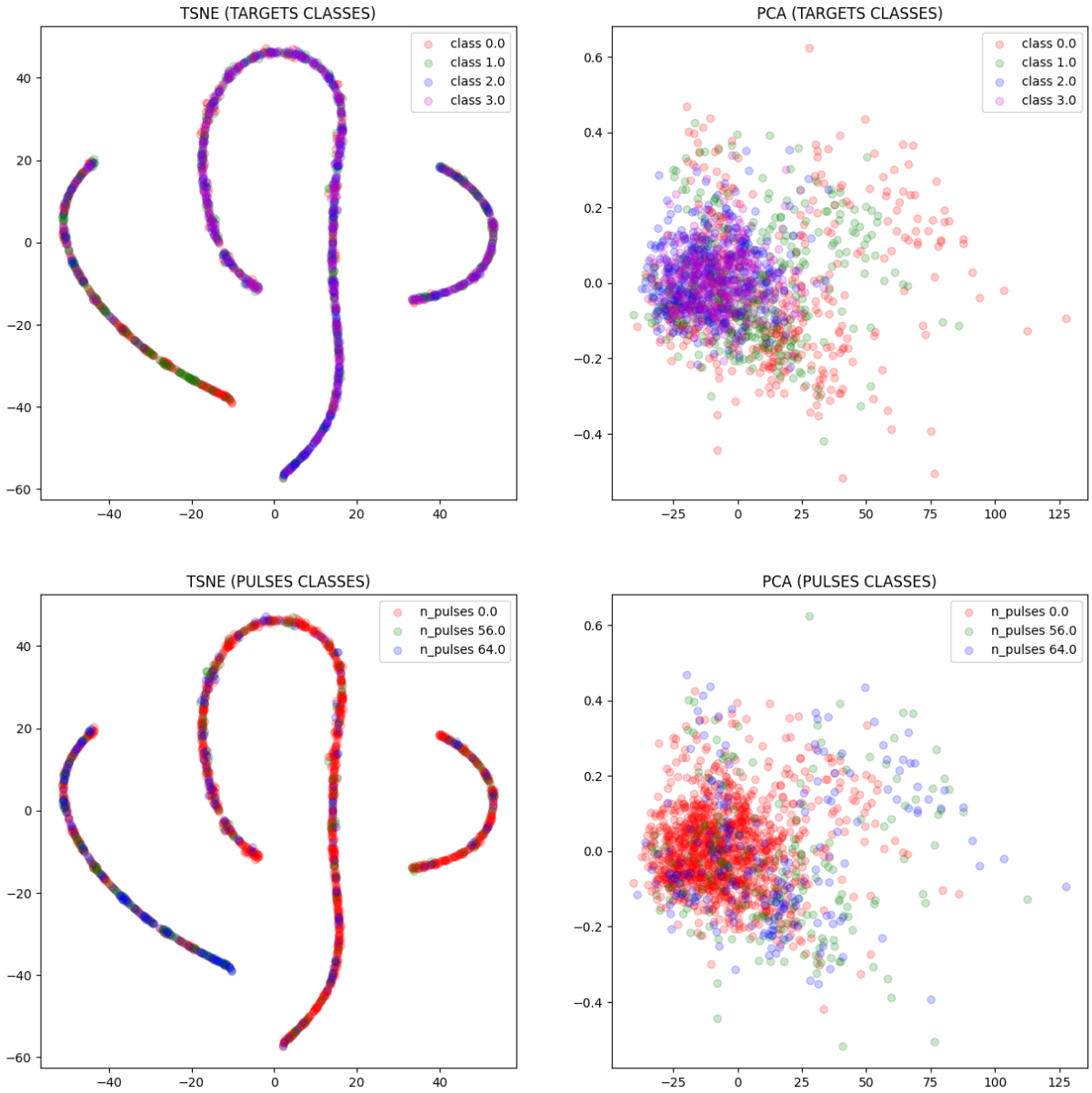


Figure 4.2: 2D visualization of the range cells neighborhoods embeddings distribution produced by the two-layers GNN, before training. One can note that this figure is still influenced by the training of the single range cell encoding neural network, which generated the single range cells representations necessary to create the dataset of range cells neighborhoods. **Top** - each color depicts one target class, i.e. one Doppler pattern. **Bottom** - each color depicts one pulse class, i.e. one Doppler resolution class.

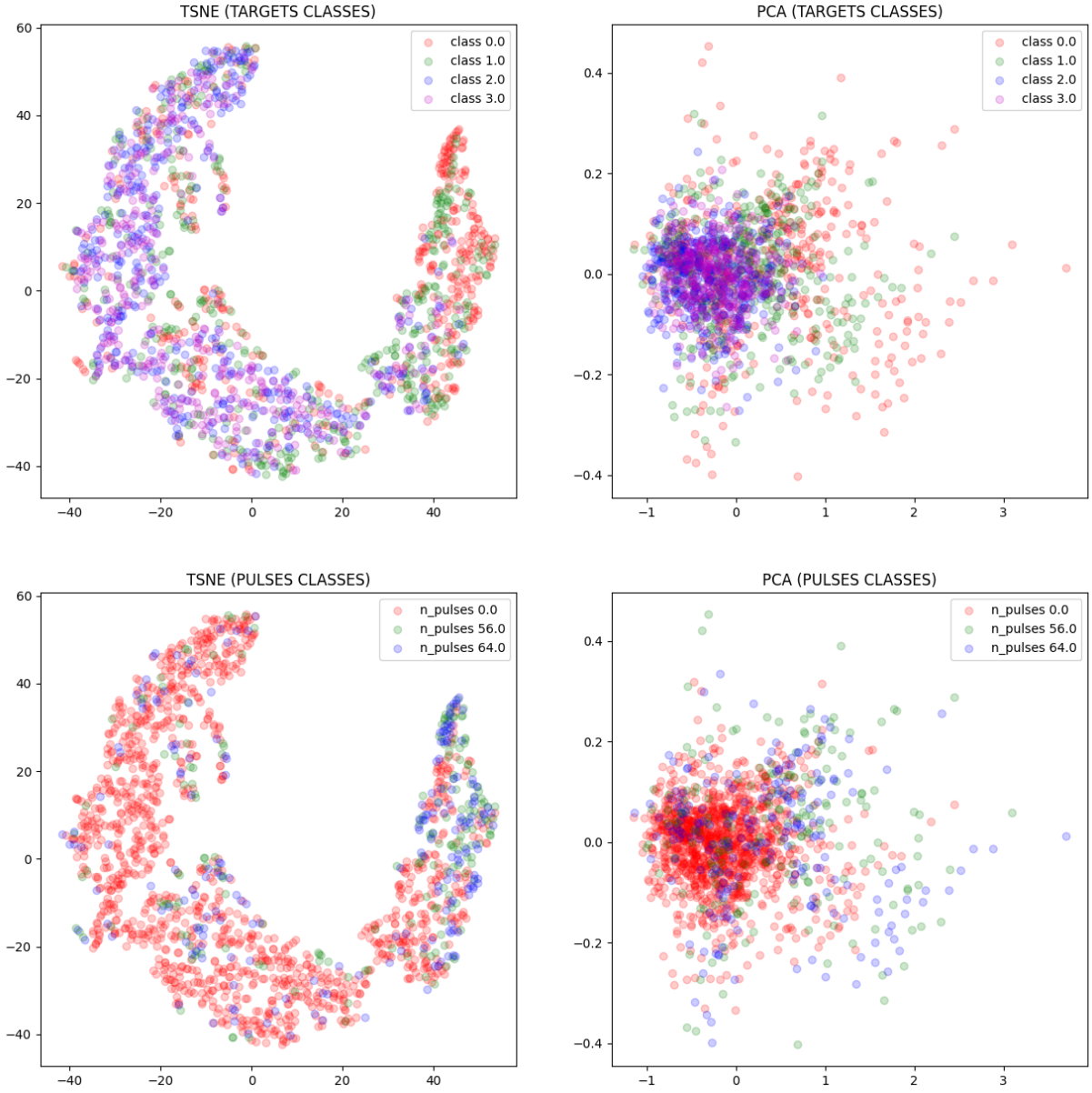


Figure 4.3: 2D visualization of the range cells neighborhoods embeddings distribution produced by the two-layers GNN, after training. No improvement can be observed with respect to the targets classes disentanglement we are seeking. **Top** - each color depicts one target class, i.e. one Doppler pattern. **Bottom** - each color depicts one pulse class, i.e. one Doppler resolution class.

