DEEP LEARNING-DRIVEN LANDMINE DETECTION: GENERALIZING REAL-WORLD DATA FROM SYNTHETIC GPR TRAINING SETS

LOUIS BROUWER

Under supervision of Dr. Ourania Patsia

Bachelor of Engineering

School of Engineering

The University of Edinburgh

2024

Memento mori

You can't connect the dots looking forward
- Steve Jobs

PERSONAL STATEMENT

This research project, conducted from January 14, 2024, to April 24, 2024, explored novel areas of Ground Penetrating Radar (GPR) and Machine Learning (ML). Despite my lack of prior experience in this field, limited coding skills, and minimal background in numerical modelling and geophysics, I was drawn to these topics out of great interest. The project was fundamentally supported by Dr. Ourania Patsia, who developed the numerical model for the research antenna, and Dr. Iraklis Giannakis, who provided the numerical PMN landmine model. Dr. Patsia also assisted with GPR data collection and provided a script for data extraction from proprietary formats. The theoretical frameworks for Chapters 2 and 3 were adapted from existing literature, while the original contributions in Chapter 6 were primarily my own, supplemented by referenced literary sources. The project progressed smoothly for the most part and closely followed the planning structure developed for the interim report. Broadly, the project can be broken down into two parts: numerical modelling and machine learning. The numerical modelling half went well and all objectives were achieved. I faced challenges with the implementation of a 3D object rotation and scaling script, but ultimately overcame these and delivered an elegant solution to the data augmentation for 3D models. The second half of the project focused on the ML aspect of the research, which was initially an extremely rewarding process that enabled me to leverage the power of this technique. However, significant challenges were faced with the generalisation of the ML algorithm to real data, a central objective of this research upon which the hypothesis relied. To address this, I theorised a two-stage training technique, which prevented model overfitting and allowed generalisation to real data.

I hereby declare that this thesis and the work reported herein, unless mentioned explicitly, was composed and originated entirely by myself, under the supervision of Dr. Ourania Patsia in the School of Engineering at The University of Edinburgh.

Edinburgh, October 9, 2024

Louis Brouwer

Deep Learning-Driven Landmine Detection: Generalizing Real-World Data from Synthetic GPR Training Sets

Louis Brouwer October 9, 2024

Background: Ground-Penetrating Radar (GPR) is a vital non-invasive technology used in civil engineering and geophysical surveys. It is particularly useful in humanitarian demining due to its ability to detect polymers, a common material in landmines, which are invisible to metal detectors. Traditional GPR interpretation, however, is prone to user error and slow processing times, hindering its efficacy for landmine detection.

Aims: This thesis aims to develop a Machine Learning (ML) model trained exclusively on synthetic GPR data capable of accurately generalizing to real-world scenarios. The objective is to overcome the limitations of sparse real data and demonstrate that synthetic training can effectively prepare ML models for practical detection tasks.

Methods: We employed numerical modeling techniques, specifically Finite-Difference Time-Domain (FDTD) methods, to generate synthetic datasets simulating GPR scans over landmines. The ML model was trained using these datasets, with a focus on optimizing model parameters to enhance its ability to generalize from synthetic to real-world data. Validation was performed using real GPR data collected from a controlled environment containing various known targets.

Results: The trained model demonstrated high accuracy in classifying both synthetic and real GPR datasets, correctly identifying landmine signatures while minimizing false positives.

Discussion: The findings underscore the potential of ML models trained on synthetic data to effectively translate to real-world applications. This approach not only mitigates the challenges associated with the scarcity of real GPR data but provides foundations in the opportunity of the operational efficiency and safety of landmine detection.

Conclusions: The research confirms that ML models can be successfully trained on synthetic data to generalize to real-world GPR signals, offering a promising new direction for humanitarian demining efforts. This work lays the groundwork for future studies to explore more complex scenarios and a wider variety of landmine types.

ACKNOWLEDGEMENTS

Firstly I would like to express my sincere gratitude to my Supervisor Dr. Ourania Patsia for her constant support, knowledge and positivity for the duration of the entire project. During this short but impactful last three months of my degree I could not have wished for a more capable supervisor with seemingly endless patience and kindness. At times of discouraging results a meeting with Rania would equip me with the insight and attitude to achieve. Secondly I would like to give a special acknowledgement to my second marker Dr. Parvez Alam who over the last year has become my closest experience to having an academic mentor. Parvez's curiosity and research attitude is infectious and inspires me to do my best work, I hope these are just the early years of what can become an enduring friendship. I would like to thank Prof. Antonis Giannopoulos for sharing such an interest in my project. The two meetings we had during the project were encouraging and reinforced my sense of purpose in the project. Your friendly demeanor and big-picture thinking related to the field of Ground Penetrating Radar has given me a lasting passion and respect for the field. Thank you to my flatmates: Sam, Johnnie and Pierce for providing me with a cosy home to which I could return after late nights. And thank you to Sofia, who has shared the front seat with me during the entire ride.

To my family; my lovely parents Dirk and Elisabeth, who throughout my life have provided me with nothing but love, opportunity, and support. My lovely sisters Lilly and Annebé with whom I can laugh and share reflections with. To my brother Machiel, the person who gives me the confidence to be the best version of myself.

ABSTRACT

Ground-Penetrating Radar (GPR) is a commonly applied remote sensing technique that relies upon the transmission and reception of electromagnetic (EM) waves to perform underground object detection. It is primarily applied in civil engineering and geophysics. The power of GPR to 'see' underground makes itself an ideal candidate for the field of humanitarian demining. Landmines are commonly made from polymers which would be impossible to detect using techniques that rely on metal-detection. GPR functions by 'seeing' dielectric contrasts between buried objects which enables object detection.

The interpretation of these GPR signals presents an entirely separate challenge as one must be able to classify underground objects into landmines (targets) and other objects (false-alarms). The accurate interpretation of the signals is a well-covered field of study. However these conventional processing techniques suffer from some key drawbacks related to landmine detection. These are namely the ability for misinterpretation by the user and a non-instantaneous response. Both of which prevent practical application to hand-held landmine detection. Machine Learning (ML) presents an attractive potential solution to this problem due to its proven ability to solve problems which required the mapping of complex relationships. To function properly, ML models must be trained on large dataset such that they can 'learn' these relationships through experience. Large datasets of real GPR responses for landmines do not exist however and would be nearly impossible to create, especially considering the variety of landmines available and the diverse range of soil environments found.

Numerical modelling facilitates the solution to this with the creation of large synthetic datasets. GPR modelling most commonly relies on Finite-Difference Time-Domain solvers to solve Maxwell's equations for different modelled scenarios and thus simulate the physics of the modelled environment and produce data for a training set. The first half of this research focuses on applying cutting edge techniques and applications in the field of numerical modelling of GPR systems. It makes use of the most recent and accurately developed antenna and target models. The 'digital twin' of the GSSI 2GHz antenna was used in this research to improve model accuracy, the semi-empirical model was used to add Debye-dispersion properties to the model to enhance realism. Real data was collected using the real GSSI transducer over a sand box which contained a variety of targets and false alarms to be used for validating the ML scheme.

The synthetic data was created by using the University of Edinburgh: Infrastructure Sensing Laboratory Haggis Server which is equipped with several GPUs,

which are useful for running large simulations. A two-stage training process was implemented for the training and tuning of the ML scheme. This model was trained entirely on synthetic data and was able to correctly classify not only the synthetic examples within its own test set with a high degree of accuracy, but was also able to generalise and correctly classify the real examples recorded from the sand box with a high degree of accuracy. This presents a key breakthrough in demonstrating the ML schemes can be trained solely on realistic synthetic data and accurately generalise to complex targets such as landmines.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Antipersonnel
BP	Backpropagation
CNN	Convolutional Neural Network
DL	Deep Learning
EM	Electromagnetic
FDTD	Finite-Difference Time-Domain
FEM	Finite Element Method
FETD	Finite Element Time-Domain
FWI	Full-Waveform Inversion
GAN	Generative Adversarial Network
GA	Genetic Algorithms
GHZ	GigaHertz
GPR	Ground Penetrating Radar
GUI	Graphical User Interface
ICA	Independent Component Analysis
kNN	k-Nearest Neighbors
MHZ	MegaHertz
MIL	Multiple Instance Learning
ML	Machine Learning
MM	Method of Moments
MTI	Microwave Tomographic Inversion
NATO	North Atlantic Treaty Organization

NN	Neural Network
PFA	Probability of False Alarm
PCA	Principal Component Analysis
PD	Probability of Detection
PEC	Perfect Electric Conductors
PMC	Perfect Magnetic Conductors
PSO	Particle Swarm Optimisation
RAM	Random-Access Memory
ROC	Receiver Operating Characteristic
RPCA	Robust Principal Component Analysis
RTM	Reverse Time Migration
SCR	Signal-to-Clutter Ratio
SNR	Signal-to-noise ratio
SVD	Singular Value Decomposition
SVM	Support Vector Machines
TNT	Trinitrotoluene
1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional

1 INTRODUCTION

1.1 MOTIVATION AND AIMS OF THE THESIS

Ground-penetrating radar (GPR) is a non-destructive technique used in various fields, where electromagnetic (EM) waves are emitted and a reflection is received when there is a contrast in dielectric properties. It is especially useful in demining for detecting metallic but also non-metallic mines, which traditional metal detectors miss [9]. The challenge lies in interpreting the diverse minefield environments that complicate signal analysis. This thesis suggests employing a Machine Learning (ML) model to automate GPR data interpretation, distinguishing landmines from other buried objects. Due to limited real-world data, the research will explore the training of these ML models on synthetic datasets created via numerical modelling. The models will have enhanced realism though the implementation of novel 3D antenna and landmine models [10, 4]. Leveraging advanced Artificial Neural Networks (ANNs), the ML model will be trained on synthetic data and validated against both its own test set and real GPR data. The aim is the development of a neural network which is able to classify real data whilst being trained only on synthetic data with the objective of advancing knowledge in the field of humanitarian demining through the integration of the latest GPR and ML technologies and methods [6].

1.2 OBJECTIVES

Outlined below are the key objectives of this research

1. To develop a comprehensive understanding of GPR technology and its application in the detection of subsurface objects, with a focus on its role in demining operations.

2. To investigate and document the challenges associated with the use of GPR in diverse minefield environments, including but not limited to variations in soil composition, vegetation cover, and the presence of extraneous metallic objects.

3. To design and synthesise a robust realistic synthetic dataset for training ML models by employing numerical modelling [4] with advanced simulation tools

like gprMax [11], ensuring the inclusion of realistic soil media, landmine, and antenna responses to EM waves.

4. To enhance the realism of the synthetic training data by integrating a newly developed antenna model and/or 3D landmine models.

5. To implement advanced ML techniques within the project, to develop an ML model capable of effectively distinguishing landmines from irrelevant subsurface objects the synthetic data training set.

6. To evaluate the performance of the trained ML model using actual GPR data, aiming to validate the model's efficacy in accurately identifying landmines, thereby contributing to safer and more efficient demining operations.

1.3 HYPOTHESIS

It is possible to develop a ML model that is trained solely on Synthetic A-scan training examples modelled in such a way that the algorithm can generalise to real data.

2

GROUND PENETRATING RADAR: PRINCIPLES AND MODELLING

2.1 INTRODUCTION

GPR uses high-frequency EM waves in the microwave region to detect subsurface objects without excavation, widely utilised in civil engineering and landmine detection [9, 12, 4]. Operating as an 'active' geophysical technique, GPR emits ultra-wideband EM pulses via a transmitting antenna and captures reflections with a receiving antenna. These reflections occur at interfaces where dielectric properties differ, aiding the detection of subsurface features. However, soil non-homogeneity and lack of sharp dielectric contrasts often complicate signal analysis. Figure 2.1 illustrates how debris and soil permittivity variations scatter EM waves, diminishing reflected wave amplitude and increasing the signal-to-clutter ratio (SCR). Additional challenges include water and other high conductivity media that attenuate and slow signals, affecting operational depth and complicating interpretation. The effective frequency range of GPR spans from 10 MHz to 3 GHz, with higher frequencies needed for small targets like landmines but also causing increased attenuation [1, 5].

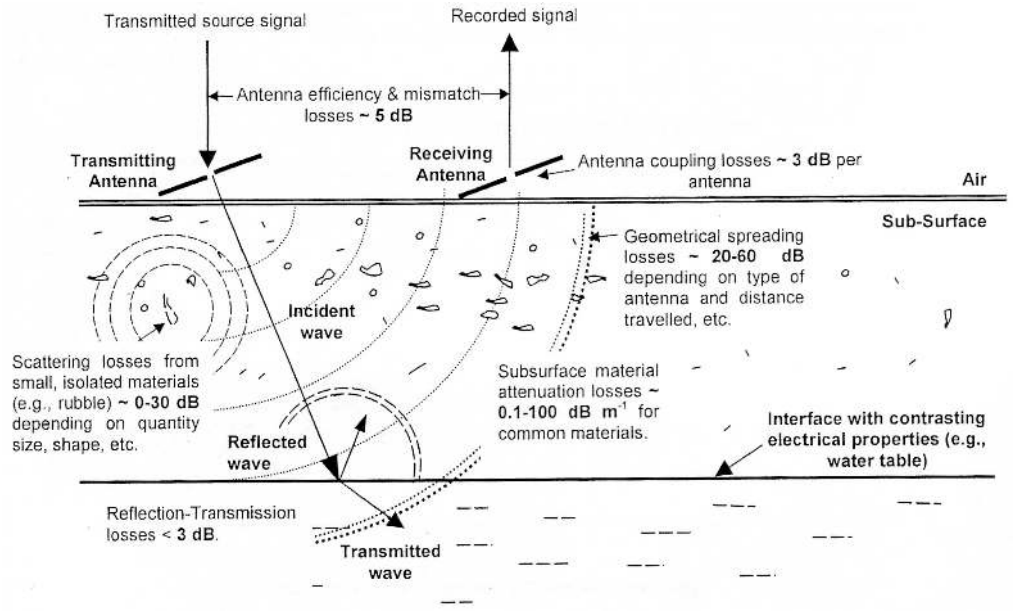


Figure 2.1: A diagram showing the typical losses such as scattering and geometric spreading losses that occur during GPR [1].

2.2 THEORY OF GPR

2.2.1 *Propagation Theory*

Although sharing many similarities with conventional radar, GPR waves, a subset of EM waves, are used primarily for the detection of underground rather than above-surface objects. The behaviour of GPR waves is primarily influenced by the conductivity and permittivity of the soil. When the GPR wave encounters a contrast between these properties in the soil that it is travelling through, part of the energy of the wave is scattered and reflected whilst the rest continues onwards. Most materials are non-magnetic and have a near free space permeability, but certain materials with high magnetic content, such as haematite or magnetite, should have permeability taken into account. The speed of a wave in a lossless material is well understood to be the relationship between the speed of light $c \approx 3 \cdot 10^8$ m/s and the relative permittivity ϵ_r given as:

$$u = \frac{c}{\sqrt{\epsilon_r}} \tag{2.1}$$

Most materials encountered however are lossy dielectrics. The velocity of the EM wave in a lossy material is given by Balanis [13].

$$u = \frac{c}{\sqrt{\frac{\mu_r \epsilon_r}{2}\left(\sqrt{1+\left(\frac{\sigma}{\omega}\right)^2}+1\right)}} \tag{2.2}$$

Where μ_r is the relative permeability, σ is the electric conductivity (S/m) and ω is the angular frequency (rad/s).

In lossy materials it is the conductivity which is responsible for the attenuation of the wave. As the wave travels through such a medium, energy is lost as heat. Other losses, as illustrated in Figure 2.1, are geometric spreading losses, scattering losses caused by reflections from small subsurface features, and losses caused by reflections from the antenna, within itself, and off the ground.

Electric permittivity is a property of materials which expresses the energy stored within a material when exposed to an electric field. It is usually given as a relative quantity with respect to the free-space permittivity $\varepsilon_0 = 8.854 \times 10^{-12}\,\mathrm{F/m}$

$$\varepsilon_r = \frac{\varepsilon}{\varepsilon_0} \tag{2.3}$$

The relative permittivity is a complex frequency-varying property within materials, it also varies greatly with material properties especially water content. It is often simplified to a static value [14]. Similarly, the magnetic permeability,

which represents the magnetic response of materials to a magnetic field, is also a relative quantity μ_r, given relative to the free space value $\mu_0 = 4\pi \times 10^{-7}\,\mathrm{H/m}$ as

$$\mu_r = \frac{\mu}{\mu_0} \tag{2.4}$$

The dielectric properties of soil, including permittivity ϵ, conductivity σ, and permeability μ, are largely determined by its linearity, isotropy, homogeneity, and dispersivity, which are often treated as scalar in analyses. Scalar permittivity suggests material polarization changes proportionally and instantaneously with an applied field. This however is a simplification. Homogeneity is similarly a simplification, as soil complexity demands detailed modelling to reflect true heterogeneity. Material properties usually vary with the applied field's frequency, making them dispersive, a crucial factor for accurate simulation [3].

As mentioned, materials are rarely homogenous and are often characterised by bulk properties. Table 2.1 shows some of these properties for commonly found materials.

Material	$\boldsymbol{\varepsilon_r}$	$\boldsymbol{\sigma}$ **(mS/m)**	$\boldsymbol{u}$ **(m/ns)**
Air	1	0	0.3
Clay (dry)	2–20	1–100	0.07–0.21
Clay (wet)	15–40	100–1000	0.05–0.08
Concrete (dry)	4–10	1–10	0.09–0.15
Concrete (wet)	10–20	10–100	0.07–0.09
Fresh water	81	0.1–10	0.03
Fresh water ice	3–4	1	0.15–0.17
Granite (dry)	5–8	0.001–0.00001	0.11–0.13
Granite (wet)	5–15	1–10	0.08–0.13
Limestone (dry)	4–8	0.001–0.0000001	0.11–0.15
Limestone (wet)	6–15	10–100	0.08–0.12
Sand (dry)	4–6	0.001–1	0.12–0.15
Sand (wet)	10–30	0.1–10	0.05–0.09
Sea water	81	4000	0.03
Sea water ice	4–8	10–100	0.11–0.15
Soil (average)	16	5	0.08

Table 2.1: Selected material properties [7].