Chapter 5

Conclusion and perspectives

5.1 Concluding remarks

This thesis is the result of the search for improvements in the discrimination of targets by air surveillance radars using techniques from the recent advances in machine learning. It is directly motivated by the constraints over pulse Doppler radars bursts used for air surveillance, *i.e.* the constantly changing small number of pulses available in each burst and the varying pulse repetition frequency. The thesis proposes an approach divided into two steps, first encoding, then discriminating, to end up with a score useful to the radar operator for targets discrimination. The discrimination proposed relies on the Doppler information contained in the I/Q sweep response and introduces an I/Q sweep feature enrichment to provide the neural networks targets filter with a spatial I/Q context to broaden the information available to the filter. The motivation behind this input features enrichment lies in the fact that the filter developed aims, to a certain extent, at reproducing the small and slow targets discrimination usually done with micro-Doppler features but without the required Doppler resolution to reveal such features. The solutions put forward propose to integrate the physical diversity of inputs in sampling frequency and number of pulses in the architecture of the graph neural network encoding the neighborhood of range cells processed. This information integration is inspired by the literature of resampling and array processing, and follows the trend of letting a neural network choose which information to use and how so. On a historical note, one could say the proposed Doppler processing is part of a third generation of pulse Doppler radar targets discrimination along with the geometric deep learning approaches [31] proposed in [43, 32]. Before that, the first generation could be the early low PRF radars with moving target indicator (MTI) that did not determine the Doppler shift but discarded zero and low Doppler shifts [15], and the second generation could gather the radars operating at higher PRF with refined slicing of the Doppler velocities to implement automatic targets discrimination with non-geometric deep learning methods.

Among the neural network architectures put forward by this work, a noticeable point is the choice of associating different kinds of architectures at an unusual level. The encoding neural network combining either a recurrent or a convolutional architecture with a graph neural network which then forwards a representation to a more classic discriminative network, our filter could be identified as a so-called hybrid architecture. That being said, since such a hybrid architecture as a whole is not trained at once with a common training objective, judging it as one single architecture may not be relevant. This is emphasized by our presentation of the proposed hits filter as two successive but independent steps in chapter 1, as illustrated on Fig. 1.10. This is further emphasized by the additional separation of the encoding hit2vec step into a cell2vec step and graph2vec

step, as depicted on Fig. 2.4.

5.2 Perspectives

While the results of the experiments presented in chapter 3 were quite conclusive, the performances put forward in chapters 2 and 4 were at best exploratory. As such, the experiments defined should be continued with better statistical relevance and extended by following the indications provided in the associated sections. The unsuccessful experiments aiming at developing a latent space eigenvalues-based regularization and an SPD-manifold aware adaptation of deep SVDD remain useful intuitions and may be refined. On another note, several leads for distinct possible approaches to improve the processing of a neighborhood of range cells were mentioned without actual developments:

- the definition of a global one-class classification pipeline applied to our range cells neighborhood discrimination, trainable as a whole with a unique training objective, and possibly using a generative GNN architecture;
- the adaptation of the neighborhood graph processing to handle SPD representations of range cells distributed over graph nodes instead of an $\mathbb{R}^Q$ embedding;
- the evaluation of the learned coefficients in the complex-valued convolutional layers as FIR filters while considering the varying sampling frequency of the input signals;
- the definition of an application case where the diverse radar I/Q signals are accessible to a resampling approach as presented in chapter 1, in order for a fair comparison with a non-machine learning method to be conducted.

These leads constitute approaches that remain to be explored and can provide perspective with respect to the proposals already developed in this work.

Since the processing defined and evaluated by this thesis was shown to be close to ECG or radio signal processing, and also to molecules representation learning, the active research motivated by these applications should be a continuous source of tools to solve the problem of neighborhood of range cells encoding. More generally, we did not make use of all of the tools and innovative neural networks layers available in the literature. The recent developments of the deep learning literature establish therefore a promising avenue for improvements. Finding an effective and explainable way to compare short signals with varying sampling parameters and dispersed over local graphs remains a task relevant for many applications, especially with the advent of the internet of things.

Lastly, one should note that even with a relevant encoding of similar targets and an effective targets separation, additional mission-specific data could be taken into account to detach targets similar physically but with behaviors making them of different levels of interest for the radar operator. This additional processing could be handled by a downstream processing step in the radar pipeline, such as the tracking stage where the targets' trajectories can be a revealing information with respect to their respective missions. For instance a small civilian drone is not necessarily an object of interest to a radar operator but if it keeps flying over a sensible site it should become one. In such a case, our filter aims at detecting the drone and dissociating it from the surrounding clutter, but it does not address the target behavior analysis task. The previous point emphasizes how much automatic processing could be necessary to actually help radar operators fulfill their missions, and thus how much machine learning could help improve this kind of sensor, assuming enough labeled data is at hand and the possibility to modify different parts of the radar processing pipeline.

Appendices

Appendix A

About the relation between RPO, Deep RPO and the Mahalanobis distance

To elaborate on the relation between RPO and the Mahalanobis distance, the latter describing an ellipsoid in the data points representation space, let us remind the results presented in [186]. Let us consider a data point $x \in \mathbb{R}^d$ belonging to a data matrix $X^{d\times n}$ following an Elliptically Symmetric Distribution (ESD) containing n samples for which we want to compute an outlyingness score $\mathcal{O}(x)$. The ESD hypothesis guarantees that the sample covariance matrix Σ is positive definite. Using Σ, one can define an outlyingness score using the Mahalanobis distance:

$$\mathcal{O}_{MAHALANOBIS}(x) = \sqrt{(x-\mu_X)^T\Sigma^{-1}(x-\mu_X)} \tag{A.1}$$

where μ_X is the data points mean, i.e. the mean column of X. According to the extended Cauchy-Schwarz inequality [96], for any nonzero vector $u \in \mathbb{R}^d$:

$$(u^T(x-\mu_X))^2 \leq (u^T\Sigma u)((x-\mu_X)^T\Sigma^{-1}(x-\mu_X)) \tag{A.2}$$

$$\implies \frac{(u^T(x-\mu_X))^2}{u^T\Sigma u} \leq (x-\mu_X)^T\Sigma^{-1}(x-\mu_X) \tag{A.3}$$

where $u^T\Sigma u \geq 0$ since Σ is positive definite. As suggested in [96] if one takes $u = \alpha\Sigma^{-1}(x-\mu_X)$, with $\alpha \in \mathbb{R}^*$, one gets:

$$\implies \frac{((\alpha\Sigma^{-1}(x-\mu_X))^T(x-\mu_X))^2}{(\alpha\Sigma^{-1}(x-\mu_X))^T\Sigma(\alpha\Sigma^{-1}(x-\mu_X))} \leq (x-\mu_X)^T\Sigma^{-1}(x-\mu_X) \tag{A.4}$$

$$\implies \frac{((\alpha\Sigma^{-1}(x-\mu_X))^T(x-\mu_X))^2}{\alpha^2(\Sigma^{-1}(x-\mu_X))^T\mathbb{I}_d(x-\mu_X)} \leq (x-\mu_X)^T\Sigma^{-1}(x-\mu_X) \tag{A.5}$$

the latter leading to the equality case, showing us the upper bound is attainable. Since the upper bound is reachable for $u = \alpha\Sigma^{-1}(x-\mu_X)$, and that such format allows u to be a RP of unit norm as required by the definition of RPO for Eq. 3.2, one can intuitively conclude that using a maximum estimator over numerous RPs the bound is approached or reached in Eq.A.3 by the left term, i.e. for $\mathbb{U}$ a large set of unit norm RPs:

$$\max_{u\in\mathbb{U}} \frac{(u^T(x-\mu_X))^2}{u^T\Sigma u} = (x-\mu_X)^T\Sigma^{-1}(x-\mu_X) \tag{A.6}$$

This result is actually well-known and called the Maximization Lemma in [96], where instead of $u \in \mathbb{U}$, u is an arbitrary nonzero vector, i.e. $\underset{u\in\mathbb{U}}{max}$ is replaced by $\underset{x\neq 0}{max}$. The proof for this lemma is actually the previous example case where $u = \alpha\Sigma^{-1}(x - \mu_X)$ is shown to make the equality case happen. Note that using the equality case format for u implies knowing the covariance matrix Σ^{-1}. This emphasizes the relevance of RPO to compute a multivariate outlyingness without requiring a covariance matrix computation. From Eq. A.6, one can deduce:

$$\underset{u\in\mathbb{U}}{max}\frac{(u^T x - u^T\mu_X)^2}{\sigma^2(u^T x)} = (x - \mu_X)^T\Sigma^{-1}(x - \mu_X) \tag{A.7}$$

this expression makes the projected sample $u^T x$ and the projected mean $u^T\mu_X$ appear. Note that for the denominator the transformation is as follows:

$$\sigma^2(u^T x) = E\left[(u^T(x - \mu_X))(u^T(x - \mu_X))^T\right] = E\left[u^T(x - \mu_X)(x - \mu_X)^T u\right] = u^T\Sigma u \tag{A.8}$$

Eq. A.7 now only differs from RPO due to the square and the estimators of first and second order statistics: RPO replaces the mean and the variance of the projected inputs $x \in X$ with the robust estimators median and median absolute deviation respectively, which in turn adds a constant factor with respect to the Mahalanobis distance. Regarding the relevance of these robust estimators choice and the additional constant factor see [186].

We thus get back to the conclusion of [186] indicating the equivalence (up to a constant factor) between the Mahalanobis distance and RPO computed with an infinite number of RPs under the multivariate elliptical distribution hypothesis. Whereas RPO as presented in Table 3.9 used the *max* estimator to integrate over RPs as defined in [186], Deep RPO replaces the *max* with a *mean* estimator in Eq. 3.2 to measure outlyingness in the latent representation space provided by a neural network. This replacement is motivated by empirical results and an interpretation provided in [25]. The drawback of this change is that the Mahalanobis equivalence guarantee is lost, since the Maximization Lemma leading to Eq. A.6 can not be used with a *mean*. Working with an ellipsoid instead of a latent hypersphere as in [153] supposedly made the latent normality boundary used by the training objective more flexible and tailored to the data, and was the original motivation of [25]. Recall that this normality boundary is fitted to training data before training and frozen, the boundary being defined by a single location estimator in the case of Deep SVDD, and by as many location and spread estimators as there are RPs in the case of Deep RPO.

Intuitively, the *mean* will pull the quantity integrated over the large set of RPs away from the upper bound. Even though the score is based on projected 1D outlyingnesses each normalized by their respective location and spread estimators, there is no assurance that once integrated with a *mean* into one final outlyingness these quantities generate a normality ellipsoid similar to the Mahalanobis one. Actually, nothing indicates the integrated outlyingness describes any kind of ellipsoid in the input vector representation space. However, since the *mean* integrates over the dimensions created by the set of RPs, and since along these dimensions each 1D coordinate is centered and normalized using its own location and spread estimators, the *mean* still relates to an ellipsoid in the high-dimensional representation space generated by the RPs.

Appendix B

Affine invariance of RPO with max and mean estimators

The content of this appendix was originally presented in [26]. We want to prove the affine invariance of the following quantity, called the random projection outlyingness (RPO):

$$O_{RPO}(x, X) = \sup_{\|u\|=1} \frac{|u^T x - Med(u^T X)|}{MAD(u^T X)} \tag{B.1}$$

that is, we want to prove the following equality:

$$O_{RPO}(Ax + b, AX + b) \stackrel{?}{=} O_{RPO}(x, X) \tag{B.2}$$

where:

- $x \in \mathbb{R}^{d\times 1}$ is the data point for which we want to compute an outlyingness measure;
- $X \in \mathbb{R}^{d\times n}$ is the data matrix containing n features vectors in $\mathbb{R}^d$ (i.e. the data distribution, including x);
- $u \in \mathbb{R}^{d\times 1}$ is a random projection vector of unit norm, *i.e.* $u \in S^{d-1}$ where $S^{d-1} = \{x \in \mathbb{R}^d : \|x\|_2 = 1\}$;
- $A \in \mathbb{R}^{d\times d}$ is a non-singular matrix (for the affine transformation);
- $b \in \mathbb{R}^{d\times 1}$ is a constant vector (for the affine transformation);
- $Med(u^T X)$ is the median of the scalars generated by the 1D projection of all x in X by u;
- $MAD(u^T X)$ is the median absolute deviation of the same scalars, *i.e.* $MAD(u^T X) = Med(|u^T x - Med(u^T X)|)$.

$AX + b$ is a permissive notation defining the affine transformation of every column features vector with A and b, *i.e.* the affine transformation of the whole data distribution on which the location and scatter estimators, respectively the Med and the MAD, are applied. We also want to prove the affine invariance of that same quantity where the *sup* is replaced by a *mean* estimator:

$$O_{RPO-MEAN}(x, X) = \underset{\|u\|=1}{mean} \frac{|u^T x - Med(u^T X)|}{MAD(u^T X)} \tag{B.3}$$

This RPO-MEAN is the RPO variant introduced in [25] and used in the encoding neural network output space for our Deep RPO experiments. The Eq. B.1 stems from [62, 90] and defines the outlyingness used to generate the statistical depth [208] called random projection depth (RPD), the latter corresponding to the following expression: $RPD(x, X) = \frac{1}{1+O_{RPO}(x,X)}$. Note that this depth, and the associated outlyingness of Eq. B.1, are defined with an infinite number of random projections $u \in \mathbb{R}^d$, thus the two quantities (RPO and RPD) can only be implemented with a stochastic approximation, *i.e.* with a large but finite number of random projections. For instance, in the experiments described in this paper, the *sup* and *mean* over all RPs of unit norm are approximated with a *max* and *mean* over 1000 RPs respectively. For other experiments on RPO with diverse quantities of RPs, see [25].

Proof: Let us start by noticing that both the upper and lower parts of the RPO ratio are invariant to the bias term b of the affine transformation:

$$|u^T(Ax+b) - Med(u^T(AX+b))| = |u^TAx + u^Tb - Med(u^TAX) - u^Tb| \tag{B.4}$$
$$= |u^TAx - Med(u^TAX)| \tag{B.5}$$
$$MAD(u^T(AX+b)) = Med(|u^T(Ax+b) - Med(u^T(AX+b))|) \tag{B.6}$$
$$= Med(|u^TAx + u^Tb - Med(u^TAX) - u^Tb|) \tag{B.7}$$
$$= Med(|u^TAx - Med(u^TAX)|) \tag{B.8}$$
$$= MAD(u^T(AX)) \tag{B.9}$$

This indicates both RPO and RPO-MEAN are translation invariant, a partial requirement to achieve affine invariance. Let us factor the upper and lower parts of the RPO ratio to make a unit norm vector $\frac{u^TA}{\|u^TA\|}$ appear:

$$|u^T(Ax) - Med(u^T(AX))| = \left| \left\|u^TA\right\| \frac{u^TA}{\|u^TA\|}x - Med\left(\left\|u^TA\right\| \frac{u^TA}{\|u^TA\|}X\right)\right| \tag{B.10}$$
$$= \left\|u^TA\right\| \left| \frac{u^TA}{\|u^TA\|}x - Med\left(\frac{u^TA}{\|u^TA\|}X\right)\right| \tag{B.11}$$
$$MAD(u^T(Ax)) = Med\left(\left|\left\|u^TA\right\| \frac{u^TA}{\|u^TA\|}x - Med\left(\left\|u^TA\right\| \frac{u^TA}{\|u^TA\|}X\right)\right|\right) \tag{B.12}$$
$$= \left\|u^TA\right\| Med\left(\left| \frac{u^TA}{\|u^TA\|}x - Med\left(\frac{u^TA}{\|u^TA\|}X\right)\right|\right) \tag{B.13}$$
$$\tag{B.14}$$