The operating frequency range of GPR systems ranges from around 10 MHz to 3GHz [15]. Higher frequency waves are characterised as being lossy and lack ability to detect deeply buried items. Lower frequency fields are diffusive in character, and lack the resolution to detect small objects. GPR systems emit waves not just at a single frequency, but across a range of frequencies. This range is known as the bandwidth, where the central frequency in this range is known as the center frequency f_c. It is this frequency that is used when referring to the frequency of a transducer, such as the GSSI 2GHz [16].

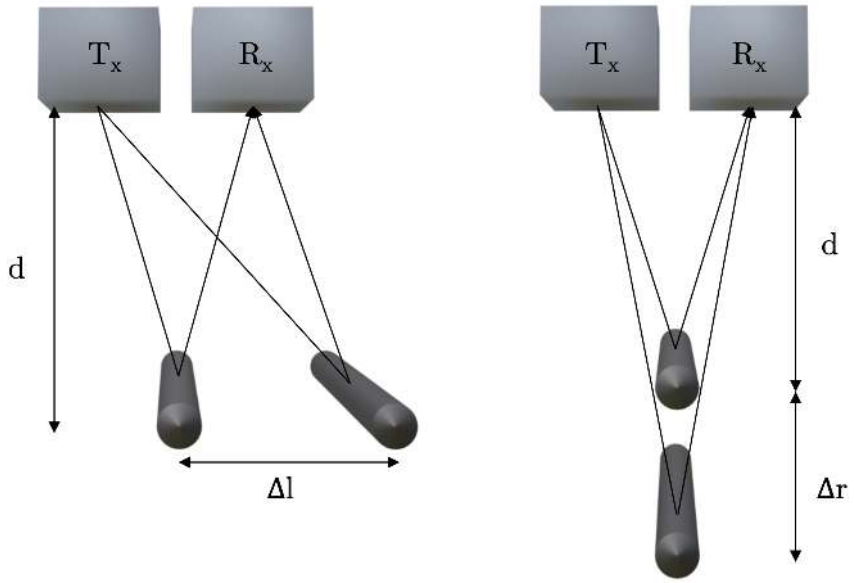


Figure 2.2: Left: The required horizontal separation of targets as a factor of d and δl Right: The required vertical separation of targets as a factor of d and δl [2].

As mentioned, within GPR there exists a trade-off between resolution and depth. Higher frequencies can detect smaller targets at shallower depth but lack penetration and vice-versa for lower frequencies. In any arrangement, the resolution determines the minimum size of targets that can be detected and the distance between two targets in order to be registered as two separate events in the reflected trace. In the case of two identical targets, these must be separated in time by half of their pulse width W or greater, defined as the width at half amplitude [2]

$$W = \frac{1}{B} = \frac{1}{f_c} \tag{2.5}$$

Translating this into the spatial domain, for the horizontal resolution l and vertical resolution r, the separations are given by:

$$\Delta l \geq \sqrt{\frac{udW}{2}} \tag{2.6}$$

$$\Delta r \geq \frac{uW}{4} \tag{2.7}$$

Figure 2.2 shows how Δl and Δr vary according with distance d.

2.2.2 *GPR Data Collection*

The reflected signal received by the transceiver is a trace of voltage against time. This trace for a single pulse is called an A-scan and provides information about the local properties in one dimension (1D). If several of these A-scans are collected at regular intervals one can create a B-scan, which is the two dimensional (2D) equivalent of an A-scan. This process is illustrated in Figure 2.3. C-scans are the three dimensional (3D) equivalent of A-scans and require multiple B-scans taken at regular intervals across a second dimension perpendicular to that used to create the B-scans.

2.2.3 *The Finite-Difference Time-Domain Method*

To construct a training dataset for a Deep Learning (DL) algorithm, one can either gather extensive real-world data or generate synthetic data through numerical modelling. In GPR, numerical modelling is critical for both performance evaluation and data synthesis for ML applications [1, 17]. This process involves simulating EM wave dynamics—propagation, reflection, scattering—and solving Maxwell's equations under defined boundary conditions. Simplistic scenarios might use analytical solutions [13], but complex environments require numerical methods due to their intricate nature. Among these methods, the Method of Moments (MoM) [18] calculates fields across soil boundaries and is effective for limited boundary models [5]. Frequency domain methods like the Finite Element Method (FEM) [19] are suited for fixed-frequency modelling but need significant RAM, similar to the Finite Element Time-Domain (FETD) approach.

The Finite-Difference Time-Domain (FDTD) method [20] directly solves Maxwell's equations using second-order derivatives, avoiding linear system resolutions and enabling broad frequency coverage [21]. FDTD is ideal for simulating PEC, PMC, and dispersive media, and has been effectively used in complex soil simulations with varied textures and vegetation since its initial use in antenna simulations by Bourgeois and Smith in 1997 [22, 4]. However, FDTD is not without its challenges, notably the constraint between time and spatial discretisation, an issue mitigable by employing methods like the Crank-Nicholson method

Figure 2.3: a) Several A-scans which are collected in sequence at equal spacing are oriented vertically and plotted b) Displaying the voltages as pixel intensities allows for the creation of a B-scan [3].

[23] or otherwise that adhere to the Courant-Friedrich-Lewy (CFL) condition. The comprehensive body of work utilising this method [4, 17, 6, 24], due to the ability of FDTD to realistically model many different complex properties of soil, underpins the selection of FDTD for this research.

2.3 MODELLING TECHNIQUES FOR GPR

Accurate modelling of the dispersive properties of soils can be achieved through Jonscher functions [25] or Cole-Cole functions [26], [27], though direct application of these functions in the FDTD code is constrained [4]. To address this limitation, approximations such as multi-Debye relaxations [28, 29] have been integrated into FDTD schemes. Pade approximations serve as an alternative to multi-Debye, yet multi-Debye is often preferred due to its computational efficiency [4]. Despite the successful application of multi-Debye functions in modelling homogeneous media [30], [31], this approach does not fully account for the complex heterogeneity characteristic of most soils, which are frequently composed of a diverse mix of materials.

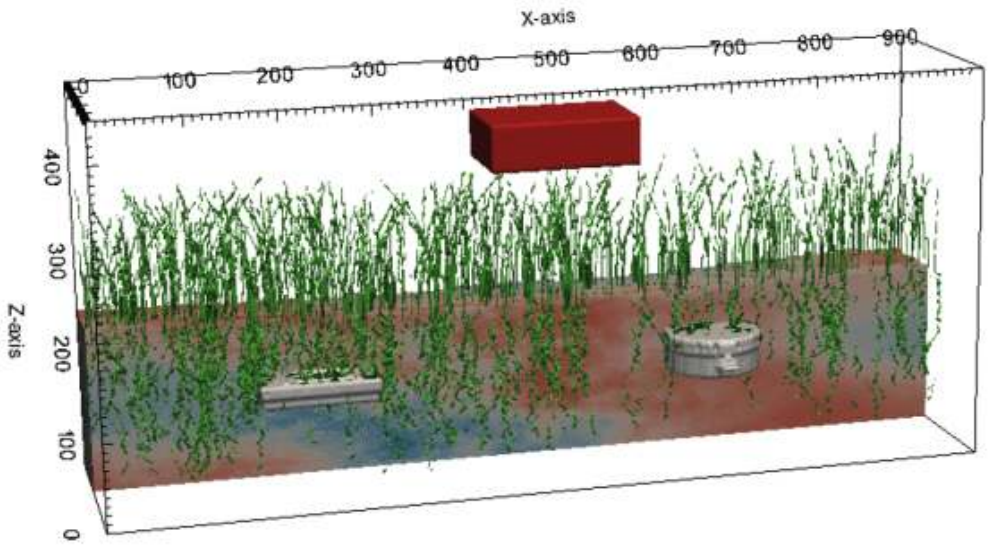


Figure 2.4: Cross-section of an example of a complex model created for [4]. The model includes both PMA and PMN landmines and a stochastically varying soil moisture distribution in the range 0-0.25. The red areas indicate lower moisture and the blue areas indicate higher areas of moisture.

A viable approach to this challenge is the adoption of the semi-empirical model [32], particularly in its revised form [33], which has proven effective in studies [4]. This model, akin in purpose and function to the Cole-Cole function, can be approximated using multi-Debye expansions for compatibility with FDTD schemes. The semi-empirical model leverages easily understood soil properties—such as soil fraction, clay fraction, and bulk density—to deduce the soil's dielectric properties [4].

Surface roughness as included in Figure 2.4 also significantly impacts the efficacy of GPR [9, 34]. Given the necessity for landmines to be buried, they are predominantly located in natural settings where perfectly flat soil is uncommon. This variability affects not only the reflections of GPR waves but also

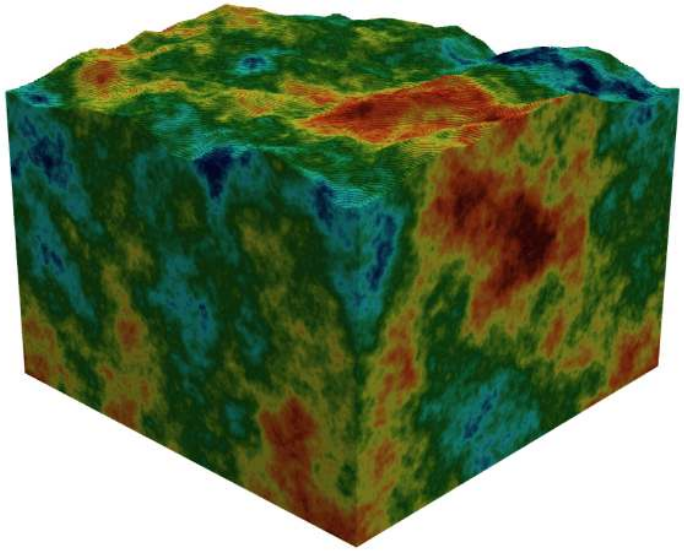


Figure 2.5: (a) A model showing a stochastic distribution of an arbitrarily chosen property of the soil such as sand fraction or water volumetric density.

the positioning of the antenna. Simulating surface roughness can be effectively achieved through the use of fractal-correlated noise [35]. Fractals can also be used to simulate stochastic distributions of material properties within soil models as illustrated in Figure 2.5. Figure 2.6 shows the calculated semivariogram for the soil in Figure 2.5 compared to a Gaussian semivariogram which shows that this method is adequate for the simulation of stochastic distributions. This method is substantiated by literature indicating this to accurately represent natural soil distributions [36].

2.3.1 *Antenna and Target Modelling*

In the realm of demining, various antennas are employed, as outlined in Table 2.2, showcasing different governmental strategies. These include both hand-held and vehicle-mounted types with assorted sizes and configurations. For instance, the HSTAMIDS program features a triangular trio of antennas [37], while another setup employs a single transmitter paired with four receivers (1Tx+4Rx), enhancing informational output but also increasing signal interference [38, 39]. While accurate, FDTD-compatible antenna models are scarce, simulated versions exist for commercial models like the GSSI 1.5 GHz and MALÅ Geoscience 1.2 GHz. These models are suitable for detecting shallow, small landmines due to their high-frequency operations [5, 17, 1].

Numerical modelling, especially via the FDTD method, is vital for antenna design and analysis, enhancing data interpretation [10, 40, 41]. Effective modelling requires accurate depictions of the antenna's geometry, the dielectric properties

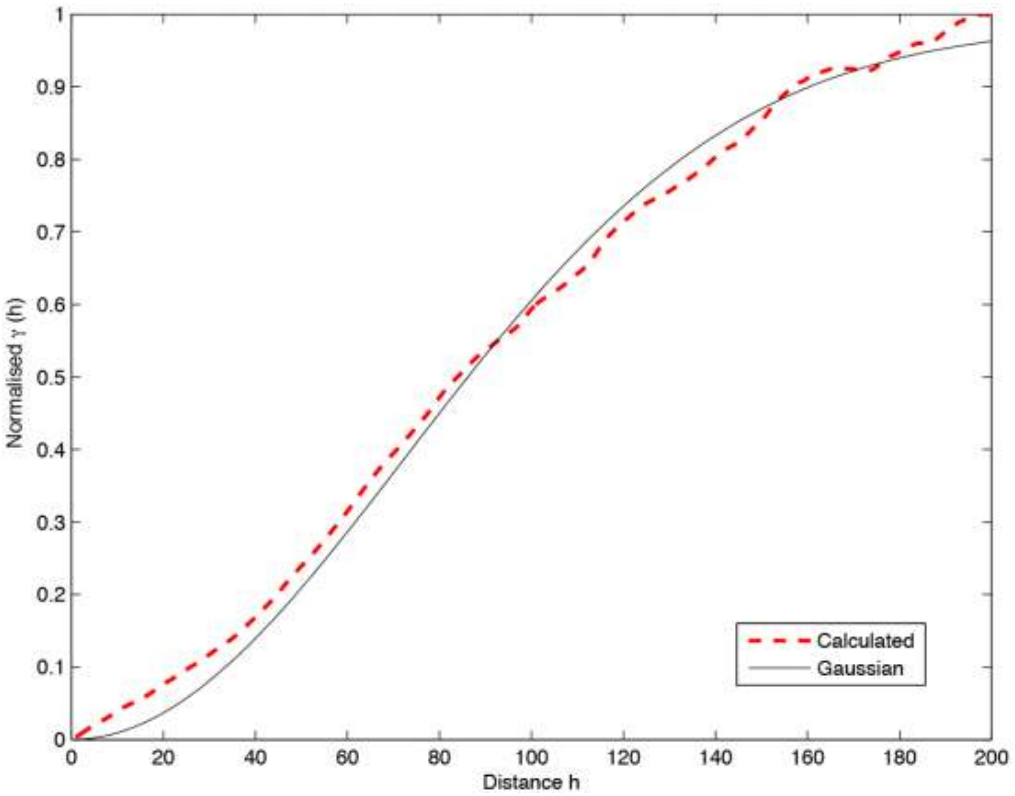


Figure 2.6: (b) The calculated semivariogram of the stochastically varying soil property compared to a simulated Gaussian semivariogram [4].

of its components, and the excitation pulse characteristics. A 'digital twin' of the GSSI-2GHz antenna illustrates this approach, employing genetic algorithms (GA) and particle swarm optimisation (PSO) to estimate component dielectric properties, confirmed by matching synthetic A-scans with those from physical tests involving PEC plates and rebars in concrete [42]. This model's fidelity and the antenna's suitability for high-frequency applications underscore its utility in advanced research settings.

(a) (b)

Figure 2.7: (a) The real PMN landmine (b) The numerical equivalent of the PMN landmine developed in [5].

Country	Program	Type	Maturity
Australia	HILDA	H	medium
	RRMS	V	high
Belgium	HUDEM	H	low
Canada	ILDP	V	high
	GEODE	V	low
	LOTOS	V	low
EU	DEMINE	H	low
	MINEREC	H	low
	HOPE	H	low
	PICE	H	low
France	SALMANDER	V	medium
Germany	MMSR	V	medium
Israel	ELTA	V	high
Japan	MEXTSNENCION	H	high
Sweden	PICE	H	medium
	MINETECT	H	high
UK	DCMC	H	medium
	MCMC	V	medium
USA	HSTAMIDS	H	high
	GSTAMIDS	V	low

Table 2.2: Various landmine detection programs being pursued globally [8].

In addition to antenna design, FDTD has been employed in target modelling. Landmines, for example, have been effectively modelled using this method. As illustrated in Figure 2.7, both real and numerically derived models of PMN landmines have been developed and validated according to the findings in [5]. Figure 2.8 shows the reflected wave received by the antenna for both the real and synthetic targets. However, this validation was performed under simplified conditions where the mine was not buried in a medium. Additionally an inferior 1.5GHz antenna model was used compared to the 2GHz model selected for this research.

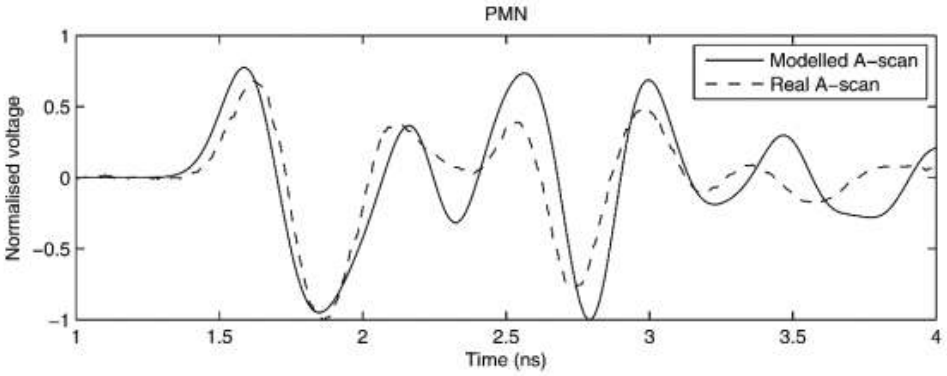


Figure 2.8: The recorded reflections from the real PMN landmine (dashed) and modelled PMN landmine (continuous) [4].

2.4 CONCLUSIONS

This chapter has provided a brief introduction to GPR theory and critically reviewed GPR applications and prevalent modelling approaches, emphasising the challenges of dielectric variability and wave attenuation by moisture. The chapter discusses modelling heterogeneous and the modelling of complex soil dielectric properties, highlighting the use of multi-Debye approximations for the semi-empirical model. Also covered is the impact of surface roughness on GPR. Antenna modelling is examined and validated the selection of the GSSI 2 GHz antenna for this research.

3

MACHINE LEARNING: PRINCIPLES AND APPLICATIONS IN GPR AND LANDMINE DETECTION

3.1 INTRODUCTION

Artificial intelligence (AI) rapidly advances and enhances various applications by developing machines with 'intelligence' capable of performing tasks typically requiring human intellect. ML, a subset of AI, is defined by Mitchel as "a computer program learns from experience E with respect to tasks T and performance measure P if its performance at T, as measured by P, improves with E" [43]. GPR, when applied to landmine detection, encounters significant challenges in complex environments like Cambodia's rainforests. These challenges stem from signal attenuation in saturated soils and variability that compromises signal clarity [9, 44]. Currently, the war in Ukraine has resulted in it becoming the most heavily landmine-contaminated area globally [45]. There is a high demand for demining technologies, yet significant challenges persist in their deployment. ML has shown promising results in applications requiring the inference of complex relationships from large datasets, which supports the implementation of ML in this research [17, 46].