This enables us to rewrite the RPO ratio as follows:

$$\frac{|u^T Ax - Med(u^T AX)|}{MAD(u^T AX)} = \frac{\left\|u^T A\right\| \left| \frac{u^T A}{\|u^T A\|} x - Med\left(\frac{u^T A}{\|u^T A\|} X \right) \right|}{\|u^T A\| \, Med\left(\left| \frac{u^T A}{\|u^T A\|} x - Med\left(\frac{u^T A}{\|u^T A\|} X \right) \right| \right)} \tag{B.15}$$

$$= \frac{\left| \frac{u^T A}{\|u^T A\|} x - Med\left(\frac{u^T A}{\|u^T A\|} X \right) \right|}{Med\left(\left| \frac{u^T A}{\|u^T A\|} x - Med\left(\frac{u^T A}{\|u^T A\|} X \right) \right| \right)} \tag{B.16}$$

Thus for any u, x and X, let $f(u)$ be:

$$f(u) := \frac{|u^T x - Med(u^T X)|}{MAD(u^T X)} \tag{B.17}$$

if one defines $\phi(u) := \frac{u^T A}{\|u^T A\|}$ and $\psi(u) := u^T A$, we have $f \circ \phi(u) = f \circ \psi(u)$. Moreover, since ϕ is a bijection from S^{d-1} to S^{d-1}, for g the *mean* or *sup* operator applied to every existing random projection u:

$$\underset{u \in S^{d-1}}{g} [f(u)] = \underset{u \in S^{d-1}}{g} [f \circ \phi(u)] \tag{B.18}$$

Combining the two last equalities provides us with the invariance to the linear transformation defined by A:

$$\underset{u \in S^{d-1}}{g} [f(u)] = \underset{u \in S^{d-1}}{g} [f \circ \psi(u)] \tag{B.19}$$

In other words, since ϕ is a bijection and all existing random projections of unit norm are considered during the integration over S^{d-1}, the operator g is not affected whether the RPs are transformed by ϕ beforehand or not. The intuition behind this invariance is that we are looking over the same infinite set of random projections, the transformation matrix A at most reshuffling the RPs in the infinite set the estimator integrates over. More generally, that is true for any permutation invariant[1] operator g.

The previous translation invariance with respect to b and the linear transformation invariance with respect to A prove the affine invariance of both RPO and RPO-MEAN.

[1] An operator is permutation invariant if any permutation of the inputs does not change the output:

$$g(x_1, \ \dots, x_t) = g(x_{\pi(1)}, \ \dots, x_{\pi(t)})$$

for any permutation π.

Appendix C

Unsuccessful architectures

This appendix contains a few of the unsuccessful neural networks architectures which have been mentioned in the manuscript. It first goes over the unsuccessful single range cell I/Q encoding architectures proposed in 2.3, and then puts forward the attempted SPD-manifold processing adaptation of Deep SVDD suggested in 3.2.3.

C.1 Cell2vec architectures

One can notice that these architectures, in addition to transforming a column of the input matrix $Z_{I/Q}$ (see Eq. (1.2)) of variable size into a fixed-size vector in $\mathbb{R}^Q$, convert the complex-valued representation into a real-valued one. These architectures correspond to the cell2vec step in the hit2vec depiction of Fig. 2.4. In the FCAE described in table C.2, as was the case for the FCN put forward in table 2.2, using a convolution with a large kernel size on the input signal is particularly interesting since it makes the potential interpretation of the learned weights as FIR filter coefficients more expressive.

C.2 Deep Riemannian one-class classification

The table C.4 describes an example architecture we experimented with to define an SPD-manifold aware equivalent to the deep SVDD [153] one-class classification method. This architecture was tested on SPD covariance matrices computed either over a fixed set of input images transformations, or over the rows or columns of the input images. The resulting covariance matrix was of size 28×28, hence the input dimension of the first BiMap layer in the proposed architecture. The intuition, the loss and the layers associated with this architecture are discussed in section 3.2.

The architecture presented here can be applied to the covariance matrices computed over the individual components of Doppler spectrums (see Fig. 3.8) or computed over the output of convolutional layers, or also to the Toeplitz autocorrelation matrices associated with AR models (see 2.3.1). It is also possible to extend this SPD-manifold adaptation of Deep SVDD to the complex-valued equivalent of SPD matrices, *i.e.* HPD matrices (see 3.2). These architectures were implemented and trained using the torchspdnet [10] library[1]. This versatility of the SPD manifold-aware processing, in addition to its potential for more efficient learning in terms of required iterations and data, motivated our experiments to develop a deep SVDD SPD equivalent.

[1] `https://gitlab.lip6.fr/schwander/torchspdnet` Accessed: 28-10-2022

Layers in forward order	**Layer parameters**
Dropout	/
$\mathbb{C}$-LSTM (encode)	3 layers, hidden dim 16, input dim 1
$\mathbb{R}$-Hidden	hidden dim 16 ($\times 2$)
$\mathbb{C}$-LSTM (decode)	3 layers, hidden dim 16, input dim 1
$\mathbb{C}$-Linear	input dim 16, output dim 1

Table C.1: LSTM-based seq2seq architecture used in our experiments without success. Only the last encoder layer hidden and cell states (hence the "$\times 2$") were kept as the embedding of the input range cell I/Q signal and passed on to the LSTM decoder. These hidden and cell states were furthermore reduced to their real part in order to limit the transit of information to a real-valued representation that will serve as range cell embedding in $\mathbb{R}^Q$. This complex to real, and back to complex is illustrated on Fig. 2.3. Input dimensionality of both LSTM networks are equal to one in order to accept the complex-valued I/Q signal. For the decoder, this input can be the ground truth stemming from the teacher forcing. A complementary complex-valued linear layer was added to further process the output of the LSTM decoder before evaluating the reconstruction error. A dropout layer without rescaling was added in some of our experiments to complicate the task of reconstruction (see 2.3.2).

Layers in forward order	**Layer parameters**
Dropout	/
$\mathbb{C}$-conv 1D	kernel 6, stride 2, in chan 1, out chan 12
$\mathbb{C}$-BN 1D	chan 12
$\mathbb{C}$-ReLU	chan 12
$\mathbb{C}$-conv 1D	kernel 1, stride 1, in chan 12, out chan 12
$\mathbb{C}$-BN 1D	chan 12
$\mathbb{C}$-ReLU	chan 12
$\mathbb{C}$-conv 1D	kernel 1, stride 1, in chan 12, out chan 16
$\mathbb{C}$-BN 1D	chan 16
$\mathbb{C}$-ReLU	chan 16
$\mathbb{R}$-Bottleneck	chan 16 (global pool here)
$\mathbb{C}$-tconv 1D	kernel 6, stride 2, in chan 16, out chan 12
$\mathbb{C}$-BN 1D	chan 12
$\mathbb{C}$-ReLU	chan 12
$\mathbb{C}$-tconv 1D	kernel 1, stride 1, in chan 12, out chan 12
$\mathbb{C}$-BN 1D	chan 12
$\mathbb{C}$-ReLU	chan 12
$\mathbb{C}$-tconv 1D	kernel 1, stride 1, in chan 12, out chan 1

Table C.2: FCAE architecture used in our experiments. This defines a convolution-based seq2seq architecture, where *tconv* stands for transposed convolution. Note that in our case, the global pooling over the features dimension is only applied when retrieving the bottleneck embedding, and is not necessary in the forward pass to compute the training loss. Thus, this global pooling does not appear in this table as a layer. To allow for the generation of a reconstruction of varying size, the bottleneck features dimensionality will vary along with the input size. The fixed-size embeddings can still be produced since the number of channels of the convolutional bottleneck remains constant. As for the LSTM-based seq2seq (see table C.1), only the real part of the output of the encoder is passed on to the decoder to limit the transit of information to a real-valued representation that will serve as range cell embeddings in $\mathbb{R}^Q$. The complex-to-real encoding is thus learned by the generative network, even though the decoder has complex-valued weights in order to output a complex-valued input reconstruction. As for the FCN (see table 2.2), using a convolution with a large kernel size on the input signal is particularly interesting since it makes the potential interpretation of the learned weights as FIR filter coefficients more expressive.

Layers in forward order	**Layer parameters**
$\mathbb{C}$-RNN (encode)	3 layers, hidden dim 16, input dim 1

Table C.3: RNN-based encoding architecture used in our experiments without success. Only the last encoder layer hidden state was kept as the embedding of the input range cell I/Q signal. This hidden state was furthermore reduced to its real part in order to produce a real-valued representation that will serve as range cell embedding in $\mathbb{R}^Q$. Input dimensionality is equal to one in order to accept the complex-valued I/Q signal.

Layers in forward order	**Layer parameters**
ReEig	"SPDNet non-linearity"
BiMap	input dim 28, output dim 24
ReEig	"SPDNet non-linearity"
BiMap	input dim 24, output dim 12
ReEig	"SPDNet non-linearity"
BiMap	input dim 12, output dim 4

Table C.4: SPD-manifold aware deep SVDD architecture adaptation used in our experiments. A usual continuous dimensionality reduction is operated during the forward pass. No LogEig operation is applied at the end of this architecture since we want the output representation to remain on the SPD matrices manifold in order to compute the distance to a reference latent SPD centroid to define the training loss.

Appendix D

Publicly available codes

A few of the codes developed during this thesis have been made publicly available at the time of publication of this document and allow for the reproduction of parts of the experiments that led to scientific publications:

- Deep MSVDD [73] reimplementation: available on GitHub `https://github.com/Blupblupblup/Deep-MSVDD-PyTorch` (Accessed: 25/11/2022)
- "Near out-of-distribution detection for low-resolution radar micro-Doppler signatures" [26] paper experiments: available on GitHub `https://github.com/Blupblupblup/Near-OOD-Doppler-Signatures` (Accessed: 25/11/2022)
- Simulated Doppler signatures generation used for [26]: available on GitHub `https://github.com/Blupblupblup/Doppler-Signatures-Generation` (Accessed: 25/11/2022)
- ICLR 2021 challenge submission repository: available on GitHub `https://github.com/Blupblupblup/submission_geomstatsICLR2021challenge_NaiveImageAD_Euclidean_vs_Riemannian` (Accessed: 25/11/2022)

Bibliography

[1] Fir filter design for complex signal. https://dsp.stackexchange.com/questions/33933/fir-filter-design-for-complex-signal#33934. Accessed: 18-11-2022.

[2] Gamekeeper 16u counter-uas radar. https://www.aveillant.com/products/gamekeeper/. Accessed: 08-08-2022.

[3] Pytorch. https://pytorch.org/. Accessed: 20/11/2022.

[4] Pytorch: Autograd for complex numbers. https://pytorch.org/docs/stable/notes/autograd.html#autograd-for-complex-numbers. Accessed: 20/11/2022.

[5] Pytorch: complex numbers. https://pytorch.org/docs/stable/complex_numbers.html. Accessed: 20-11-2022.

[6] Receivers bandwidth. https://www.radartutorial.eu/09.receivers/rx10.en.html. Accessed: 31-08-2022.

[7] Thales Ground Master 200. https://www.thalesgroup.com/en/markets/defence-and-security/air-forces/airspace-protection/mid-range-radars/ground-master-200. Accessed: 05-08-2022.

[8] Thales Ground Master 400. https://www.thalesgroup.com/en/markets/defence-and-security/air-forces/airspace-protection/surface-radars/ground-master-400-alpha. Accessed: 05-08-2022.

[9] Thales Ground Master 60. https://www.thalesgroup.com/en/markets/defence-and-security/air-forces/airspace-protection/short-range-radar/ground-master-60. Accessed: 05-08-2022.

[10] torchspdnet. https://gitlab.lip6.fr/schwander/torchspdnet. Accessed: 05-10-2022.

[11] IEEE standard letter designations for radar-frequency bands. *IEEE Std 521-2019 (Revision of IEEE Std 521-2002)*, pages 1–15, 2020.

[12] P-A Absil and Jérôme Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158, 2012.

[13] Dinesh Acharya, Zhiwu Huang, Danda Pani Paudel, and Luc Van Gool. Covariance pooling for facial expression recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 367–374, 2018.

[14] Claude Adnet. *Unification des méthodes d'analyse spectrale (Fourier et haute résolution) en vue de la réalisation d'un système expert d'aide à l'analyse*. PhD thesis, Grenoble INPG, 1990.

[15] Clive Alabaster. *Pulse Doppler Radar: Principles, Technology, Applications*. SciTech Publishing, 2012.

[16] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[17] Linda Hadded Aouchiche. *Localisation à haute résolution de cibles lentes et de petite taille à l'aide de radars de sol hautement ambigus*. PhD thesis, Université Rennes 1, 2018.

[18] Marc Arnaudon, Frédéric Barbaresco, and Le Yang. Riemannian medians and means with applications to radar signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 7(4):595–604, 2013.

[19] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[20] JA Barrachina, Chengfang Ren, Gilles Vieillard, Christele Morisseau, and J-P Ovarlez. Real-and complex-valued neural networks for sar image segmentation through different polarimetric representations. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1456–1460. IEEE, 2022.

[21] Jose Agustin Barrachina. Negu93/cvnn: Complex-valued neural networks. `https://doi.org/10.5281/zenodo.7303587`. Accessed: 20-11-2022.

[22] Jose Agustin Barrachina, Chenfang Ren, Christele Morisseau, Gilles Vieillard, and J-P Ovarlez. Complex-valued vs. real-valued neural networks for classification perspectives: An example on non-circular data. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2990–2994. IEEE, 2021.

[23] Peter L Bartlett and Marten H Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(8), 2008.

[24] Martin Bauw, Santiago Velasco-Forero, Jesus Angulo, Claude Adnet, and Olivier Airiau. From unsupervised to semi-supervised anomaly detection methods for hrrp targets. In *2020 IEEE Radar Conference (RadarConf20)*, pages 1–6. IEEE, 2020.

[25] Martin Bauw, Santiago Velasco-Forero, Jesus Angulo, Claude Adnet, and Olivier Airiau. Deep random projection outlyingness for unsupervised anomaly detection. In *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*, 2021.

[26] Martin Bauw, Santiago Velasco-Forero, Jesus Angulo, Claude Adnet, and Olivier Airiau. Near out-of-distribution detection for low-resolution radar micro-doppler signatures. *arXiv preprint arXiv:2205.07869*, 2022.

[27] Piotr Bielak and Tomasz Jan Kajdanowicz. Pytorch-geometric edge - a library for learning representations of graph edges. In *Learning on Graphs Conference*, 2022.

[28] Svante Björklund. Target detection and classification of small drones by boosting on radar micro-doppler. In *2018 15th European Radar Conference (EuRAD)*, pages 182–185. IEEE, 2018.

[29] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[30] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[31] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[32] Daniel Brooks. *Deep Learning and Information Geometry for Time-Series Classification*. PhD dissertation, Sorbonne Université, 2020.

[33] Daniel Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Temporal deep learning for drone micro-doppler classification. In *2018 19th International Radar Symposium (IRS)*, pages 1–10. IEEE, 2018.

[34] Daniel Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. A hermitian positive definite neural network for micro-doppler complex covariance processing. In *2019 International Radar Conference (RADAR)*, pages 1–6. IEEE, 2019.

[35] Daniel Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Riemannian batch normalization for spd neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[36] Daniel Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Second-order networks in pytorch. In *International Conference on Geometric Science of Information*, pages 751–758. Springer, 2019.

[37] Daniel A Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Complex-valued neural networks for fully-temporal micro-doppler classification. In *2019 20th International Radar Symposium (IRS)*, pages 1–10. IEEE, 2019.

[38] Daniel A Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Exploring complex time-series representations for riemannian machine learning of radar data. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3672–3676. IEEE, 2019.

[39] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[40] Stefan Brüggenwirth, Marcel Warnke, Simon Wagner, and Kilian Barth. Cognitive radar for classification. *IEEE Aerospace and Electronic Systems Magazine*, 34(12):30–38, 2019.

[41] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[42] John Parker Burg. *Maximum entropy spectral analysis*. Stanford University, 1975.

[43] Yann Cabanes. *Multidimensional complex stationary centered Gaussian autoregressive time series machine learning in Poincaré and Siegel disks: application for audio and radar clutter classification*. PhD dissertation, Université de Bordeaux, 2022.

[44] Yann Cabanes, Frédéric Barbaresco, Marc Arnaudon, and Jérémie Bigot. Toeplitz hermitian positive definite matrix machine learning based on fisher metric. In *International Conference on Geometric Science of Information*, pages 261–270. Springer, 2019.