3.2 THEORY

Essentially, ML algorithms learn from experience by capturing mathematical relationships between the features of examples (inputs) and if available their corresponding labels (outputs). ML is categorized into two principal classes: supervised and unsupervised learning. Unsupervised learning uses unlabelled data to discover hidden patterns, often for clustering or dimensionality reduction. This research, however, will utilise supervised learning, where the input features $\mathbf{x}^{(i)}$ are paired with labels $y^{(i)}$. For binary classification, such as distinguishing between landmines and false targets, labels are defined as $y^{(i)} \in \{0, 1\}$, with 0 typically representing false targets and 1 representing landmines. ML can be effectively applied to a broad variety of problems, such as:

1. Regression: Where the algorithm predicts a continuous value based on the input. This algorithm is popular in price prediction applications. Another

example would be to predict the depth of landmines based on the input trace.

2. Denoising: In this case, ML is utilised to clean a noisy signal. An advanced implementation of this was used in [47] to remove noise from GPR traces.

3. Classification: Where the program classifies an example amongst k classes of outputs. In landmine detection, binary classification ($k = 2$) is a popular application of this type of ML where one class would represent targets and the other would represent false-alarms.

The objective of training Artificial Neural Networks (ANNs) is to minimise the loss function, a measure of the differences between the predictions made by the model and the actual outcomes. Key loss functions include mean squared error for regression tasks and cross-entropy for classification tasks. Training typically involves several epochs, with each epoch representing a complete pass of the training data through the model. Critical elements in training include the learning rate, which dictates the size of steps taken towards minimising the error, and the optimiser, which adjusts model parameters to reduce loss. The Adam optimiser is preferred for its ability to adaptively modify learning rates, improving training efficiency. ANNs utilise activation functions such as ReLU and sigmoid to introduce non-linearity, essential for learning complex patterns; ReLU returns the input if it's positive and zero otherwise, while sigmoid is useful for binary classifications as it maps inputs into a 0-1 range.

Training data are structured into a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, where each row, $\mathbf{x}^{(i)}$, corresponds to the i-th example with n features. This format enhances efficient, parallel processing of data. Accurate feature selection is critical, as irrelevant features can mislead the model and impair its performance, such as mistakenly including patient numbers in a cancer classification dataset.

For classification tasks, model performance is primarily assessed by accuracy, the proportion of correctly classified cases. While high training accuracy is desirable, it does not necessarily predict effective real-world performance, as the model may not generalise well to new, unseen data. Thus, models are also evaluated on separate test sets to measure their generalisation capabilities. Low training accuracy usually results in low test accuracy, a phenomenon known as underfitting, where the model fails to capture any consistent mathematical relationship between the input features and the labels. Conversely, overfitting occurs when a model performs exceptionally well on training data but poorly on test data, often learning to recognise noise as part of the actual signal. Various machine learning algorithms are employed, with ANNs being particularly popular in deep learning contexts.

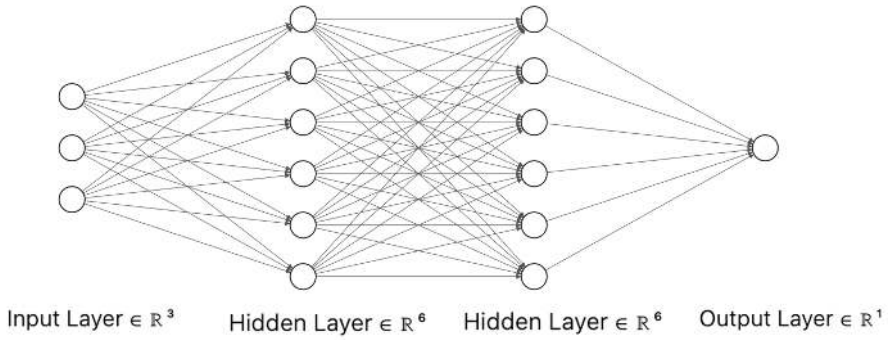


Figure 3.1: Example of a small neural network consisting of an input layer with 3 nodes, two 6-layer hidden nodes, and a singular output node.

ANNs are computing structures which are inspired by biological systems that make up the brains of animals. ANNs 'learn' similarly to other ML methods by being exposed to many examples. ANNs are often applied when the relationship between the input and output is complex and non-linear. Figure 3.1 shows how ANNs are built from many interconnected nodes which are arranged into layers. The nodes in the input layer represent the input features of the training examples whilst the output layer represents the prediction. Each node has an associated weight **W** and the objective of the NN is to find the weight of each node that corresponds to the highest accuracy.

Figure 3.2 shows a two dimensional binary classification problem, where the two dimensions specify the two axes of input features and the binary nature is represented by the red and green colour. The right images show the only partially resolved feature spaces with progressively increasing number of training examples for each row. The image highlights how a lack of data will still allow for a solution to be developed which satisfies the training data but can't generalise to the entire feature space. The image in the bottom right does not contain all the data, but contains sufficient amounts for the algorithm to develop a solution which can accurately represent all cases [48].

3.3 MACHINE LEARNING IN GPR

ML has significantly advanced GPR applications in object detection, classification, and signal processing. Gamba et al. [49] used feed-forward NNs to detect hyperbolic patterns in B-scan images, training on real data and achieving high accuracy against human benchmarks. Singh et al. [50] applied hyperbola detection in B-scans for estimating target depth and position. Zhang et al. [51] utilised Generative Adversarial Networks (GANs) alongside Convolutional Neural

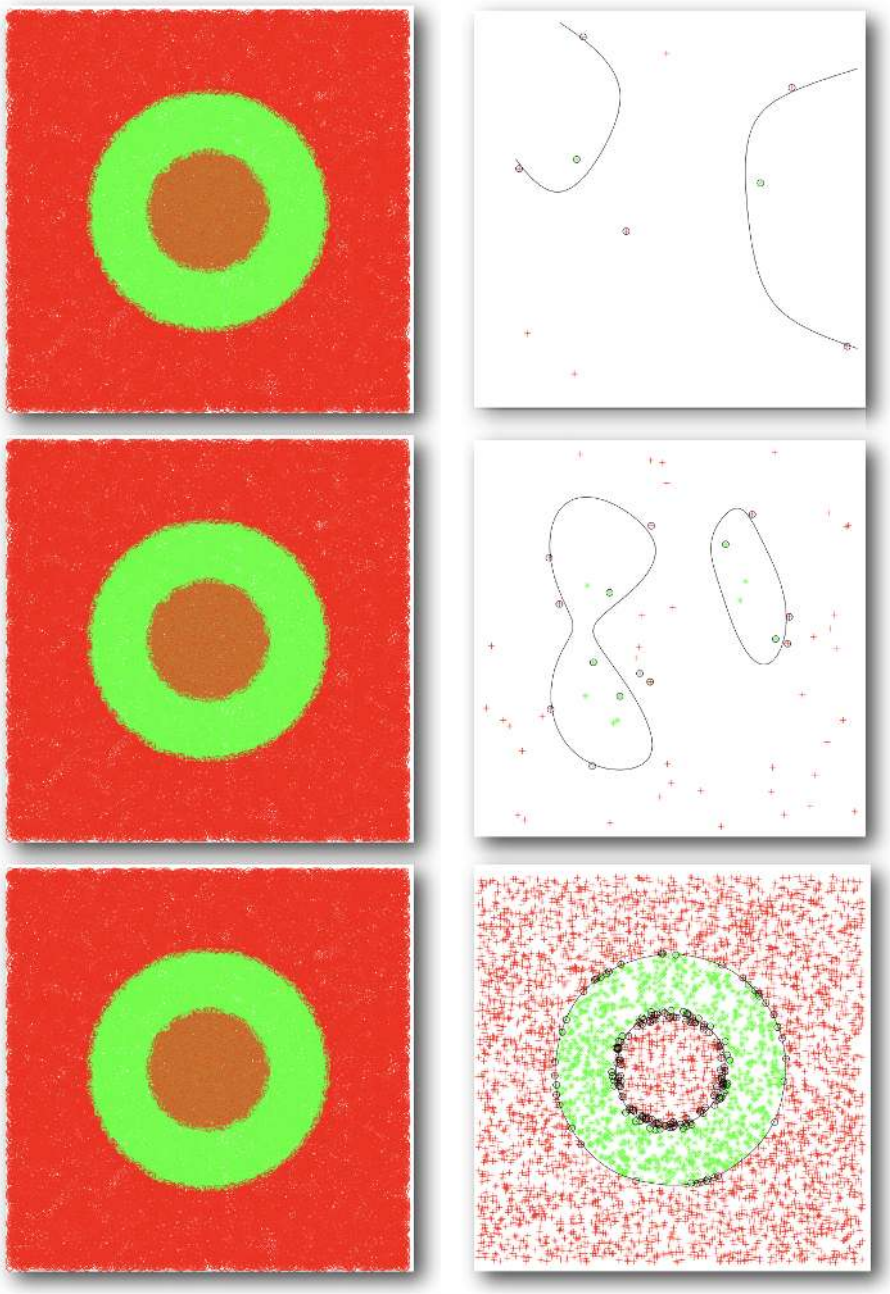


Figure 3.2: An example of a two dimensional binary classification problem. The left images show the full feature space. The right images show the only partially resolved feature spaces with progressively increasing number of training examples for each row. The classes are classified using an SVM. The image shows how a lack of examples will lead to a solution which does not generalise well for the entire feature space [5].

Networks (CNNs) for data augmentation [52]. In 2022, Dai et al. [53] developed a deep learning-based 2-D GPR forward solver that predicts subsurface objects in heterogeneous soil with a mean relative error of 1.18%. Todkar et al. [54] detected debondings in pavements using SVMs. Pham et al. [55] adapted the Faster-RCNN framework for underground detection, training on the Cifar-10 database and synthetic B-scan data from gprMax, although detailed effectiveness analyses were lacking [11].

Recent ML advancements, particularly ANNs, have also improved GPR data processing for clutter removal and signal focusing. Techniques like Time-Gating and Principal Component Analysis (PCA), which attempts to separate signal from clutter, face challenges in complex soil conditions and detecting horizontal targets [56]. ANNs have shown potential in background clutter removal and enhancing target identification over traditional methods [47]. GPR data processing often involves pre-processing steps like eliminating direct air and ground wave responses and signal focusing via migration and medium velocity calculation through hyperbola fitting to reduce the SCR [9].

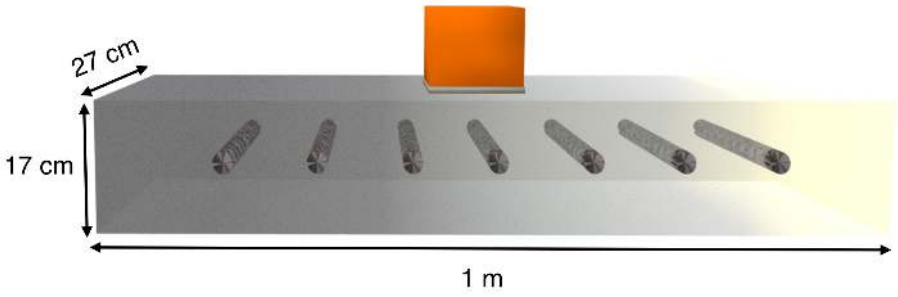


Figure 3.3: 3D visualisation of the model, showing the linearly varying permittivity [6].

Patsia et al. [6] demonstrated an approach that involves using ANNs to predict and subtract background clutter, and to estimate permittivity for enhancing signal response with Reverse Time Migration (RTM) and find the depth to targets as shown in Figure 3.4. This research uses A-scan data, preferred for ML training over B-scans and C-scans demonstrated to be favourable for training ML schemes in [57, 6].

The scheme, effective on both synthetic and real data, identified every target, outperforming traditional methods [6]. It focused on rebar and metal plates such as the example shown in Figure 3.3, differing from the PMN and PMA landmines targeted in this research. The models used linearly varying permittivity, simplifying the complex variability found in natural soils.

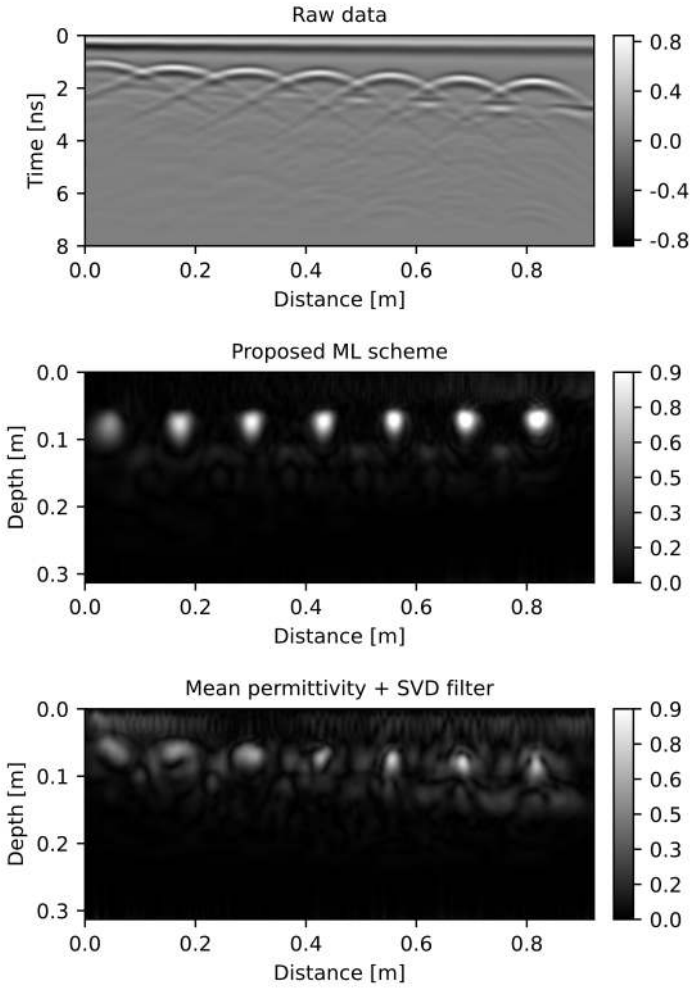


Figure 3.4: Top: raw data, Middle: RTM output produced by the ML scheme, Bottom: RTM output produced using SVD [6].

3.3.1 *Data Processing*

GPR data interpretation often requires pre-processing to remove 'background clutter,' essential for both conventional and ML-based methods to enhance outcomes [9, 17]. Numerous signal processing and reconstruction techniques exist, including Microwave Tomographic Inversion (MWT), which treats imaging as solving an inverse scattering problem and excels in detecting the smallest mines compared to Backpropagation (BP) and Stolt migration [4, 58, 59, 60].

Time-gating reduces the initial GPR signal to diminish direct air and ground waves but risks omitting relevant traces [61]. Principal Component Analysis (PCA) and its robust variant (RPCA) separate signal from clutter by prioritizing subcomponents, showing significant improvements in Receiver Operating Characteristic (ROC) performance for landmine detection [62, 63, 64]. Similarly, Singular Value Decomposition (SVD) and Independent Component Analysis

(ICA) remove horizontal reflections and have been used in GPR signal processing [65, 66, 56, 67]. Despite their effectiveness, these methods are case-sensitive and performance depends heavily on model parameters, antenna properties, and target characteristics. They also struggle to detect horizontal targets due to their nature of removing repetitive features [4, 6].

3.4 MACHINE LEARNING IN LANDMINE DETECTION

Numerous examples demonstrate ML's application in landmine detection. Unlike traditional signal processing methods such as hyperbola fitting with the Hough transform [68], Lameri et al. [24] employ an ML approach based on feature extraction from GPR B-scan traces. This method minimises pre-processing, removing only direct antenna paths, less extensive than the pipeline in [6]. It's important to note that [6] does not utilise B-scans, making direct comparison challenging. Additionally, convolutional networks, preferred in image processing ML applications, contrast with dense networks in their application suitability. Incorporating real data into the training set, which increased PD from 83% to 95%, demonstrates this method's effectiveness, though it doesn't reach the accuracy potential of [6] nor the increased SCR. Additionally the paper does not specify the complexity of the training model, thus it is difficult to comment on the applicability of the method to more complex real world situations. In [69], an ANN was trained using data from a single mine lane. Azimi-Sadjadi et al. [70] applied PCA and trained a feed-forward neural network on C-scans. The creation of B-scans and C-sans rely on positioning information and therefore are representative of handheld devices targeted in this research.

In [71], a step-by-step classification process identifies and then classifies targets. This research however only uses data from a single mine lane which prevents conclusions from being drawn about its ability to generalise. Giannakis et al. [17] trained a 2-layer dense NN on a 2D dataset of 4000 examples, incorporating fractals to simulate soil inhomogeneity and a semi-empirical model, enhancing realism [4].The model incorporated realistic false-alarm targets such as bullets. The model used a two-layer feed-forward back-propagation network structure. It was found that increasing the complexity of the ANN did not affect the performance. This could partially be caused by the simplicity of the A-scans and the lack of information per training example. Despite this, the ROC curves obtained from testing the training set indicate the ability of the model to successfully distinguish targets from false alarms with a high degree of accuracy. The author goes on to mention:

'Further work employing realistic 3D models and real data could result in an efficient ANN classification process to be tested in the field.'

3.5 CONCLUSIONS

Chapter 3 begins by providing an overview of ML theory. It then explores literature on GPR data processing and the deployment of ANNs for target detection, highlighting the improvements that ML based systems have over conventional techniques. ML methods have proven effective in reducing background noise and boosting signal clarity, significantly improving the signal-to-noise ratio (SNR). Additionally ANNs have been used for target detection which has been verified with real data. Research shows promising results for the detection of landmines using synthetic data. This study aims to investigate the potential of extending these ML techniques to complex and accurate 3D analyses, and additionally validating these results with real data.

4

MODELLING

4.1 INTRODUCTION

This chapter delineates the methodological framework employed in the modelling of the numerical environment used to create the synthetic dataset, detailing the scientific processes involved. It encompasses a discussion on the forefront of progress within the pertinent field and delineates the research's scope. The design of this research is predicated upon the aims delineated in Chapter 1. The preliminary objectives have been addressed within the context of the literature review. Consequently, this segment will expound upon the methodologies adopted to fulfil the ensuing four objectives.

To fulfil objectives five and six, two distinct endeavors are identified: The validation of the model on its own test set and the validation of this same model on the real data. Prior to addressing objectives five and six, it is imperative to generate an accurate and representative synthetic dataset, using the latest literature available on the subject. As indicated by objectives three and four. The four most pertinent factors to the successful implementation of this are listed as follows:

1. The fidelity of the modeled antenna.
2. The accuracy of the modeled targets and false alarms.
3. The congruence of the synthetic soil media with actual soil media.
4. The specifications and dimensions of the synthetic domain.

4.2 DESIGN AND CONSTRAINTS OF THE MODELED ENVIRONMENT

The environment, as synthetically modeled, is designed to closely parallel—yet not replicate—the actual testing conditions. This approach aims to facilitate the evaluation of ML algorithms against the real test data. The model sought is anticipated to demonstrate reliable predictiveness in the categorisation of real data.

The proof-of-concept nature of this research specifies a controlled environment. This was selected to be the sand box, the same testing environment used in [10]. Since the permittivity and conductivity properties of this environment are known

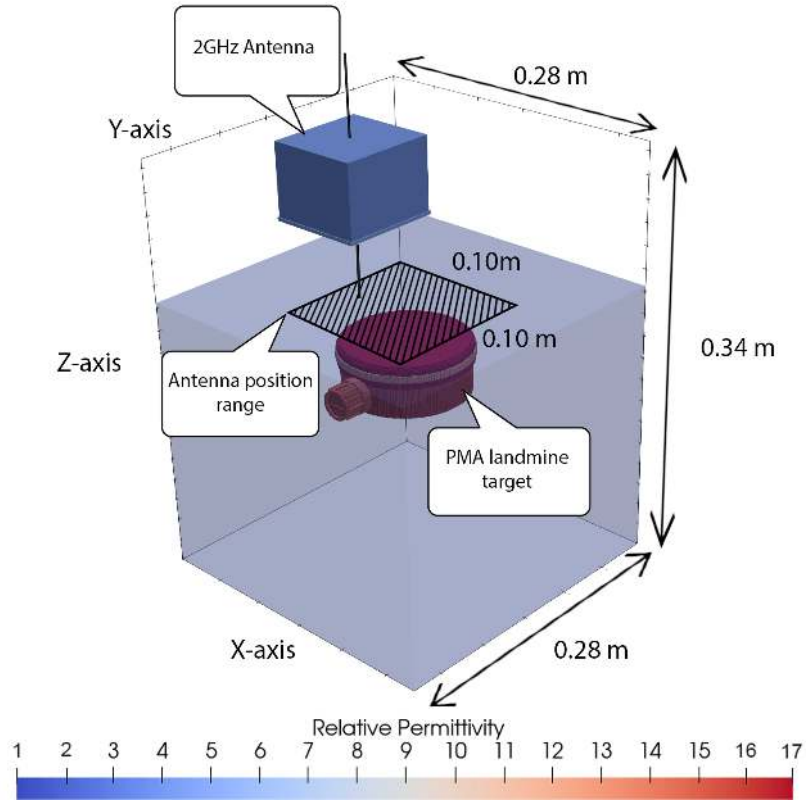


Figure 4.1: The arrangement showing the domain of the numerical scheme used to create the synthetic dataset. Included is the antenna position range. A PMN target and the Antenna itself.

to be $\epsilon_r = 3.5$ and $\sigma_r = 0.0001$, it is possible to design a synthetically modelled environment that includes scenarios similar to the real case but is also diverse in order to maximise the ability to generalise. As shown in [10], the sand within this sand box can be accurately represented using a homogeneous non-dispersive model. Debye dispersion was added, according to the semi-empirical model as specified by [33] and applied in [4] in some scenarios to simulate soils with considerable water content.

4.2.1 *Homogeneous Model*

The training sets in this study are created using a 3D FDTD method with a numerical model of the GSSI 2GHz palm antenna and a PMN landmine target. To enhance model complexity and include potential false alarms, bullet casings and rocks are added to the training set. The domain dimensions are $0.28m \times 0.28m \times 0.34m$, calculated to ensure maximum antenna offset from

center boundaries with a minimum of 20 cells of spacing. The target is centrally placed, while the antenna position varies, whilst maintaining the aforementioned 20 cell spacing from the boundary. Table 4.1 details all varied parameters within the model. Parameter variations occur every 6 scans, with antenna positions changing each scan. Every 6th scan is empty, which facilitates background removal. This setup allows for training sets ranging from one to five scans per example, supporting the training of models that generalise to specific target types and facilitating effective operation of handheld devices without fixed positioning. Figures 6.5 and 6.6 display selected mixed and target-specific examples from the training set.

Parameter	**Value (range)**	**Unit**	**Datum**
Ground position	0.155	m	from top of model
Soil relative permittivity	2.5 - 25	F/m	-
Soil relative conductivity	0.001 - 1	S/m	-
Antenna x-position	0 - 0.05	m	from centre
Antenna y-position	0 - 0.05	m	from centre
Antenna z-position	0.35 - 0.75	m	from surface
Landmine depth	0.005 - 0.012	m	from surface
Rock count	0 - 2	-	-
Rock x-position	0 - 0.055	m	from centre
Rock y-position	0 - 0.055	m	from centre
Rock z-position	0.03 - 0.13	m	from surface
Rock relative permittivity	5.0 - 15	-	-
Rock relative conductivity	0.00001 - 10	m	-
Bullet casing count	0 - 3	-	-
Bullet pitch	-90, 90	deg	-
Bullet yaw	-180, 180	deg	-
Bullet x-position	0 - 0.09	m	from centre
Bullet y-position	0 - 0.09	m	from centre
Bullet z-position	0.015 - 0.012	m	from surface

Table 4.1: Parameters within the sand model and their value ranges.

4.3 ANTENNAS AND TARGETS

4.3.1 *Antennas*

As delineated in Chapter 2, the present research will employ the GSSI 2GHz antenna, a model validated in prior studies [10]. Despite its conventional application in ground-coupled surveys for infrastructure examination—such as detecting embedded metal in masonry or mapping rebar in concrete—it has been effectively utilised for rebar detection within concrete in [10]. The antenna's performance characteristics, including a resolution depth of up to 0.4 m and a central frequency of 2GHz, still render it highly suitable for the objectives of this project.

4.3.2 *Targets*

The selection of targets for this research was limited to the TS-50 and PMN landmines due to project constraints that necessitated the availability of both simulated real targets and numerical equivalents. The PMN landmine, a Russian-manufactured anti-personnel (AP) mine, was chosen for its historical prevalence and continued use in conflict zones, attributed to widespread distribution and extensive stockpiles held by Russian-allied nations during the Cold War [72, 73, 44]. Its substantial explosive charge (240g of TNT) and sizeable dimensions (height 50mm and diameter 115mm) make the PMN a pertinent choice for detection research. AP mines are typically buried between 0 and 0.24m to optimise functionality and detection sensitivity, with some studies testing depths up to 0.1m [74, 75, 5, 4]. This research uses a burial depth range of 0 to 0.15m, chosen to reflect realistic conditions, and mines are placed randomly within this range.

4.4 FALSE ALARMS

Incorporating false alarm targets is essential for the comprehensive validation of any ML-based scheme, as these targets contribute to training models to differentiate between threats and innocuous objects. False alarm targets can manifest in various forms, such as soil anomalies like localised moisture concentrations, rocks or man-made objects like spent bullet casings and crushed cans.

4.4.1 *Bullet Casings*

A primary false alarm target in this study is the spent bullet casings of the 7.62x51 NATO caliber, a size common among NATO forces and similar to cartridges

like the 6.5x53mm, 7.62x54R, 8x50mm, and 7.92x57mm [76]. These casings are modeled as hollow, capped PEC cylinders. Due to their small size and 1mm spatial discretisation in all three dimensions, a high-resolution model is not required, simplifying the geometry to a cylindrical shape. Appendix A.3.1 provides the simulation model integration code. Casings are randomly positioned within the domain with arbitrary pitch and yaw orientations; roll is omitted due to axial symmetry. These orientations determine the casing's end positions, calculated using Equations (4.1) to (4.8).

Random pitch (θ) and yaw (ϕ) angles are generated first, where *Uniform* denotes the Python function for generating a random number within a specified range according to a uniform distribution. The pitch angle ranges from $[-90°, 90°]$ in radians, and the yaw angle from $[-180°, 180°]$, also in radians.

$$\theta = \text{Uniform}\left(\frac{-\pi}{4}, \frac{\pi}{4}\right) \tag{4.1}$$

$$\phi = \text{Uniform}\left(\frac{-\pi}{2}, \frac{\pi}{2}\right) \tag{4.2}$$

The position of one end of the bullet casing $(bpos_x, bpos_y, bpos_z)$ is defined in relation to the model's domain size $(Domain_x, Domain_y)$ and a variable for ground depth ($ground\_depth$).

$$bpos_x = \frac{Domain_x}{2} + \text{Uniform}(-1, 1) \times \left(\frac{Domain_x}{2} - 0.07\right) \tag{4.3}$$

$$bpos_y = \frac{Domain_y}{2} + \text{Uniform}(-1, 1) \times \left(\frac{Domain_x}{2} - 0.07\right) \tag{4.4}$$

$$bpos_z = (ground\_depth + 0.04) + \text{Uniform}(-1, 1) \times 0.03 \tag{4.5}$$

The position of the second end of the bullet casing $(bpos2_x, bpos2_y, bpos2_z)$ is determined based on the bullet model's first position, the pitch and yaw angles, and the length variable set by the user.

$$bpos2_x = bpos_x + \text{length} \cdot \cos(\phi) \cdot \cos(\theta) \tag{4.6}$$

$$bpos2_y = bpos_y + \text{length} \cdot \sin(\phi) \cdot \cos(\theta) \tag{4.7}$$

$$bpos2_z = bpos_z + \text{length} \cdot \sin(\theta) \tag{4.8}$$

The methodology replicates the hollow interior of the casing to simulate additional reflections, despite the fact that EM waves cannot penetrate metals.

This is pertinent when the casing's open end aligns with the wave path, mimicking reflections seen in real false alarms. Although the actual wall thickness of the casing is 0.75mm, it is standardised to 1mm in the model—aligning with the domain's minimum discretisation thickness—to ensure accuracy, especially when the casing is oriented at non-orthogonal angles. Table 4.2 displays training examples featuring these bullet false alarms.

4.4.2 *Rocks*

In addition to bullet casings, rocks are significant false-alarm targets in demining. Represented by asymmetric volumes with diverse sizes, permittivities, and conductivities as detailed in Table 2.1, rocks present a strong addition to the dataset. Three rock models with existing geometry were provided by Dr. Ourania Patsia. For this research a script was developed to dynamically alter the orientation and scale of rock models in three dimensions for each. This introduces nine degrees of augmentation through three axes of positional freedom, employing data augmentation techniques to enhance dataset diversity and improve the model's generalization across new scenarios [77]. Data augmentation has been particularly effective in fields like medical image analysis, where limited data variety can impede model performance. Shorten et al. [78] highlighted these benefits, demonstrating substantial improvements in predictive accuracy. The rock model augmentation uses the Rodrigues matrix for rotations and is implemented in the script found in Appendix A.3.1. The Rodrigues' rotation formula stands as an efficient means to calculate the rotation matrix corresponding to a rotation by an angle θ around a normalised vector $\mathbf{v}$.

Consider a normalised rotation axis represented by the unit vector $\mathbf{v} = (v_x, v_y, v_z)$ and a rotation angle θ. The rotation matrix $\mathbf{R}$ can be expressed as:

$$\mathbf{R} = \mathbf{I} + (\sin\theta)\mathbf{W} + (1 - \cos\theta)\mathbf{W}^2 \tag{4.9}$$

Here, $\mathbf{I}$ denotes the identity matrix and $\mathbf{W}$ signifies the skew-symmetric matrix constructed from vector $\mathbf{v}$, defined as:

$$\mathbf{W} = \begin{pmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{pmatrix} \tag{4.10}$$

The skew-symmetric matrix $\mathbf{W}$ encapsulates the cross product operation. The term $(\sin\theta)\mathbf{W}$ addresses the rotation component perpendicular to the axis, whereas $(1-\cos\theta)\mathbf{W}^2$ pertains to the plane of rotation's component.

To effectuate the rotation of a three-dimensional object or voxel matrix, one applies the rotation matrix $\mathbf{R}$ to each constituent vector (or point) $\mathbf{x}$ through the matrix-vector multiplication $\mathbf{Rx}$, yielding the rotated vector $\mathbf{x}'$:

$$\mathbf{x}' = \mathbf{Rx} \tag{4.11}$$

This rotation application mechanism selected for its computational efficiency and lucid geometric interpretation, rendering it indispensable in simulations and graphical environments where complex rotational transformations are paramount for achieving desired object orientations. However, rotating a 3D volume around its center or an arbitrary point in space, rather than around the origin, requires more than just the Rodrigues rotation matrix. This is where the affine transformation matrix is applied. An affine transformation matrix in three dimensions is a 4×4 matrix used to perform linear transformations including rotation, translation, scaling, and shearing. The affine transformation matrix can be represented as follows:

$$\begin{pmatrix} R & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix} \tag{4.12}$$

where R is a 3×3 Rodrigues rotation matrix for specific rotations, $\mathbf{T}$ is a 3×1 translation vector, $\mathbf{0}$ is a 1×3 zero vector facilitating linear transformations in homogeneous coordinates, and 1 is the identity scalar ensuring homogeneity. The affine matrix allows translating the center of the 3D volume to the origin, applying the rotation, and then translating it back, effectively rotating the volume about its center. The translation vector $\mathbf{T}$ compensates for the shift of the volume's center to and from the origin. This method enables seamless rotation of the volume around its center or any arbitrary point, crucial for data augmentation or 3D volume manipulation, preserving the volume's spatial integrity. Table 4.3 illustrates the insertion of multiple rocks with varied sizes, scales, and properties into the model.

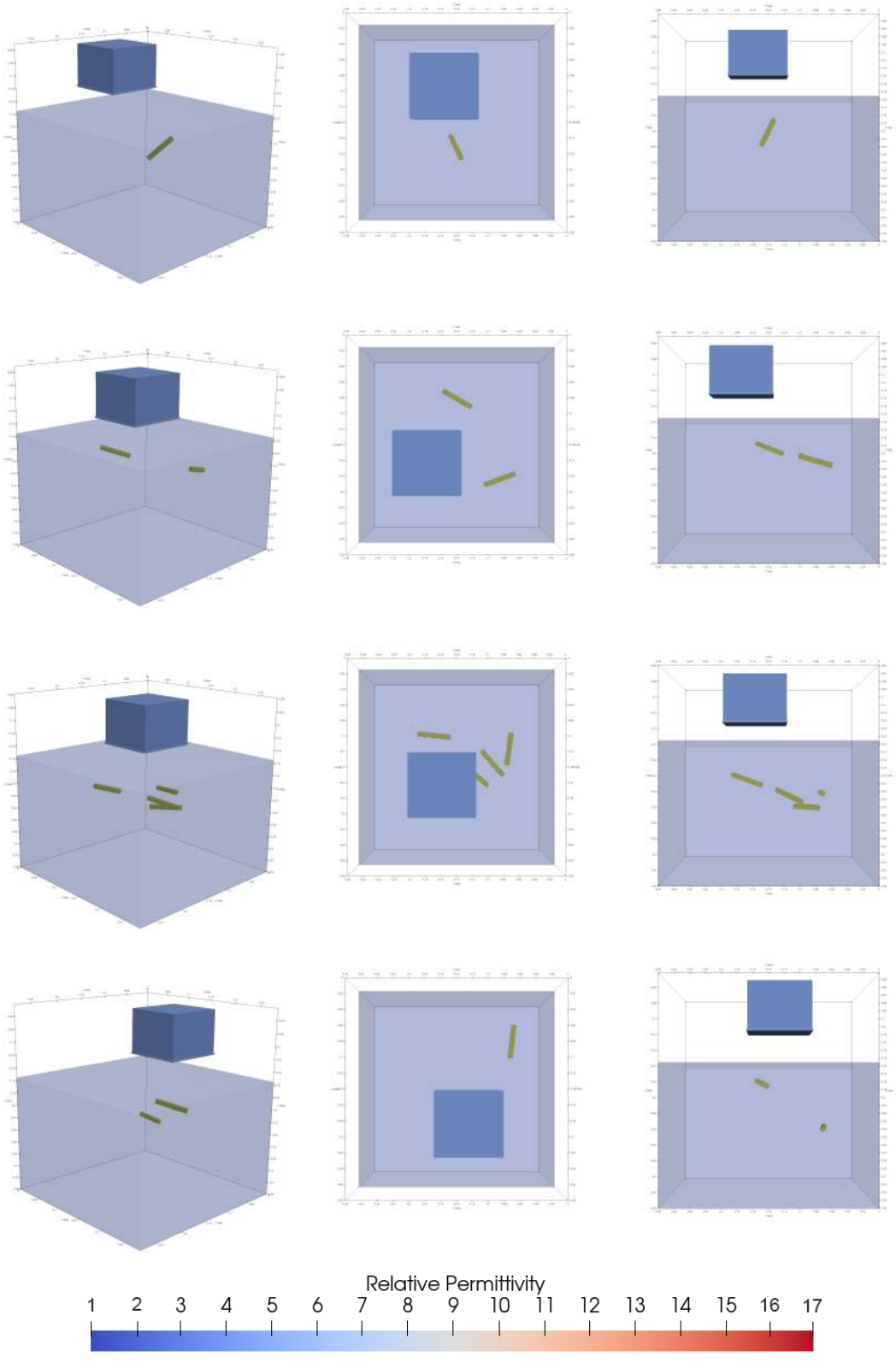


Figure 4.2: Four different false-alarm training examples containing bullet casings with randomly selected positions and orientations. Each row shows a different training example and each column shows three different orientations. Column 1 shows isometric views, column 2 shows top-down views, and column 3 shows side-on views.