[45] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

[46] Miquel Calvo and Josep M Oller. A distance between multivariate normal distributions based in an embedding into the siegel group. *Journal of multivariate analysis*, 35(2):223–242, 1990.

[47] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[48] Benson Chen, Gary Bécigneul, Octavian-Eugen Ganea, Regina Barzilay, and Tommi Jaakkola. Optimal transport graph neural networks. *arXiv preprint arXiv:2006.04804*, 2020.

[49] Victor C Chen. Advances in applications of radar micro-doppler signatures. In *2014 IEEE Conference on Antenna Measurements & Applications (CAMA)*, pages 1–4. IEEE, 2014.

[50] Victor C Chen. *The micro-Doppler effect in radar*. Artech house, 2019.

[51] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[52] Penny Chong, Lukas Ruff, Marius Kloft, and Alexander Binder. Simple and effective prevention of mode collapse in deep one-class classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

[53] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[54] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[55] Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. *arXiv preprint arXiv:1603.00982*, 2016.

[56] European Commission. Artificial intelligence for europe. `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM%3A2018%3A237%3AFIN`, 2018. Accessed: 20/11/2022.

[57] Emanuele Dalsasso, Loïc Denis, and Florence Tupin. Sar2sar: A semi-supervised despeckling algorithm for sar images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:4321–4329, 2021.

[58] Rapport de la Task Force IA. L'intelligence artificielle au service de la défense. `https://www.defense.gouv.fr/sites/default/files/aid/20200108-NP-Rapport%20de%20la%20Task%20Force%20IA%20Septembre.pdf`, 2019. Accessed: 20/11/2022.

[59] Agence de l'innovation de défense. Aap thèses aid classiques - thématiques prioritaires oar, 2019.

[60] Agence de l'innovation de défense. Aap thèses aid classiques - thématiques prioritaires oar, 2020.

[61] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[62] David L Donoho, Miriam Gasko, et al. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *The Annals of Statistics*, 20(4):1803–1827, 1992.

[63] Lan Du, Xu Liu, Bo Li, and Shuwen Xu. Hrrp clutter rejection via one-class classifier with hausdorff distance. *IEEE Transactions on Aerospace and Electronic Systems*, 2019.

[64] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[65] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[66] Yonina C Eldar. *Sampling theory: Beyond bandlimited systems*. Cambridge University Press, 2015.

[67] Bo Feng, Bo Chen, and Hongwei Liu. Radar hrrp target recognition with deep networks. *Pattern Recognition*, 61:379–393, 2017.

[68] Patrick Flandrin. *Temps–Fréquence (Traité des Nouvelles Technologies, série Traitement du Signal)*. Hermès, 1993.

[69] Jonathan Frankle, David J Schwab, and Ari S Morcos. Training batchnorm and only batchnorm: On the expressive power of random features in cnns. *arXiv preprint arXiv:2003.00152*, 2020.

[70] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *International Conference on Machine Learning*, pages 2083–2092. PMLR, 2019.

[71] George M Georgiou and Cris Koutsougeras. Complex domain backpropagation. *IEEE transactions on Circuits and systems II: analog and digital signal processing*, 39(5):330–334, 1992.

[72] Julien Gérard, Joanna Tomasik, Christèle Morisseau, Arpad Rimmel, and Gilles Vieillard. Micro-doppler signal representation for drone classification by deep learning. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 1561–1565. IEEE, 2021.

[73] Zahra Ghafoori and Christopher Leckie. Deep multi-sphere support vector data description. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 109–117. SIAM, 2020.

[74] Raja Giryes, Guillermo Sapiro, and Alex M Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457, 2016.

[75] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1705–1714, 2019.

[76] Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9211–9219, 2019.

[77] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[78] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks*, volume 2, pages 729–734, 2005.

[79] Nitzan Guberman. On complex valued convolutional neural networks. *arXiv preprint arXiv:1602.09046*, 2016.

[80] Lionel Gueguen, Santiago Velasco-Forero, and Pierre Soille. Local mutual information for dissimilarity-based image segmentation. *Journal of mathematical imaging and vision*, 48(3):625–644, 2014.

[81] Ronny Hänsch. Complex-valued multi-layer perceptrons–an application to polarimetric sar data. *Photogrammetric Engineering & Remote Sensing*, 76(9):1081–1088, 2010.

[82] Mehrtash Harandi and Basura Fernando. Generalized backpropagation,etude de cas: Orthogonality. *arXiv preprint arXiv:1611.05927*, 2016.

[83] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[84] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[85] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[86] Akira Hirose. *Complex-valued neural networks: theories and applications*, volume 5. World Scientific, 2003.

[87] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[88] Borui Hou, Jianyong Yang, Pu Wang, and Ruqiang Yan. Lstm-based auto-encoder model for ecg arrhythmias classification. *IEEE Transactions on Instrumentation and Measurement*, 69(4):1232–1240, 2019.

[89] Zhiwu Huang and Luc Van Gool. A riemannian network for spd matrix learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[90] Peter J Huber. Projection pursuit. *The annals of Statistics*, pages 435–475, 1985.

[91] Yan Hui and MR Smith. Mri reconstruction from truncated data using a complex domain backpropagation neural network. In *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing. Proceedings*, pages 513–516. IEEE, 1995.

[92] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, 2015.

[93] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *Proceedings of the IEEE international conference on computer vision*, pages 2965–2973, 2015.

[94] Danilo Jimenez Rezende, SM Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. *Advances in neural information processing systems*, 29, 2016.

[95] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*, 2021.

[96] Richard Arnold Johnson, Dean W Wichern, et al. *Applied multivariate statistical analysis*, volume 6. Pearson London, UK:, 2014.

[97] Hermann Karcher. Riemannian center of mass and mollifier smoothing. *Communications on pure and applied mathematics*, 30(5):509–541, 1977.

[98] Byung Kwan Kim, Hyun-Seong Kang, and Seong-Ook Park. Drone classification using convolutional neural networks with merged doppler images. *IEEE Geoscience and Remote Sensing Letters*, 14(1):38–42, 2016.

[99] Taehwan Kim and Tülay Adali. Fully complex multi-layer perceptron network for nonlinear signal processing. *Journal of VLSI signal processing systems for signal, image and video technology*, 32(1):29–43, 2002.

[100] Youngwook Kim and Hao Ling. Human activity classification based on micro-doppler signatures using a support vector machine. *IEEE transactions on geoscience and remote sensing*, 47(5):1328–1337, 2009.

[101] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[102] Reinmar J Kobler, Jun-ichiro Hirayama, and Motoaki Kawanabe. Controlling the fréchet variance improves batch normalization on the symmetric positive definite manifold. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3863–3867. IEEE, 2022.

[103] Reinmar J Kobler, Jun-ichiro Hirayama, Qibin Zhao, and Motoaki Kawanabe. Spd domain-specific batch normalization to crack interpretable unsupervised domain adaptation in eeg. *arXiv preprint arXiv:2206.01323*, 2022.

[104] Oleg A Krasnov and Alexander G Yarovoy. Radar micro-doppler of wind turbines: simulation and analysis using rotating linear wire structures. *International Journal of Microwave and Wireless Technologies*, 7(3-4):459–467, 2015.

[105] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[106] Jakob Krzyston, Rajib Bhattacharjea, and Andrew Stark. High-capacity complex convolutional neural networks for i/q modulation classification. *arXiv preprint arXiv:2010.10717*, 2020.

[107] Rikard Laxhammar. Anomaly detection for sea surveillance. In *2008 11th international conference on information fusion*, pages 1–8. IEEE, 2008.

[108] Alice Le Brigant. *Probability on the spaces of curves and the associated metric spaces via information geometry; radar applications.* PhD thesis, Université de Bordeaux, 2017.

[109] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[110] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[111] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*, 2018.

[112] Henry Leung and Simon Haykin. The complex backpropagation algorithm. *IEEE Transactions on signal processing*, 39(9):2101–2104, 1991.

[113] Nadav Levanon. *Radar principles.* John Wiley & Sons, 1988.

[114] Nadav Levanon and Eli Mozeson. *Radar signals.* John Wiley & Sons, 2004.