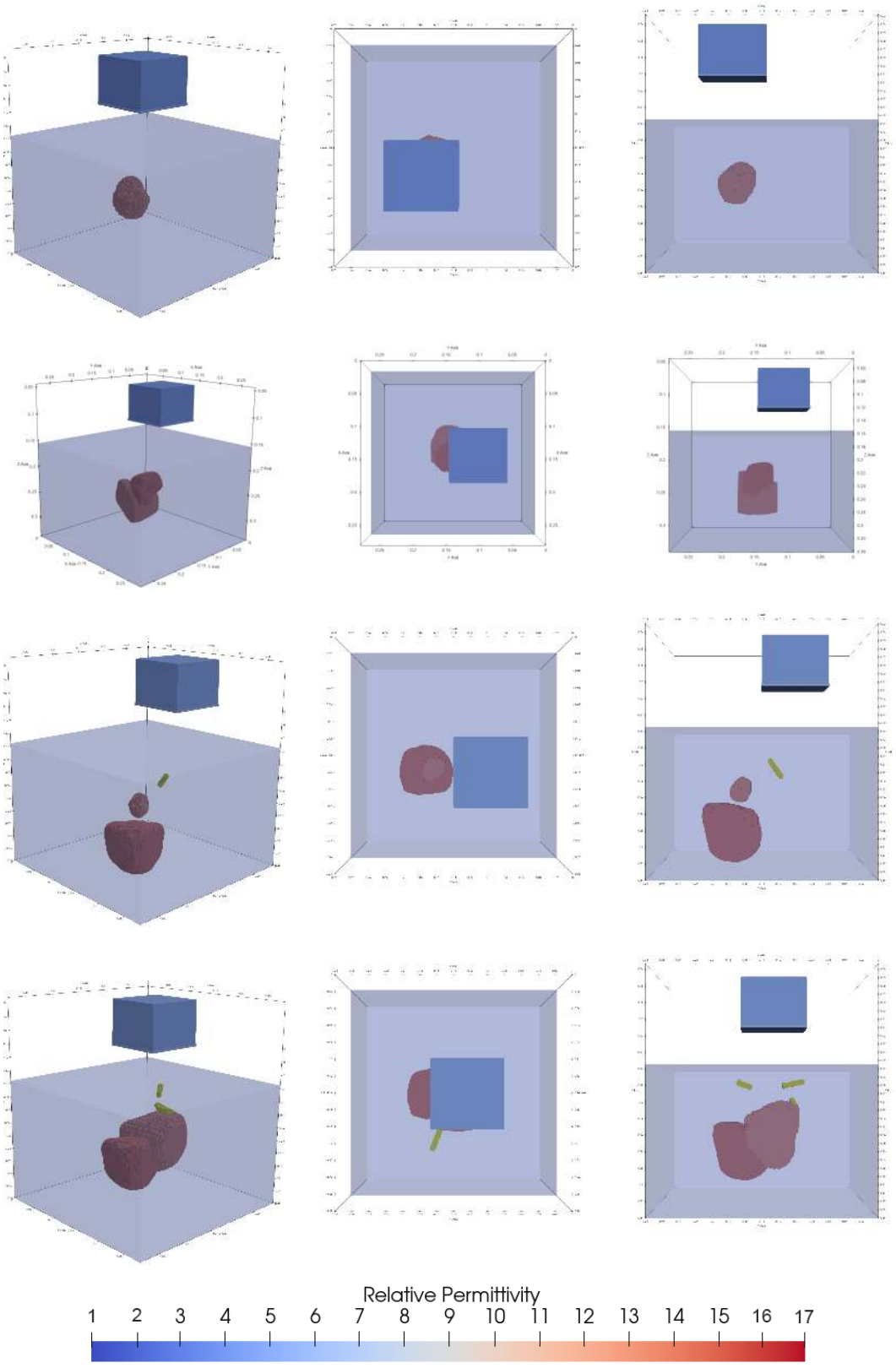


Figure 4.3: Four different false-alarm training examples containing varying amounts of rocks and bullet casings. Each row shows a different training example and each column shows three different orientations. Column 1 shows isometric views, column 2 shows top-down views, and column 3 shows side-on views.

5

REAL DATA COLLECTION

The following chapter deals with the methods used for collecting and processing the real data used in this research.

5.1 COLLECTING REAL DATA

Objective 6 of this research specified the validation of the trained ML model with real data obtained in a physical environment using the physical equivalents of the antenna, targets, false alarms, and soil media. The School of Geoscience: Infrastructure Sensing Laboratory is equipped with a sand-box which has sand of known permittivity and conductivity values. Replicas of a PMN AP mine and spent 7.62 x 51 casings are also available. Most importantly however, the lab is equipped with the GSSI 2 GHz antenna as shown in Figure 5.1, whose numerical model was used for creating the simulated dataset.

Figure 5.1: The GSSI 2GHz 'palm' antenna setup. This includes the antenna itself and the SIR system.

5.1.1 *Measurement Taking*

A comprehensive dataset comprising 5 distinct configurations was developed, with each configuration subjected to five scans, resulting in a total of 25 scans. Each A-scan was recorded with a total of 512 time samples over a time window of 6 ns. This extensive dataset is designed to facilitate the validation of models trained both of single and multiple-scan examples. The training set simulates an air-coupled antenna system, a configuration typically employed in handheld landmine detection scenarios [79]. Table 5.1 delineates the principal parameters characterising each scan. The landmine and false-alarm depth was varying, while the antenna was randomly positioned above the targets, similarly to the synthetic cases.

(a) (b)

Figure 5.2: (a) A top-down view of all the targets and false-alarm objects for which real data was collected. (b) A side-on view of the same targets.

To mitigate the potential interference from wall reflections, the measurement area was strategically positioned within the sandbox. This placement ensured that reflections from the sandbox walls would not be captured within the 6ns

measurement window. Moreover, to standardise the height of the air-coupled antenna configurations and thus maintain a semi-consistent measurement environment, a foam block was utilised as a spacer between the antenna and the soil medium as the foam has similar dielectric properties to air. Before predictions can take place on the real dataset, the real data is processed similarly to the synthetic data using the procedures outlined in Section 6.1.2. The noise contained in the real data was reduced using discrete wavelet transform method [80, 81].

Example	Scan index	Target depth	False-alarm
1	0-4	45	N/A
2	5-9	45	3 bullet
3	10-14	85	2 Rocks
4	15-19	N/A	3 bullets
5	20-24	N/A	N/A

Table 5.1: Summary of recorded data of A-scans with example indices, scan indices, antenna configurations, targets, and false-alarm target types.

5.2 COMPARING SYNTHETIC AND REAL DATA

Figure 5.3 depicts A-scan traces from synthetic and real data under identical scenarios; the synthetic setup is detailed in Figure 4.1. Materials are modeled as homogeneous and of infinite extent, contrasting with the finite-domain real sand box. Despite noise removal in real data as described in Section 6.1.2, additional noise sources are apparent that are absent in synthetic traces. Figure 5.3 highlights discrepancies between the synthetic and real GPR traces, despite the clearer GPR trace of the PMN. This highlights the requirement for a model which doesn't overfit and can generalise well to unseen data.

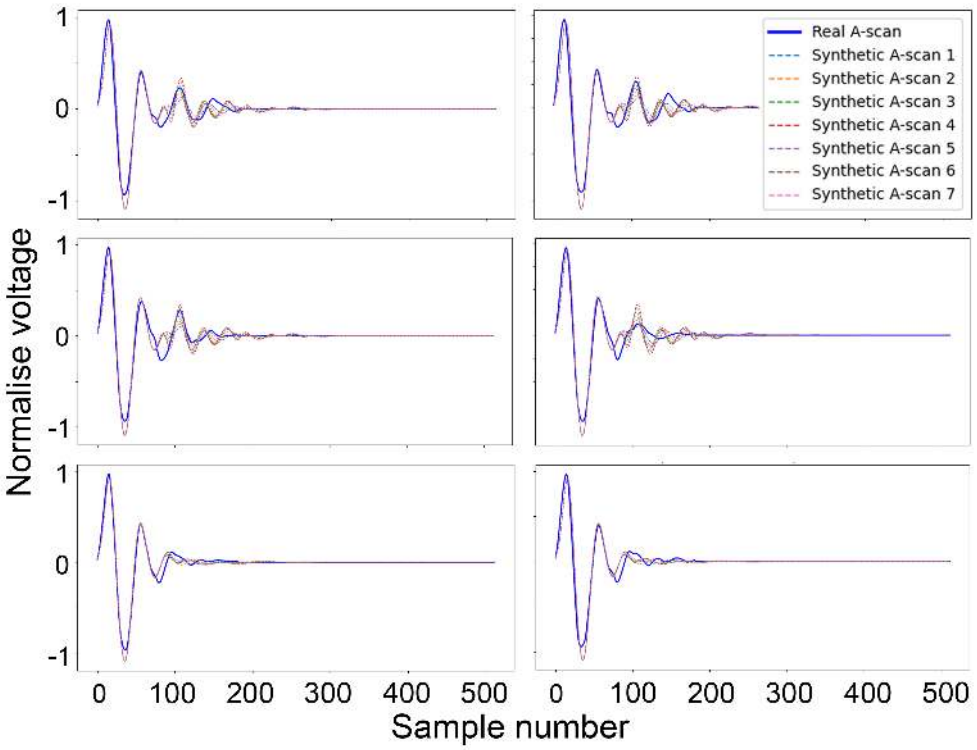


Figure 5.3: Six figures showing A-scan traces collected from the sandbox for real data scans compared against seven synthetic traces which are positioned in an evenly distributed line across the measurement domain. The first row represents the PMN landmine, the second row shows mixed examples of the PMN landmine and bullet casings, and the final row shows the data with three scattered bullet casings.

6

MACHINE LEARNING

This section describes the use of ML, specifically ANNs, for landmine detection via binary classification of GPR A-scan traces into targets and false alarms. Before deploying the ANN, it is crucial to train it using a comprehensive dataset to ensure robust model performance in real-world applications. In the context of GPR for landmine detection, critical variables such as the antenna's three-dimensional position, the target's depth, and soil properties must be precisely varied to mirror real-life conditions.This research evaluates two distinct training sets. The first set is designed to test the model's ability to generalise from simpler scenarios to real data. This involves straightforward examples where the key variables are controlled but not extensively varied. The second set comprises more complex examples, which include a broader range of conditions and parameter combinations. This set aims to enhance the model's performance on synthetic datasets that more closely mimic challenging real-world environments.

6.1 DESIGNING DATASETS

The effectiveness of ML model training significantly hinges on the design of the training set [46]. A well-constructed training set should be sufficiently comprehensive to prevent a sparse feature space [82] and must balance between targets and false-alarms to avoid overfitting and enhance convergence rates [83]. Ideally, a 'complete' training set encompasses all scenarios likely to be encountered in practice, supporting the development of a universally applicable detection model. Employing a realistic numerical modelling approach can expediently generate such comprehensive datasets [5].

6.1.1 *Feature Selection*

The choice of feature vectors significantly impacts the efficacy of an ANN. Extensive studies, such as those on B-scans and C-scans for landmine detection, demonstrate the critical role of feature selection. Klesk et al. [84] reported enhanced detection using C-scans combined with boosted decision trees. Furthermore, integrating additional features like GPS coordinates of the detection site has proven beneficial [85]. A diverse array of ML methods, including decision trees [84] and logistic regression, has been deployed in landmine detection.

Núñez-Nieto et al. [86] found that neural networks coupled with 2GHz antennas achieved the highest accuracy of 92%, although the training process on real GPR B-scan traces is labor-intensive. Lameri et al. [24] successfully trained a CNN on synthetic B-scans, attaining a 95% accuracy rate when the dataset included 20% real traces, which dropped to 83% with solely synthetic data. This research will focus on using GPR A-scans as the basis for neural network training. The design of the modelling section allows for the construction of either single or 5-scan training examples. Scans are collected from random positions within a measurement domain as to accurate simulate the location-independence of handheld measurement systems. This prevents the use of other features in the training examples besides the A-scan trace such as in [85].

6.1.2 *Data Processing*

The training sets for this research were subjected to a mix of clutter suppression and data processing techniques, including time-0 correction, normalisation, PCA, background removal, and resampling. It was observed that removing the background responses using the 6th scan from each training set did not improve the accuracy or generalisation capabilities of the ML models. These models efficiently discern and discount repetitive, irrelevant features such as background elements by assigning them low weights, which effectively performs background removal. Due to the nature of real data collection it was impossible to replicate this method of background removal thus introducing additional discrepancies between the data. While training sets processed with PCA achieved high accuracy on the training set, they struggled with generalisation, possibly due to the case sensitivity issues discussed in [4]. The finalised training sets utilised a combination of normalisation, resampling and time-0 correction for optimal results. The procedures of which are explained below and the implementation of which can be seen in Appendix A.3.2. Figure 6.1 shows the the A-scan traces for a target and false-alarm examples before and after processing. Arrows are present to highlight key components of the trace.

6.1.2.1 *Normalisation*

Normalisation is the first step to take and is essential in the pre-processing of any data for an ML scheme. More importantly, it takes place for both synthetic and real data with different normalisation parameters such that a direct comparison can be made. For both synthetic and real data this takes the form of dividing the signal by the free space maximum signal amplitude, which in the case of synthetic data is 8.08 V and in the case of real data is 365655 V.

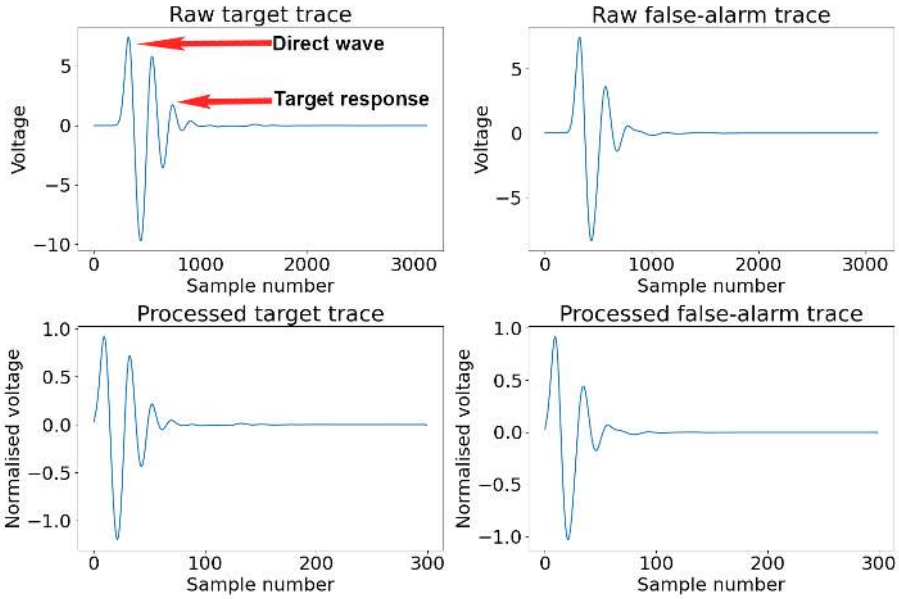


Figure 6.1: The effect of processing on the A-scan traces. The first row shows the raw traces. The second row shows the processed A-scans.

6.1.3 *Time-zero Correction*

Time-0 correction is implemented for the purpose of aligning the arrival of the direct wave reflection with the antenna for each training example. This essentially shifts the A-scans to the correct starting point and allows for accurate depth calculations. As shown in Appendix A.3.2, this functions by detecting the first part of the signal that exceeds 0.05 normalised voltage and shifting this sampling point to the new zero position.

6.1.3.1 *Resampling*

The A-scans produced by the model consist of 3117 samples per scan, and for the real data these consist of 512 samples per scan. Before they can be applied to ML they must have the same amount of samples. Preferably the amount of samples is kept at a minimum as this drastically reduced training time for the models, which is one of the primary advantages of downsampling and PCA. The minimum sample count can be calculated using the Nyquist frequency which represent the minimum samples based on the bandwidth of the antenna and the pulse duration. Using a frequency bandwidth of 2GHz leads to a minimum sampling of 36 samples per scan. However due to the higher losses associated with the higher frequencies. This is significantly less that the minimum requirement set forth by the real A-scan traces which is 256 samples. These considerations lead to the selection of 300 samples per trace for this research.

6.2 NEURAL NETWORK TRAINING

6.2.1 *First Training Set*

In an attempt to prevent over fitting, the model training was performed in two-stages as outlined in Figure 6.2. Initially, a grid-search was carried out using AutoKeras [87]. AutoKeras is a type of neural architecture search (NAS), which has been shown to effectively tune and propose deep neural networks [87]. The code to execute the grid search can be seen in Appendix A.3.3. The grid search was arranged to optimise the neural network architecture and various hyperparameters such as the optimiser, learning rate, and dropout rate.

The grid search training was carried out on a set which consists of $N = 3800$ training examples. Each training example is represented as a pair $(\mathbf{x}^{(i)}, y^{(i)})$, where $i = 1, 2, \ldots, N$. The input feature vector $\mathbf{x}^{(i)}$ for each training example i consists of 300 input features representing the resampled A-scan. Hence, $\mathbf{x}^{(i)} \in \mathbb{R}^{300}$, indicating that each example contains 300 input features. The output $y^{(i)}$ for each training example is a binary value, representing the label associated with the input $\mathbf{x}^{(i)}$. This binary output can either be 0 or 1, making it suitable for this binary classification task. Therefore, $y^{(i)} \in \{0, 1\}$. In this case and in the case of all training sets in this research, 1 will refer to a training example containing a landmine, either isolated or in a mixture with false-alarms. The value 0 represents false alarms of any kind, rocks or bullet casings. In summary, the training set $\mathcal{D}_1$ can be described as:

$$\mathcal{D}_1 = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(3615)}, y^{(3615)})\}$$

with $\mathbf{x}^{(i)} \in \mathbb{R}^{300}$ and $y^{(i)} \in \{0, 1\}$.

The dataset contained both target and false-alarm examples from a selected dataset containing targets and false alarm examples in conditions highly similar to those found in the sandbox. At a 50-50 distribution of 3165 targets and 3615 false-alarms. The dataset was split, where 80% was used for training and 10% was used for validation and 10% for testing.

6.2.2 *Neural Network Structure*

The grid search involved 600 trials with a cap of 750 epochs per trial, incorporating early stopping that terminated trials without improvement after 15 epochs. These trials were executed on the University of Edinburgh's Haggis server using an NVIDIA GeForce GTX 1080 Ti GPU, totaling 25 hours of computing time. Figure 6.3 depicts the neural network architecture, beginning with an input layer, followed by two pre-processing layers: a multi-category encoding layer and a

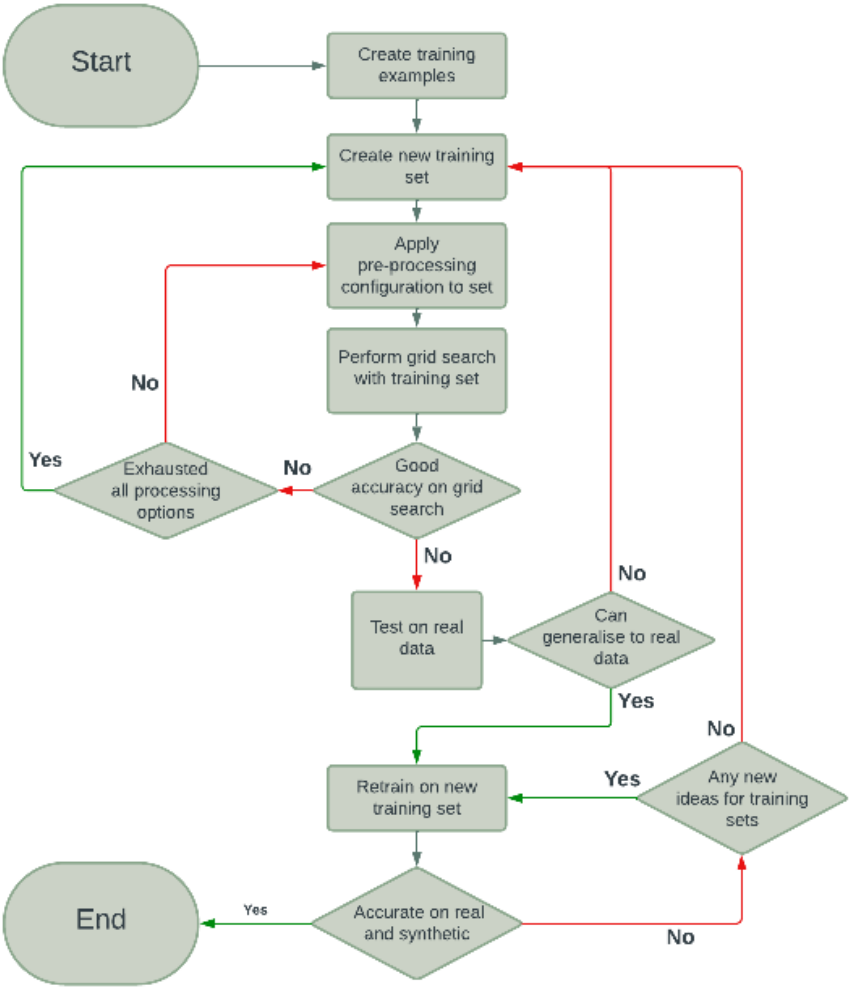


Figure 6.2: The process flowchart describing the workflow in finding, training and tuning an ML model.

normalisation layer. The multi-category encoding layer, standard in AutoKeras models, remains inactive for non-categorical data. Normalisation adjusts the data to a mean of 0 and standard deviation of 1, promoting faster convergence. The core network features two 128-neuron fully connected layers with ReLU activation, a dropout layer to mitigate overfitting by randomly setting neuron weights to zero during training, and a dense output layer with classification head functioning similarly to a sigmoid activation for classification. The final optimal hyperparameters found through grid search for the model were: optimiser:Adam, learning rate:0.001, and dropout rate:0 respectively.

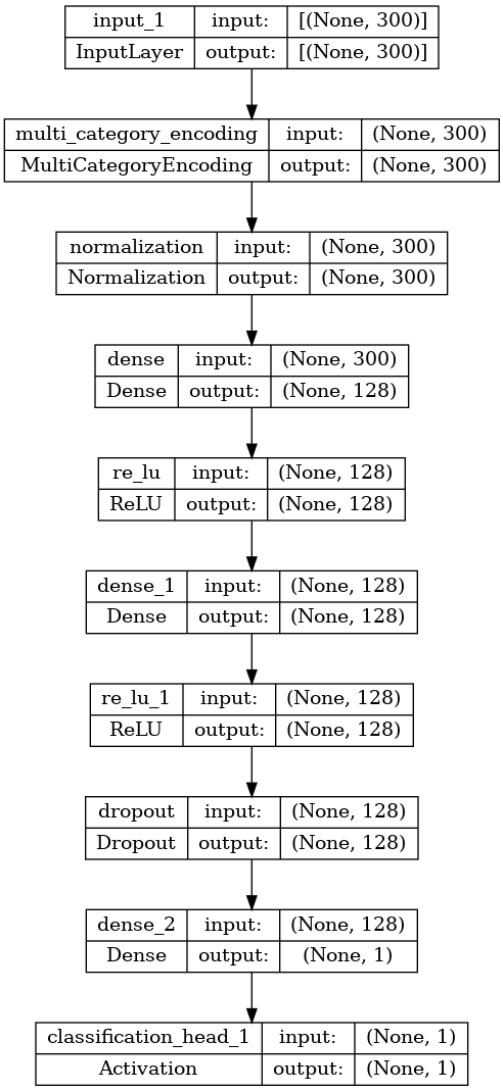


Figure 6.3: The architecture of the neural network. It consists of 2 processing layers in the MultiCategoryEncoding and Normalisation layers. And two fully connected dense 128 neuron layers which are followed by a dropout layer. It is sandwiched between a 300 layer input layer and a single neuron dense output layer with a classification head.

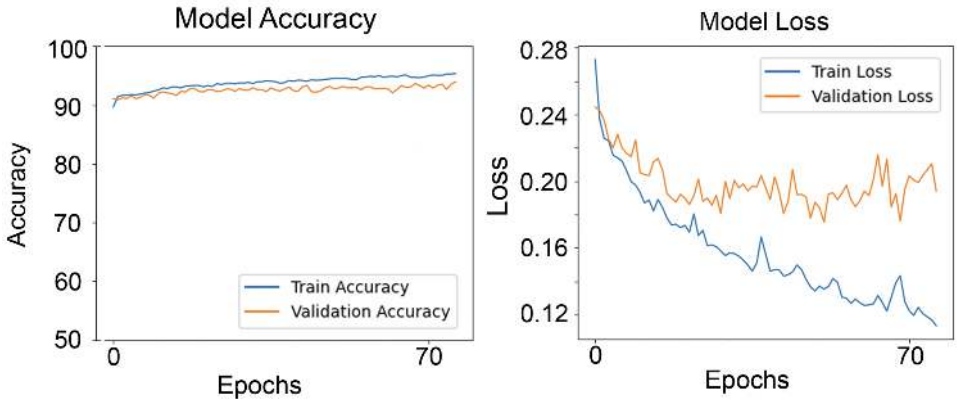
6.2.3 *Second Training Set*

Following this procedure the model was loaded and its pre-trained weights erased. It was then trained on a more diverse and complete dataset, according to the parameters specified in Table A.1, in order to maximise its ability to generalise. The dataset contained 2480 target, 1230 mixed and 1250 false-alarm examples. The dataset was split into 80% training 10% validation and a 10% testing set.

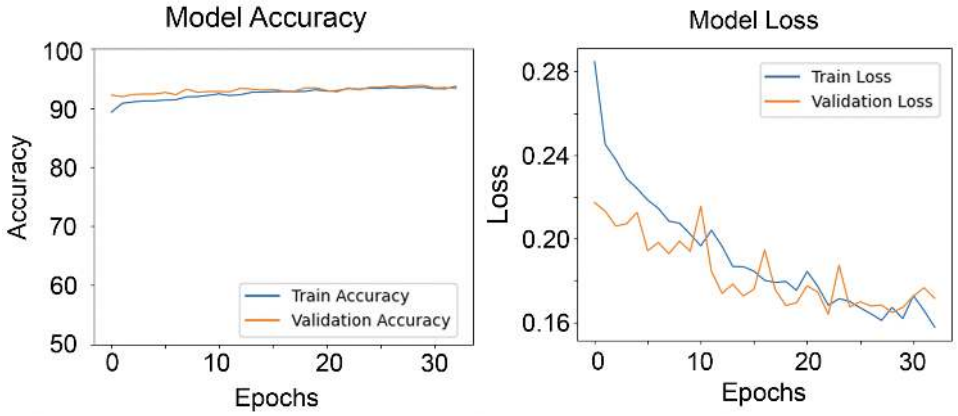
The second training $\mathcal{D}_2$ set can be described as:

$$\mathcal{D}_1 = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(4960)}, y^{(4960)})\}$$

with $\mathbf{x}^{(i)} \in \mathbb{R}^{300}$ and $y^{(i)} \in \{0, 1\}$.



(a) Patience = 25



(b) Patience = 10

Figure 6.4: (a) The retraining of the NN with a patience of 25 results in an overfitting model (b) retraining of the same model with a patience of 10 results in a high-accuracy low-loss combination.

Figure 6.4 shows the accuracy and loss curves for the retraining of the NN, which was initially configured for a different dataset. It shows promising results upon retraining with the new, more complex dataset. Notably, the training accuracy curve exhibits a steady upward trend, suggesting the model's growing proficiency in correctly classifying the training data. Meanwhile, the validation accuracy, despite its fluctuations, also improves over time, a positive indicator of the model's ability to generalise to unseen data. These fluctuations could reflect the complexities inherent in the new dataset. Furthermore, the close proximity between the training and validation accuracy suggests that the model is not overfitting, maintaining a good balance between learning from the training data and generalising to the validation data. In terms of loss, both the training and

validation loss metrics decrease as the epochs progress, with the validation loss mirroring the accuracy curve's fluctuations. This reduction in loss aligns with the improved accuracy, indicating the model's predictions are becoming increasingly reliable. The absence of a widening gap between the training and validation loss underscores the model's consistent performance across both datasets, without succumbing to overfitting. This is particularly noteworthy given the increased complexity of the new dataset, implying that the AutoKeras-derived architecture is robust enough to handle the intricacies of varied data. The final ML model was saved an implemented into a Graphical User Interface (GUI) to be used for real data testing and model validation. Figure A.1 shows a screenshot of the GUI and a snippet of its code.

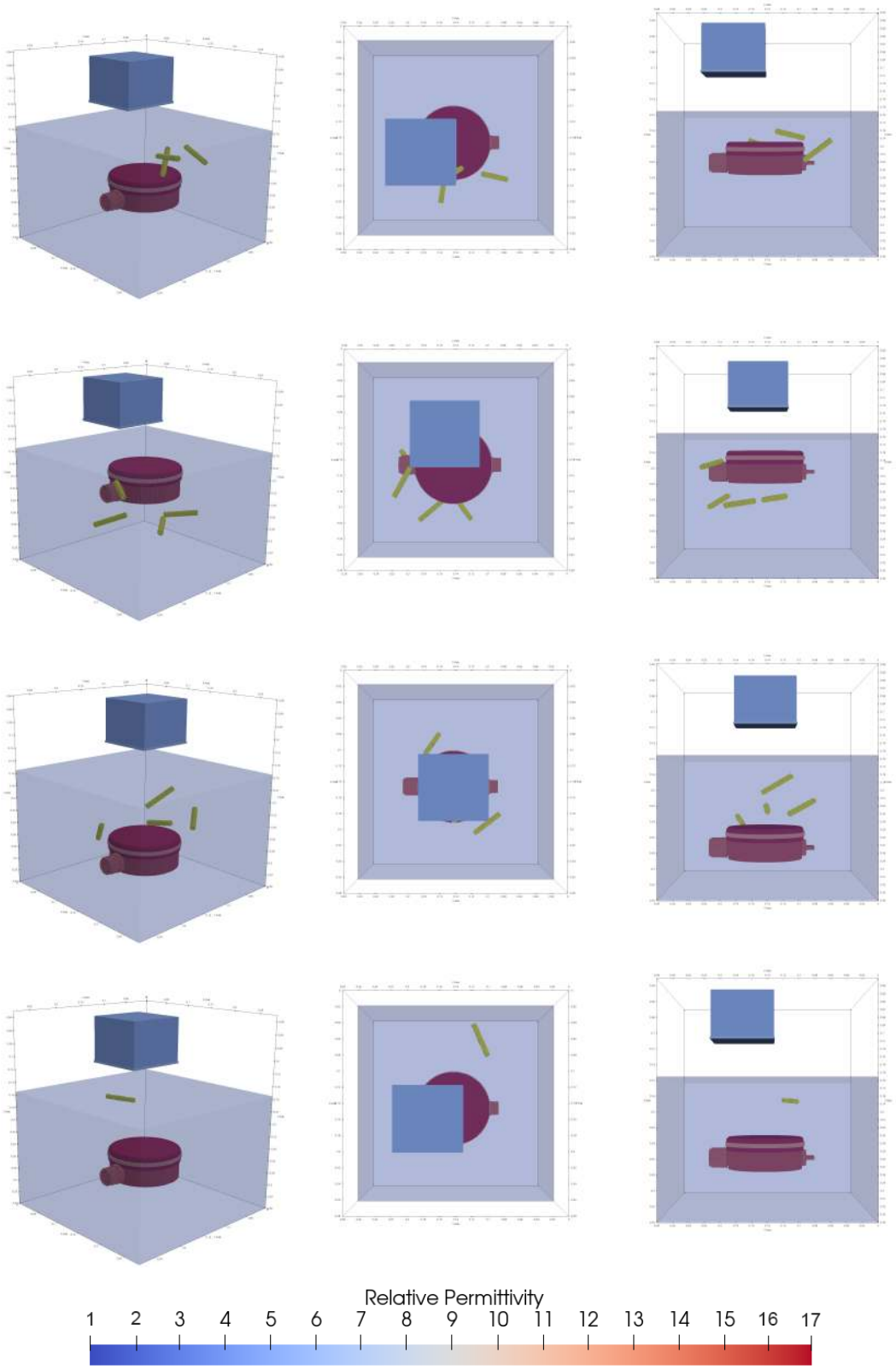


Figure 6.5: Four different mixed training examples containing both numerical PMN targets at various depths and bullet casings scattered between them. Each row shows a different training example and each column shows three different orientations. Column 1 shows isometric views, column 2 shows top-down views, and column 3 shows side-on views.

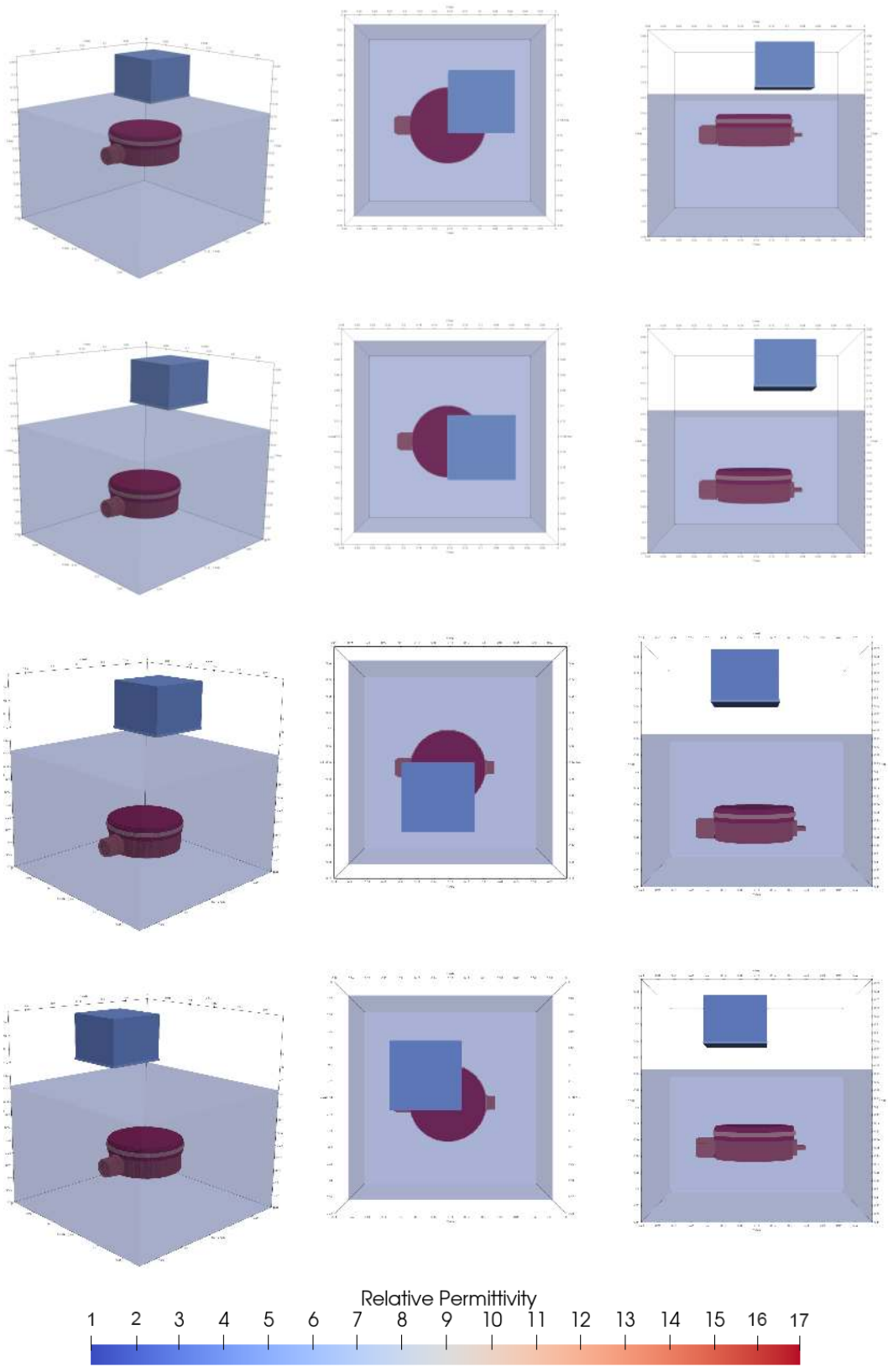


Figure 6.6: Four different target training examples containing numerical PMN targets at various depths. Each row shows a different training example and each column shows three different orientations. Column 1 shows isometric views, column 2 shows top-down views, and column 3 shows side-on views.