[115] Xiaoyu Li, Buyue Qian, Jishang Wei, An Li, Xuan Liu, and Qinghua Zheng. Classify eeg and reveal latent graph structure with spatio-temporal graph convolutional neural network. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 389–398. IEEE, 2019.

[116] Xuejun Liao, Zheng Bao, and Mengdao Xing. On the aspect sensitivity of high resolution range profiles and its reduction methods. In *Record of the IEEE 2000 International Radar Conference [Cat. No. 00CH37037]*, pages 310–315. IEEE, 2000.

[117] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[118] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[119] Yabo Liu, Yi Liu, and Cheng Yang. Modulation recognition with graph convolutional network. *IEEE Wireless Communications Letters*, 9(5):624–627, 2020.

[120] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[121] Aaron Lou, Isay Katsman, Qingxuan Jiang, Serge Belongie, Ser-Nam Lim, and Christopher De Sa. Differentiating through the fréchet mean. In *International Conference on Machine Learning*, pages 6393–6403. PMLR, 2020.

[122] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[123] P. C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India*, 2(1):49–55, 1936.

[124] Atefeh Mahdavi and Marco Carvalho. A survey on open set recognition. In *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 37–44. IEEE, 2021.

[125] MATLAB. *version 9.11.0 (R2021b)*. The MathWorks Inc., Natick, Massachusetts, 2021.

[126] Maxime W. Matthès, Yaron Bromberg, Julien de Rosny, and Sébastien M. Popoff. Learning and avoiding disorder in multimode fibers. *Phys. Rev. X*, 11:021060, Jun 2021.

[127] Ammar Mian, Antoine Collas, Arnaud Breloy, Guillaume Ginolhac, and Jean-Philippe Ovarlez. Robust low-rank change detection for multivariate sar image time series. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:3545–3556, 2020.

[128] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[129] Nina Miolane, Matteo Caorsi, Umberto Lupo, Marius Guerard, Nicolas Guigui, Johan Mathe, Yann Cabanes, Wojciech Reise, Thomas Davies, António Leitão, et al. Iclr 2021 challenge for computational geometry & topology: Design and results. *arXiv preprint arXiv:2108.09810*, 2021.

[130] Nina Miolane, Nicolas Guigui, Alice Le Brigant, Johan Mathe, Benjamin Hou, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, Hatem Hajri, Yann Cabanes, Thomas Gerald, Paul Chauchat, Christian Shewmake, Daniel Brooks, Bernhard Kainz, Claire Donnat, Susan Holmes, and Xavier Pennec. Geomstats: A python package for riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223):1–9, 2020.

[131] P Molchanov, K Egiazarian, J Astola, RIA Harmanny, and JJM de Wit. Classification of small uavs and birds by micro-doppler signatures. In *2013 European Radar Conference*, pages 172–175. IEEE, 2013.

[132] Michel Moruzzis, Pierre Saulais, Tan Hock Tat, and Tong Cherng Huei. Automatic target classification for naval radars. *Military Technology*, 29(7):91–94, 2005.

[133] Fred E Nathanson. Radar design principles. *Signal processing and the environment*, 1991.

[134] Duc Tam Nguyen, Zhongyu Lou, Michael Klar, and Thomas Brox. Anomaly detection with multiple-hypotheses predictions. In *International Conference on Machine Learning*, pages 4800–4809. PMLR, 2019.

[135] Frank Nielsen. An elementary introduction to information geometry. *Entropy*, 22(10):1100, 2020.

[136] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2019.

[137] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[138] Timothy J O'Shea, Johnathan Corgan, and T Charles Clancy. Convolutional radio modulation recognition networks. In *International conference on engineering applications of neural networks*, pages 213–226. Springer, 2016.

[139] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[140] Xavier Pennec. Barycentric subspace analysis on manifolds. *The Annals of Statistics*, 46(6A):2711–2746, 2018.

[141] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A riemannian framework for tensor computing. *International Journal of computer vision*, 66(1):41–66, 2006.

[142] Harry Pratt, Bryan Williams, Frans Coenen, and Yalin Zheng. Fcnn: Fourier convolutional neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 786–798. Springer, 2017.

[143] Samiur Rahman and Duncan A Robertson. Radar micro-doppler signatures of drones and birds at k-band and w-band. *Scientific reports*, 8(1):1–11, 2018.

[144] Samiur Rahman and Duncan A Robertson. Classification of drones and birds using convolutional neural networks applied to radar micro-doppler spectrogram images. *IET radar, sonar & navigation*, 14(5):653–661, 2020.

[145] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*, 2017.

[146] Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan. A simple fix to mahalanobis distance for improving near-ood detection. *arXiv preprint arXiv:2106.09022*, 2021.

[147] Oliver Rippel, Patrick Mertens, Eike König, and Dorit Merhof. Gaussian anomaly detection by modeling the distribution of normal data in pretrained deep features. *IEEE Transactions on Instrumentation and Measurement*, 70:1–13, 2021.

[148] Oliver Rippel, Patrick Mertens, and Dorit Merhof. Modeling the distribution of normal data in pre-trained deep features for anomaly detection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6726–6733. IEEE, 2021.

[149] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[150] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[151] Lukas Ruff, Robert A Vandermeulen, Billy Joe Franks, Klaus-Robert Müller, and Marius Kloft. Rethinking assumptions in deep anomaly detection. *arXiv preprint arXiv:2006.00339*, 2020.

[152] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep semi-supervised anomaly detection. In *International Conference on Learning Representations*, 2020.

[153] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Lucas Deecke, Shoaib A. Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4393–4402, 2018.

[154] Mohammad Sabbaqi, Riccardo Taormina, Alan Hanjalic, and Elvin Isufi. Graph-time convolutional autoencoders. In *Learning on Graphs Conference*, 2022.

[155] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pages 4–11, 2014.

[156] Mohammadreza Salehi, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *arXiv preprint arXiv:2110.14051*, 2021.

[157] Artsiom Sanakoyeu, Vadim Tschernezki, Uta Buchler, and Bjorn Ommer. Divide and conquer the embedding space for metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 471–480, 2019.

[158] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[159] Natasa Sarafijanovic-Djukic and Jesse Davis. Fast distance-based anomaly detection in images using an inception-like autoencoder. In *International Conference on Discovery Science*, pages 493–508. Springer, 2019.

[160] Andrew M Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *Icml*, 2011.

[161] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[162] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human brain mapping*, 38(11):5391–5420, 2017.

[163] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[164] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[165] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[166] Merrill Skolnik. *Radar handbook*. McGraw-Hill Publishing Company, 2nd edition, 1990.

[167] Merrill Ivan Skolnik. *Introduction to radar systems*. Electrical engineering series. McGraw-Hill, 3rd edition, 2001.

[168] Graeme E Smith, Karl Woodbridge, and Chris J Baker. Template based micro-doppler signature classification. In *2006 IET Seminar on High Resolution Imaging and Target Classification*, pages 127–144. IET, 2006.

[169] Stefan Sommer, François Lauze, Søren Hauberg, and Mads Nielsen. Manifold valued statistics, exact principal geodesic analysis and the effect of linear approximations. In *European conference on computer vision*, pages 43–56. Springer, 2010.

[170] Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Iberoamerican congress on pattern recognition*, pages 117–124. Springer, 2013.

[171] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[172] Ingo Steinwart, Don Hush, and Clint Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 6(2), 2005.

[173] Andriyan Bayu Suksmono and Akira Hirose. Beamforming of ultra-wideband pulses by a complex-valued spatio-temporal multilayer neural network. *International Journal of Neural Systems*, 15(01n02):85–91, 2005.

[174] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[175] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[176] David MJ Tax and Robert PW Duin. Growing a multi-class classifier with a reject option. *Pattern Recognition Letters*, 29(10):1565–1570, 2008.

[177] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. 2018.

[178] Kim Phuc Tran, Anh Tuan Mai, et al. Anomaly detection in wireless sensor networks via support vector data description with mahalanobis kernels and discriminative adjustment. In *2017 4th NAFOSTED Conference on Information and Computer Science*, pages 7–12. IEEE, 2017.

[179] Mark Tygert, Joan Bruna, Soumith Chintala, Yann LeCun, Serkan Piantino, and Arthur Szlam. A mathematical motivation for complex-valued convolutional networks. *Neural computation*, 28(5):815–825, 2016.