7

RESULTS

The following chapter covers the results of this research and is split into two different sections. The first section focuses on the results related to the performance of the NN on the synthetic dataset and second on the results covering the performance on the real data.

7.1 SYNTHETIC DATA

The model achieved an accuracy on its training set of 93.76% and 90.12% on its test set. For the test set this can be further broken down into a recall which measures the proportion of correctly predicted positives of 93.70% and a precision (correctly predicted negatives) of 84.85% leading to an F1 score of 89.11% which represents the two combined. The equations for which are given below. This aligns with the evaluation of the confusion matrix in Figure 7.1, where it is evident that the model performs roughly equally for both targets and false-alarms.

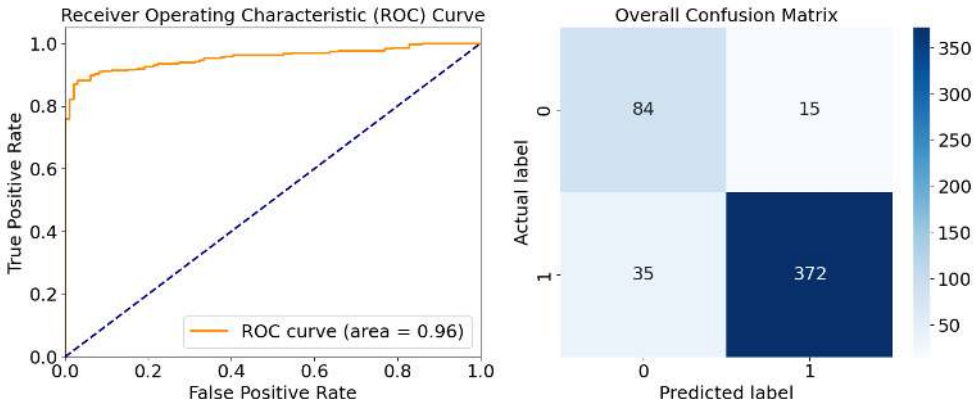


Figure 7.1: The overall ROC curve and confusion matrix for the test set. The training set consisted of about 25% false-alarms. The test set consists of about 20% false-alarms.

Figure 7.2 shows the accuracy of the network on each data type. Mixed examples were classified most effectively by this network, this however is likely due to the small size of the mixed set and the relatively low variability within the set itself. Figure 7.3 shows the ROC curves for different sizes of the same training set. It shows that for the current training set adding more examples within the same range of parameters is unlikely to increase model performance.

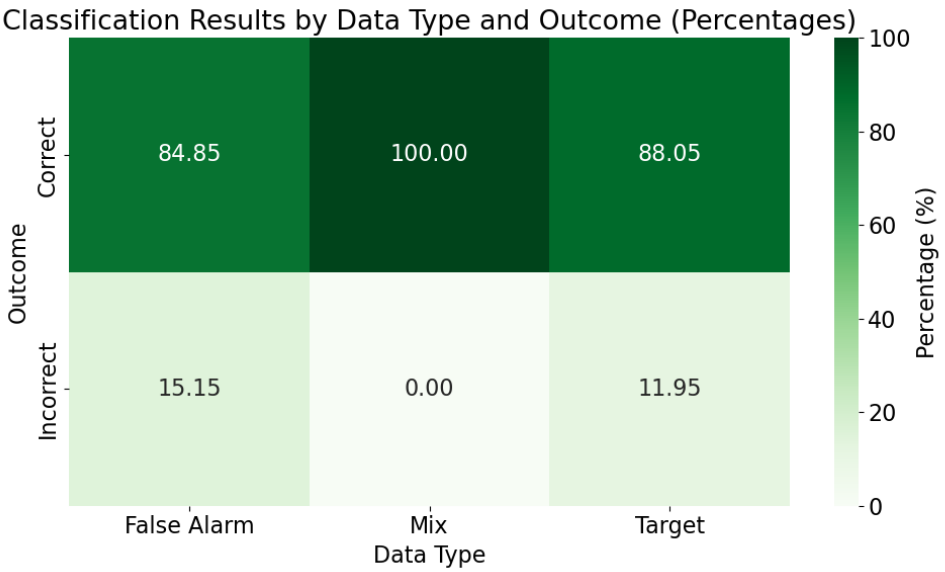


Figure 7.2: The classification accuracy by data type. The table shows the correct vs incorrect predictions for each data type. Targets and false-alarms perform roughly evenly, while the model appears to perform well on mixed examples.

The model has been trained to evaluate each A-scan separately, however the A-scans were generated in groups of 5. When the model assesses each A-scan from the group individually and then takes the weighted average of all the predictions within the group, the accuracy of the model increases substantially to reach 98.99%. This can be further broken down into a recall of 100% and a precision of 94.74% leading to an F1 score of 97.30%. This aligns with the evaluation of the confusion matrix in Figure 7.5. For the weighted average predictions for both the real and synthetic data, this was calculated by considering the weighted average of the binary predictions rather than that of the confidence values. This was done in order to minimise the impact of 'random' predictions which do not recognise the signal and assign either a 0 or 1 confidence value.

7.2 DISCUSSION

7.2.1 *Trials*

In total during this research, over 60 different training sets were developed, 20 grid searches performed and 40 different architectures tested. Combinations of the following preprocessing steps were trialed: time-0 correction, normalisation, background removal, PCA, and resampling. These preprocessing steps were trialed for both 5-scan training sets and single scan training sets. Generalisation

was achieved with 5-scan training sets, however with lower accuracies on both synthetic and real examples of around 75%.

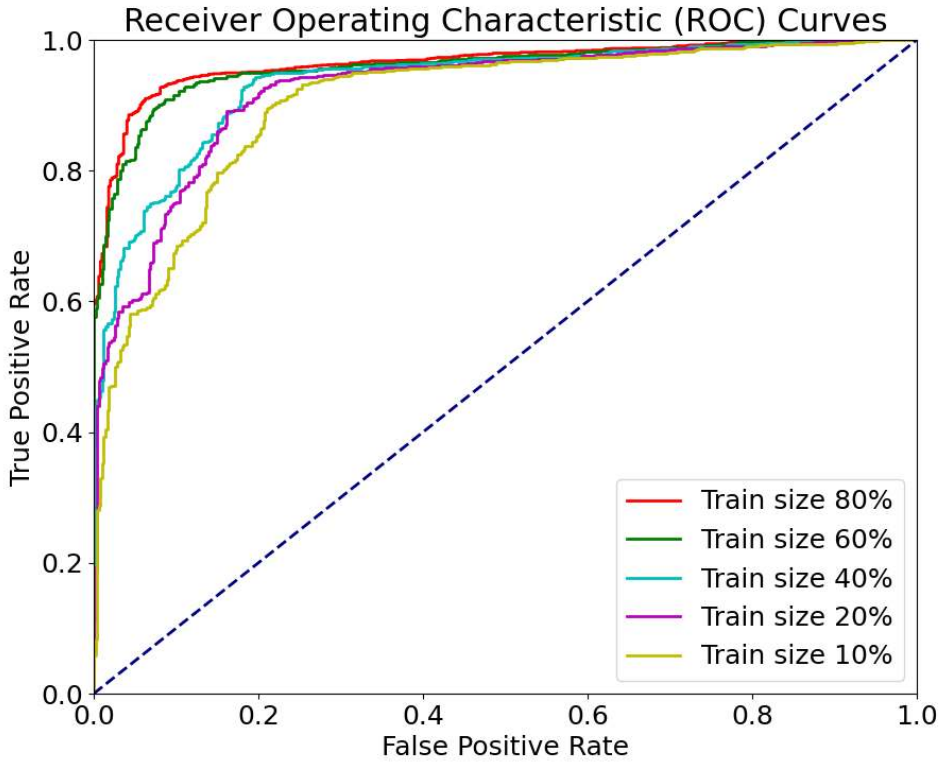


Figure 7.3: The effect of decreasing training set size on the accuracy of the model. Also shown is the convergence of the training set size, with 60% and 80% proving near equally effective.

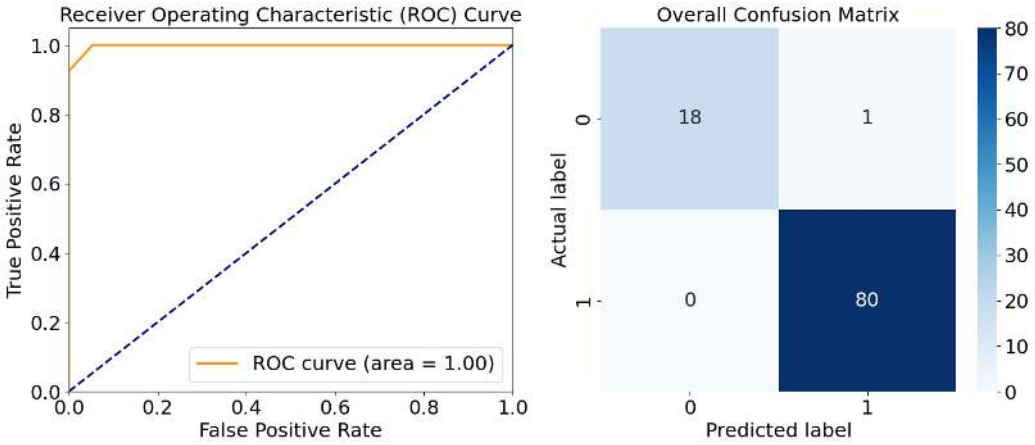


Figure 7.4: The overall ROC curve and confusion matrix for the test set. The training set consisted of about 25% false-alarms. The test set consists of about 20% false-alarms.

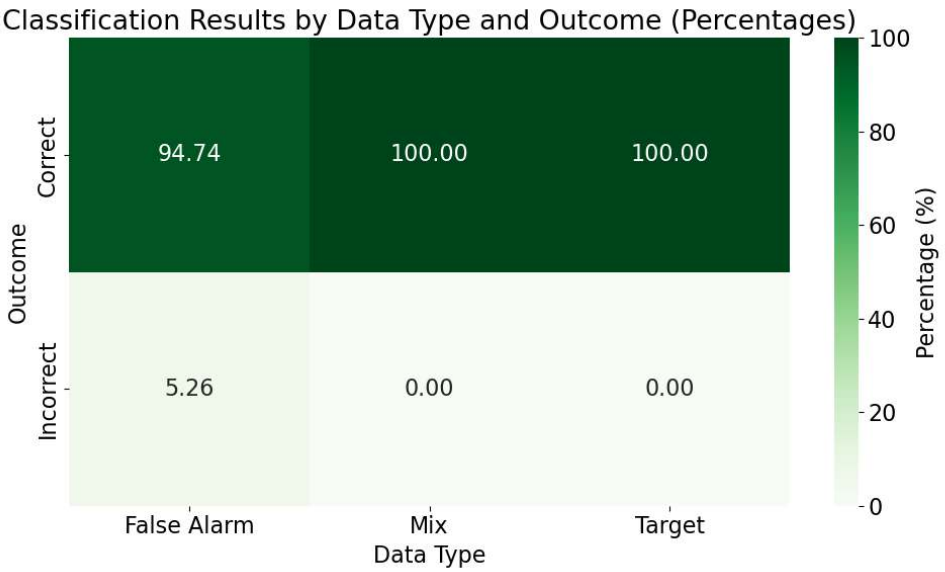


Figure 7.5: The classification accuracy by data type. The table shows the correct vs incorrect predictions for each data type. The model performs flawlessly on targets and mixed examples and only slightly worse on false-alarms.

7.2.2 *Final Configuration*

During the development and training with a great variety of datasets, it was noticed that a model trained on more complete and equally distributed training sets could achieve up to 99.7% accuracy on single scan test sets if a grid search was run with that specific training set. These models often had more complex architectures with greater amounts of neurons in their hidden layers. Although mostly preserved the structure shown in Figure 6.3, these models were unable to generalise to the real data, which presents a sign of overfitting to the synthetic data. But crucially does show that there is sufficient information within single A-scans for accurate classification. It was then hypothesised that a grid search performed on a simpler dataset would yield a less complex architecture that would be able to generalise better, albeit at the cost of slightly lower accuracy. Models found using this method were then retrained on different datasets before a model was found that could consistently generalise to the real data. Training sets which had equal distributions tended to classify each real scan as a false-alarm with high confidence. Analysis of the false-alarm targets showed great similarity between false-alarm scans within a single arrangement. It was hypothesised that training a neural network on repetitive or very similar examples would cause overfitting. And thus by removing repetitive false-alarm targets generalisation could be achieved albeit resulting in a skewed dataset.

7.3 REAL DATA

The prediction results from the model are shown in Table 7.2. Predictions were carried out on all air-coupled antenna arrangements. The model achieved an overall accuracy on a per A-scan basis of 80%. However, if the weighted average of all predictions was combined, all examples were correctly predicted and in this case, the results improve to 100% accuracy as can be seen in Table 7.1.

Example	**Indices**	**Prediction**	**Confidence**	**Class**	**Accuracy**
Target	0-4	1	0.600	1	True
Mixed Bullets	5-9	1	0.800	1	True
Mixed Rocks	10-14	1	0.800	1	True
False-alarm	15-19	0	0.000	0	True
Background	20-24	0	0.200	0	True

Table 7.1: The results for the data when a weighted average is taken. The target example contains solely a PMN mine. The mixed rocks and mixed bullets contain both a PMN mine and 3 bullets casings and 2 rocks in the setup. The False-alarm contains three bullet casings and the background contains no objects at all.

The individual plots shown in Figures A.2, A.3, A.4 were examined for reasons why the model may have made incorrect predictions. However no conclusive insights could be gained from this. Notably, most traces have reasonable confidence values, where this implies that the model is interpreting key features about the trace rather than simply guessing, as zero or one confidence would apply, especially if incorrect.

Example	Index	Prediction	Confidence	Class	Accuracy
Target	0	0	0.459	1	False
Target	1	1	0.872	1	True
Target	2	1	0.866	1	True
Target	3	1	0.671	1	True
Target	4	0	0.000	1	False
Mixed Bullets	5	1	1.000	1	True
Mixed Bullets	6	0	0.122	1	False
Mixed Bullets	7	1	1.000	1	True
Mixed Bullets	8	1	0.525	1	True
Mixed Bullets	9	1	0.691	1	True
Mixed Rocks	10	1	0.792	1	True
Mixed Rocks	11	0	0.476	1	False
Mixed Rocks	12	1	0.982	1	True
Mixed Rocks	13	1	0.959	1	True
Mixed Rocks	14	1	0.824	1	True
False-alarm	15	0	0.000	0	True
False-alarm	16	0	0.147	0	True
False-alarm	17	0	0.071	0	True
False-alarm	18	0	0.023	0	True
False-alarm	19	0	0.291	0	True
Background	20	1	0.999	0	False
Background	21	0	0.000	0	True
Background	22	0	0.000	0	True
Background	23	0	0.000	0	True
Background	24	0	0.000	0	True

Table 7.2: The predicted classifications from the ML model on the real data. The indices correspond to the indices in Table 7.1, where the setup descriptions and object compositions are given.

7.4 STATISTICAL ANALYSIS

The efficacy of the ML scheme on real data was evaluated using a binomial test to determine if the observed accuracy significantly exceeded the level expected by chance. Here are the detailed steps and hypotheses:

- **Null Hypothesis** (H_0): The accuracy of the ML scheme is 50%, i.e., the same as random guessing.
- **Alternative Hypothesis** (H_a): The accuracy of the ML scheme is greater than 50%, indicating that the ML scheme provides a predictive performance better than random guessing.

The performance was quantified by observing an accuracy of 80% (20 correct predictions out of 25 trials). The binomial test was applied under the assumption that each trial is independent and the probability of a correct prediction under the null hypothesis is 0.5. The test formula used is:

$$p = \sum_{k=x}^{n} \binom{n}{k} p^k (1-p)^{n-k} \tag{7.1}$$

where:

- $n = 25$ represents the total number of trials (predictions),
- $x = 20$ represents the number of successful predictions,
- $p = 0.5$ is the probability of success under the null hypothesis.

The computed p-value was 0.002. This p-value represents the probability of observing as many or more successful predictions under the null hypothesis of chance alone. Given that the p-value (0.002) is less than the significance level ($\alpha = 0.05$), we reject the null hypothesis. This result suggests that the performance of the ML classifier is statistically significantly better than what would be expected by chance, supporting the efficacy of the classifier with a confidence level of 95%.

8

CONCLUSIONS

The main aim of this thesis was to investigate the ability of an ML scheme trained solely on synthetic data to make accurate predictions on their real-world equivalents for landmine detection. Before such a scheme could be tested, a dataset which is accurate and representative of the real scenarios had to be created first using advanced modelling techniques and utilising the latest research and development on antennas.

The following findings were concluded from this research.

- Advanced numerical modelling techniques and accurate models can be used to develop synthetic data that closely resemble real data for simple homogeneous cases.
- In simple homogeneous sand models, it is possible for ML scheme to achieve high accuracies of up to 99.73% on their test sets.
- Using a two-stage training process it is possible to achieve a model that achieves reasonable accuracy's on its test set and is able to generalise to real data with and accuracy of 90.12% for the synthetic test data and 80% for the real data.
- These accuracies can improve to 98.99% and 100% respectively if a weighted average of individual predictions within a set is considered.

These conclusions satisfy the remaining 4 objectives as presented in Chapter 1.

In summary we can therefore conclude that: *It is possible to design an ML scheme using advanced modelling techniques that is able to classify real data, whilst being trained only on synthetic data.* And that therefore, our hypothesis can be accepted.

9

LIMITATIONS AND RECOMMENDATIONS

9.1 LIMITATIONS

Despite the promising results shown in this work it is critical that it is evaluated in the context of the scope of humanitarian demining. There are some key limitations in this work that must be understood, these are highlighted below:

- The proof-of-concept nature of the research means that the model has only been trained and tested on a very specific controlled environment. And is therefore unlikely to be able to generalise to conditions beyond this environment.
- The modelling of the PMN landmine was conducted on what is known as 'surrogate' mines, whereas the synthetic model is based on a real mine. Although a small discrepancy can potentially affect the results, its effects are not known and were not studied in this research.
- The final training of the neural network was conducted using a skewed dataset, consisting of a 75-25 ratio of targets to false-alarms. This presents an imbalance in class distribution and the effects of this imbalance in the bias of the model should be studied.
- The model has only been validated with 25 real scans. Although above the level of significance, this is significantly lower than the synthetic test set.
- The permittivity, conductivity and water fraction were assigned random variables independently. A model should be included to link these variables and produce more realistic models, as these are not independent.
- The antenna model, despite being class-leading in accuracy, does not identically mimic the behaviour of the real antenna.
- Despite attempts to remove noise from the real data, it will always contain noise that cannot otherwise be removed.