[180] European Union. A strategic compass for security and defence. `https://www.eeas.europa.eu/eeas/strategic-compass-security-and-defence-1_en`, 2022. Accessed: 20/11/2022.

[181] Stefan Van Aelst and Peter Rousseeuw. Minimum volume ellipsoid. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):71–82, 2009.

[182] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[183] JH Van Vleck. The absorption of microwaves by oxygen. *Physical Review*, 71(7):413, 1947.

[184] JH Van Vleck. The absorption of microwaves by uncondensed water vapor. *Physical Review*, 71(7):425, 1947.

[185] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[186] Santiago Velasco-Forero and Jesus Angulo. Robust rx anomaly detector without covariance matrix estimation. In *2012 4th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–4. IEEE, 2012.

[187] Santiago Velasco-Forero, Marcus Chen, Alvina Goh, and Sze Kim Pang. Comparative analysis of covariance matrix estimation for anomaly detection in hyperspectral images. *IEEE journal of selected topics in signal processing*, 9(6):1061–1073, 2015.

[188] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[189] Rosemarie Velik. Discrete fourier transform computation using neural networks. In *2008 International Conference on Computational Intelligence and Security*, volume 1, pages 120–123. IEEE, 2008.

[190] Cédric Villani. Donner un sens à l'intelligence artificielle - pour une stratégie nationale et européenne. `https://www.aiforhumanity.fr/pdfs/9782111457089_Rapport_Villani_accessible.pdf`, 2018. Accessed: 20/11/2022.

[191] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine learning*, pages 1096–1103, 2008.

[192] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[193] Simon Wagner and Winfried Johannes. Target detection using autoencoders in a radar surveillance system. In *2019 International Radar Conference (RADAR)*, pages 1–5. IEEE, 2019.

[194] Jinwei Wan, Bo Chen, Bin Xu, Hongwei Liu, and Lin Jin. Convolutional neural networks for radar hrrp target recognition and rejection. *EURASIP Journal on Advances in Signal Processing*, 2019(1):5, 2019.

[195] Jue Wang and Anoop Cherian. Gods: Generalized one-class discriminative subspaces for anomaly detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8201–8211, 2019.

[196] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

[197] Wilhelm Wirtinger. Zur formalen theorie der funktionen von mehr komplexen veränderlichen. *Mathematische Annalen*, 97(1):357–375, 1927.

[198] Piotr Iwo Wójcik and Marcin Kurdziel. Training neural networks on high-dimensional data using random projection. *Pattern Analysis and Applications*, 22(3):1221–1231, 2019.

[199] Yan Xia, Xudong Cao, Fang Wen, Gang Hua, and Jian Sun. Learning discriminative reconstructions for unsupervised outlier removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1519, 2015.

[200] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[201] Jiancheng Yang, Rui Shi, and Bingbing Ni. Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 191–195. IEEE, 2021.

[202] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

[203] Le Yang. *Medians of probability measures in Riemannian manifolds and applications to radar target detection.* PhD thesis, Université de Poitiers, 2011.

[204] Le Yang, Marc Arnaudon, and Frédéric Barbaresco. Riemannian median, geometry of covariance matrices and radar target detection. In *The 7th European Radar Conference*, pages 415–418. IEEE, 2010.

[205] Di Yao, Chao Zhang, Zhihua Zhu, Jianhui Huang, and Jingping Bi. Trajectory clustering via deep representation learning. In *2017 international joint conference on neural networks (IJCNN)*, pages 3880–3887. IEEE, 2017.

[206] Ozal Yildirim, Ru San Tan, and U Rajendra Acharya. An efficient compression of ecg signals using deep convolutional autoencoders. *Cognitive Systems Research*, 52:198–211, 2018.

[207] Kaicheng Yu and Mathieu Salzmann. Second-order convolutional neural networks. *arXiv preprint arXiv:1703.06817*, 2017.

[208] Yijun Zuo and Robert Serfling. General notions of statistical depth function. *Annals of statistics*, pages 461–482, 2000.

RÉSUMÉ

Les radars Doppler pulsés (RDP) de surveillance aérienne ont pour mission de discriminer des cibles en se basant sur le signal réfléchi de petites rafales d'impulsions modulées. Les antennes tournantes imposent un faible nombre d'impulsions pour caractériser le contenu de cases distance, celles-ci résultant d'une discrétisation radiale et azimutale. Cette caractérisation est tirée de signaux courts échantillonnés à la période de répétition des impulsions (PRI), un échantillon par impulsion étant disponible dans la rafale transmise. Le nombre d'impulsions et la PRF, qui changent constamment dans un radar en opération, définissent donc la résolution et les fréquences extrêmales du spectre Doppler qui définit les cibles à l'échelle d'une case distance. L'avènement de drones petits et bon marché impose l'amélioration de la détection des cibles petites et lentes qui pouvaient auparavant se retrouver rejetées en tant que fouillis. Cette thèse propose une chaîne de traitement branchée après l'étape de détection d'un RDP pour discriminer les hits afin de permettre l'abaissement des seuils de détection. Cette chaîne de traitement se divise en deux étapes: un encodage *hit2vec*, et une étape de discrimination. L'encodage traite une matrice à valeurs complexes et de taille variable pour produire un vecteur *embedding* à valeurs réelles de taille fixe. Cette représentation d'entrée contient un signal I/Q réfléchi par la case distance porteuse d'une détection enrichi par le signal réfléchi par les cases distance voisines. L'étape de discrimination met en oeuvre une classification mono-classe sur les représentations d'entrée encodées dans le but de séparer les cibles dans un contexte de faible supervision. La chaîne de traitement complète s'appuie donc sur l'hypothèse que le voisinage de spectres Doppler, même à faible résolution, contient l'information nécessaire à la discrimination. Autrement dit, la solution mise en avant se base sur l'apprentissage de représentation pour faire face à l'hétérogénéité des signaux I/Q qui doivent être séparés et pour permettre à une discrimination à faible supervision en aval de produire une distance vis-à-vis de cibles de référence utile à l'opérateur radar.

MOTS CLÉS

Discrimination de cibles radar, encodage de signal, fouillis radar, détection d'anomalie, classification mono-classe, apprentissage semi-supervisé, micro-Doppler, réseau de neurones à valeurs complexes, réseau de neurones pour graphes, appprentissage profond géométrique, apprentissage de représentation, apprentissage profond, radar, traitement du signal.

ABSTRACT

Air surveillance pulse Doppler radars (PDR) need to discriminate targets using the backscatter generated by small bursts of modulated pulses. Rotating antennas constrain the number of pulses available to characterize the content of range cells, the latter resulting from azimuthal and radial discretization. This characterization is based on short signals sampled at the pulse repetition frequency (PRF), one sample being available per pulse in the transmitted burst. The number of pulses and the PRF, which constantly change in an operating radar, thus define the resolution and the maximal frequencies of the Doppler spectrum that defines targets within a range cell. With the advent of small and cheap drones, air surveillance radars are required to improve their detection of small and slow targets that previously had chances to end up discarded as clutter. This thesis proposes a processing pipeline plugged after the detection stage of a PDR to discriminate between hits in order to allow for the lowering of detection thresholds. This processing pipeline is made of two steps: an encoding *hit2vec* step, and a discrimination step. The encoding handles an input complex-valued matrix of varying size and outputs a real-valued vector embedding of fixed size. This input contains the backscattered I/Q signal of the range cell carrying a detection enriched by the backscatter from the neighboring range cells. The discrimination step applies a one-class classification to the encoded inputs to separate targets representations in a low-supervision context. The complete pipeline is therefore based on the assumption that the neighborhood of Doppler spectrums, even at low resolutions, contains the required discriminatory information. Said in other words, the solution put forward uses representation learning to tackle the sampling heterogeneity of the I/Q signals that ought to be separated, and to enable a subsequent low-supervision discrimination to provide a radar operator with a useful distance to reference radar targets.

KEYWORDS

Radar targets discrimination, signal encoding, radar clutter, anomaly detection, out-of-distribution detection, one-class classification, semi-supervised learning, micro-Doppler, complex-valued neural network, graph neural network, geometric deep learning, representation learning, deep learning, radar, signal processing.