9.2 RECOMMENDATIONS

Following the work in this research, there are some recommendations for further work to be conducted using the output results and the insights gained from this research:

- The results of this work should be validated by collecting additional real A-scans from the sand box and other soils.
- The extent of the ability for the script to generalise to synthetic cases which lie beyond the ranges of the training set should be tested. Models should be generated using the script in Appendix A.3.1 with properties outside the ranges specified in Table 4.1 to gain insight into its performance limitations.
- Test and extent of the ability of the script to generalise to real cases with unknown parameters. The models should be buried in wet or damp sand and the responses recorded and tested in the validation script.
- Attempts should be made to enhance the model by adding random noise to the traces to investigate if this improves its ability to generalise.
- The model should be evaluated by investigating its ability to generalise and classify different targets.
- Training the model on real data should be investigated, to examine if this affects the accuracy of the model on more complex scenarios.

The inability for the ML scheme to generalise when trained on a large varied dataset using a complex architecture may stem from the discrepancy between the PMN model and physical PMN target. Work investigating the reproduction of the synthetic traces from real traces using GANs or otherwise may present an opportunity to circumvent this shortcoming and allow detection to be applied to more complex and varied models with greater accuracy.

A

APPENDIX

A.1 TABLES

Semi-empirical model description of input variables

Variable	**Explanation**
ϵ_0	Permittivity of free space
ρ_b	Bulk density
ρ_s	Particle density
ϵ_a^s	Dielectric permittivity of the material in the absence of free water
m_b'	Moisture bulk parameter
m_u	Moisture uniformity parameter
a	Empirically determined constant
ϵ_w'	Real part of water's permittivity
ϵ'	Real part of the complex permittivity
ϵ''	Imaginary part of the complex permittivity
σ_f	Electrical conductivity
ω	Angular frequency
ϵ_s	Static dielectric constant
S	Sand fraction
C	Clay fraction
β'	Empirically determined soil type constant
β"	Empirically determined soil type constant
$\epsilon_{w,\infty}$	High-frequency limit of water's permittivity
$\epsilon_{w,s}$	Static dielectric constant of water
$\tau_{0,w}$	Relaxation time of water
j	Imaginary unit

Table A.1: Semi-empirical model description of input variables

A.2 EQUATIONS

A.2.1 *The Semi-empirical Model*

$$\epsilon'_{(1.4-18\,\mathrm{GHz})} = \epsilon_0 \left(1 + \frac{\rho_b}{\rho_s}(\epsilon_a^s - 1) + m_v^{\beta'}\epsilon_w'^a - m_u\right)^{\frac{1}{a}} \tag{A.1}$$

$$\epsilon'_{(0.3-1.3\,\mathrm{GHz})} = 1.15\epsilon'_{(1.4-18\,\mathrm{GHz})} - 0.68 \tag{A.2}$$

$$\epsilon'' = \epsilon_0 m_u^{\frac{\beta''}{a}} \left(\frac{\sigma_f}{\omega\epsilon_0}\frac{(\rho_s - \rho_b)}{\rho_s m_u} - \epsilon''_w\right) \tag{A.3}$$

$$\epsilon_s = (1.01 + 0.44\rho_s)^2 - 0.062 \tag{A.4}$$

$$\beta' = 1.2748 - 0.519S - 0.152C \tag{A.5}$$

$$\beta'' = 1.33797 - 0.603S - 0.166C \tag{A.6}$$

$$\epsilon_w = \epsilon_{w,\infty} + \frac{\epsilon_{w,s} - \epsilon_{w,\infty}}{1 + j\omega\tau_{0,w}} \tag{A.7}$$

$$\sigma_{f(1.4-18\,\mathrm{GHz})} = -1.645 + 1.939\rho_b - 2.2562S + 1.594C \tag{A.8}$$

$$\sigma_{f(0.3-1.3\,\mathrm{GHz})} = 0.0467 + 0.2204\rho_b - 0.411S + 0.6614C. \tag{A.9}$$

A.3 CODE

A.3.1 *gprMax input models*

```
# Synthetic dataset central definitions
# !!NOTE: Do not change any parameter names !!

Batch name = 'Batch name'

Domain_x = 0.280
Domain_y = 0.280
```

```
Domain_z = 0.340

Scans per training example = 5
Mean bullet count in model = 3

Training examples Target = 0
Training examples Mix = 0
Training examples False Alarm = 0

Training examples = 200
```

Code/Dataset_definitions.txt

```
#python:

# importing necessary libraries
import numpy as np
from GSSI2000 import GSSI_2
import random as rd
import math
import ast
from scipy.ndimage import zoom
import h5py
from scipy.ndimage import affine_transform
import os
import csv

# constants

ground_depth = 0.155
length = 0.0391
b_rad = 0.00381
axes = [[0, 1, 0], [1, 0, 0], [0, 0, 1]]   # Rotation axes
landmine_status = 1
actual_bullets = mean_bullet_count
antenna_y_variation = 0.05 #check domain constraints - This has
    been set to half the hitbox of the landmine
antenna_x_variation = 0.05  #check domain constraints - This has
    been set to half the hitbox of the landmine
antenna_z_variation = 0.01  #check domain constraints
landmine_depth_variation = 0.06 #check domain constraints
rock_d1 = 0
rock_d2 = 0
```

```
29
30 x_posA = Domain_x/2 - 0.0415 + (antenna_x_variation * rd.uniform
       (-1, 1))
31 y_posA = Domain_y/2 - 0.0415 + (antenna_y_variation * rd.uniform
       (-1, 1))
32
33 T = 25 # Temperature
34 S = 1.0 # Sand fraction
35 C = 0.0 # clay fraction
36 rs = 2.65 # soil density
37 rb = 1.65 # bulk density
38
39
40 # function definitions
41
42 def get_value(key, file_path):
43     with open(file_path, 'r') as file:
44         for line in file:
45             if line.startswith(key):
46                 # Splitting by '=' and stripping to remove any
       whitespace
47                 return line.split("=")[1].strip()
48     return None
49
50 def append_to_csv(file_path, data):
51     with open(file_path, 'a', newline='') as file:
52         writer = csv.writer(file)
53         writer.writerow(data)
54
55 def clear_csv_file(file_path):
56     with open(file_path, 'w') as file:
57         file.write('')
58
59 def edit_file_values_in_place(file_path, new_permittivity,
       new_conductivity):
60     # Read the original file content
61     with open(file_path, 'r') as file:
62         content = file.readlines()
63
64     # Edit the specific line with the values
65     for i, line in enumerate(content):
66         if line.startswith('#material:'):
```

```
            parts = line.split(' ')
            parts[1] = str(new_permittivity)  # Update permittivity
            parts[2] = str(new_conductivity)  # Update conductivity
            content[i] = ' '.join(parts)
            break

    # Write the modified content back to the same file
    with open(file_path, 'w') as file:
        file.writelines(content)


def getRodriguesMatrix(theta, axis):
    v_length = np.linalg.norm(axis)
    if v_length == 0:
        raise ValueError("Length of rotation axis cannot be zero.")
    if theta == 0.0:
        return np.eye(3)

    v = np.array(axis) / v_length
    W = np.array([[0, -v[2], v[1]],
                  [v[2], 0, -v[0]],
                  [-v[1], v[0], 0]])

    rot3d_mat = np.identity(3) + W * np.sin(theta) + np.dot(W, W) *
     (1.0 - np.cos(theta))
    return rot3d_mat



def transform_volume(data, thetas, axes, scale_factors):
    # Ensure the data is of type float
    data = data.astype(np.float32)

    # Get the center of the volume
    center = np.array(data.shape) // 2

    # Initialize the transformed volume as the original data
    transformed_volume = data

    for theta, axis in zip(thetas, axes):
        # Normalize the rotation axis
        axis = np.array(axis)
```

```
        axis = axis / np.linalg.norm(axis)

        # Get the rotation matrix
        rot_mat = getRodriguesMatrix(theta, axis)

        # Create an affine transformation matrix that includes translation to the center
        affine_mat = np.eye(4)
        affine_mat[:3, :3] = rot_mat
        affine_mat[:3, 3] = center - np.dot(rot_mat, center)

        # Apply the rotation using affine_transform with the affine matrix
        transformed_volume = affine_transform(transformed_volume, affine_mat, order=0, mode='constant', cval=transformed_volume.min())

    # Apply the scaling
    transformed_volume = zoom(transformed_volume, scale_factors, order=0)

    return transformed_volume

# Apply transformations to the volume

def semi_empirical(T, S, C, rs, rb, fw):
    ewinf = 4.9
    ews = 88.045 - 0.4147*T + 6.295*(10**(-4))*(T**2) + 1.075*(10**(-5))*(T**3)
    t0w = (1/(2*np.pi))*((1.1109*(10**(-10)))*T**3 - 3.824*(10**(-12))*T**2 + 6.938*(10**(-14))*(T**2) - 5.096*(10**(-16))*(T**3))

    es = ((1.01 + 0.44*rs**2)) - 0.062
    b1 = 1.2748 - 0.519*S - 0.152*C
    b2 = 1.33797 - 0.603*S - 0.166*C
    sigma_f = -1.645 + 1.939*rb - 2.25622*S + 1.594*C
    e_0 = (1 + (rb/rs)*((es**0.65) - 1) + (fw**b1)*(ews**0.65) - fw)**(1/0.65)

    s = (fw**b2/(0.65))*sigma_f*((rs - rb)/(rs*fw))
    td = t0w
```

```
141     eds = e_0
142     edinf = e_0 - fw**(b2/0.65)*(ews - ewinf)
143 
144 
145     print("#add_dispersion_debye: 1 {} {} {}".format(eds-edinf,td,
        'my_sand'))
146 
147     return eds, edinf, td
148 
149 def process_and_save_data(input_file_path, output_file_path, thetas
        , axes, scale_factors, x_posA, y_posA, pos_z, rock_type):
150     # Check if the output file exists and delete it if it does
151     if os.path.exists(output_file_path):
152         os.remove(output_file_path)
153 
154     # Open the input file, read the data
155     with h5py.File(input_file_path, 'r') as input_file:
156         data = input_file['data'][:]
157 
158         data_rotated = transform_volume(data, thetas, axes,
        scale_factors)
159 
160         # Save the rotated data and other attributes into a new
        HDF5 file
161         with h5py.File(output_file_path, 'w') as output_file:
162             # Create a dataset for the rotated data
163             output_file.create_dataset('data', data=data_rotated)
164 
165             # Copy all other attributes from the input file to the
        output file
166             for key in input_file.attrs.keys():
167                 output_file.attrs[key] = input_file.attrs[key]
168 
169             # Copy all other datasets from the input file to the
        output file, except for 'data'
170             for key in input_file.keys():
171                 if key != 'data':
172                     input_file.copy(key, output_file)
173 
174 
175 def reset_scan_variables():
176     global landmine_status, actual_bullets, rock_count
```

```
177     landmine_status = 0
178     actual_bullets = 0
179     rock_count = 0
180     print("6th scan: Empty model. Resetting landmine_status,
        rock_count, and actual_bullets to 0")
181
182
183 def configure_landmine_and_sand_properties(ground_depth,
        landmine_depth_variation, mean_bullet_count, antenna_z_variation
        ):
184     global landmine_position, sand_rel_permittivity,
        sand_rel_conductivity, actual_bullets, landmine_status, z_posA,
        fw
185     landmine_position = (ground_depth + 0.045) + (
        landmine_depth_variation * rd.uniform(0, 1))
186     sand_rel_permittivity = rd.uniform(2.5, 25)
187     sand_rel_conductivity = rd.uniform(0.001, 1)
188     actual_bullets = mean_bullet_count  # Assuming you want to
        reset this at the start of each set
189     landmine_status = 1  # Ensure landmine is included for the next
         5 scans
190     z_posA = 0.037 + (antenna_z_variation * rd.uniform(-1, 1))
191     fw = rd.uniform(0.0001,0.12 # water fraction
192
193     # Assuming you might want to use these values outside, you can
        return them
194
195 def setup_rock_variables():
196     global rock_count, rock1_x_posA, rock1_y_posA, rock_1_pos_z,
        rock2_x_posA, rock2_y_posA, rock_2_pos_z
197     global rock_1_type, rock_2_type, rock_1_permittivity,
        rock_1_conductivity, rock_2_permittivity, rock_2_conductivity
198     global rock_1_thetas, rock_1_scale_factors, rock_2_thetas,
        rock_2_scale_factors
199     global input_file_path_1, output_file_path_1, input_file_path_2
        , output_file_path_2, file1_path, file2_path
200
201     # rock_count = int(rd.uniform(1, 2))
202     rock_count = 2
203
204     rock1_x_posA = Domain_x/2 - 0.0215 + (antenna_x_variation * rd.
        uniform(-1.1, 1.1))
```

```
205     rock1_y_posA = Domain_y/2 - 0.0215 + (antenna_y_variation * rd.
        uniform(-1.1,1.1))
206     rock_1_pos_z = ground_depth + rd.uniform(0.03, 0.1)
207
208     rock2_x_posA = Domain_x/2 - 0.0215 + (antenna_x_variation * rd.
        uniform(-1.1, 1.1))
209     rock2_y_posA = Domain_y/2 - 0.0215 + (antenna_y_variation * rd.
        uniform(-1.1, 1.1))
210     rock_2_pos_z = ground_depth + rd.uniform(0.03, 0.1)
211
212     rock_1_type, rock_2_type = int(rd.uniform(1, 3)), int(rd.
        uniform(1, 3))
213     rock_1_permittivity, rock_2_permittivity = rd.uniform(5, 10),
        rd.uniform(5, 10)
214     rock_1_conductivity, rock_2_conductivity = rd.uniform
        (0.0000001, 1), rd.uniform(0.0000001, 1)
215
216     rock_1_thetas, rock_2_thetas = [rd.uniform(0, np.pi) for _ in
        range(3)], [rd.uniform(0, np.pi) for _ in range(3)]
217     rock_1_scale, rock_2_scale = rd.uniform(0.5, 1.3), rd.uniform
        (0.5, 1.3)
218     rock_1_scale_factors, rock_2_scale_factors = [rock_1_scale] *
        3, [rock_2_scale] * 3
219
220     input_file_path_1, output_file_path_1 = f'Rocks/rock{
        rock_1_type}.h5', f'Rocks/rock_ro_{rock_1_type}.h5'
221     input_file_path_2, output_file_path_2 = f'Rocks/rock{
        rock_2_type}.h5', f'Rocks/rock_ro_{rock_2_type}.h5'
222     file1_path, file2_path = f'Rocks/rock{rock_1_type}_new_material
        .txt', f'Rocks/rock{rock_2_type}_new_material.txt'
223
224     edit_file_values_in_place(file1_path, rock_1_permittivity,
        rock_1_conductivity)
225     edit_file_values_in_place(file2_path, rock_2_permittivity,
        rock_2_conductivity)
226
227 def calculate_casing_positions(Domain_x, Domain_y, ground_depth,
        bullet_depth, length, b_rad):
228     pitch_angle = rd.uniform(-45, 45)  # casing orientation
229     yaw_angle = rd.uniform(-180, 180)  # casing orientation
230     bpos_x = Domain_x / 2 + rd.uniform(-1, 1) * ((Domain_x / 2) -
        0.09)  # casing position
```

```
    bpos_y = Domain_y / 2 + rd.uniform(-1, 1) * ((Domain_x / 2) - 0.09)  # casing position
    bpos_z = (ground_depth + bullet_depth) + (rd.uniform(-1, 1) * 0.025)  # casing position

    pitch = math.radians(pitch_angle)
    yaw = math.radians(yaw_angle)

    # Calculate the direction vector based on pitch and yaw angles
    direction_x = math.cos(yaw) * math.cos(pitch)
    direction_y = math.sin(yaw) * math.cos(pitch)
    direction_z = math.sin(pitch)

    # Calculate the end position of the cylinder
    bpos2_x = bpos_x + length * direction_x
    bpos2_y = bpos_y + length * direction_y
    bpos2_z = bpos_z + length * direction_z

    bpos3_x = bpos_x + 0.002 * direction_x
    bpos3_y = bpos_y + 0.002 * direction_y
    bpos3_z = bpos_z + 0.002 * direction_z

    print("#cylinder: {} {} {} {} {} {} {} pec y".format(bpos_x, bpos_y, bpos_z, bpos2_x, bpos2_y, bpos2_z, b_rad))
    print("#cylinder: {} {} {} {} {} {} {} free_space y".format(bpos3_x, bpos3_y, bpos3_z, bpos2_x, bpos2_y, bpos2_z, b_rad - 0.0015))


Dataset_definitions_file_path = "Dataset definitions3.txt"
# Convert these values to floats as they represent numerical data
Training_examples_TGT = int(get_value("Training examples Target", Dataset_definitions_file_path))
Training_examples_MIX = int(get_value("Training examples Mix", Dataset_definitions_file_path))
Training_examples_FA = int(get_value("Training examples False Alarm", Dataset_definitions_file_path))
scans_per_example = int(get_value("Scans per training example", Dataset_definitions_file_path))
mean_bullet_count = int(get_value("Mean bullet count in model", Dataset_definitions_file_path))
```

```
Domain_x = float(get_value("Domain_x",
    Dataset_definitions_file_path))
Domain_y = float(get_value("Domain_y",
    Dataset_definitions_file_path))
Domain_z = float(get_value("Domain_z",
    Dataset_definitions_file_path))
Batch_name = get_value("Batch name", Dataset_definitions_file_path)
# Extracting key parameters and assigning variables
Training_scans_TGT = int(scans_per_example * Training_examples_TGT)
Training_scans_MIX = int(scans_per_example * Training_examples_MIX)
Training_scans_FA = int(scans_per_example * Training_examples_FA)
total_scans = int(Training_scans_TGT + Training_scans_MIX +
    Training_scans_FA)

csv_file_path = f"Current_code/Outputs/Model_data_SAND_MIX_{
    Batch_name}_.csv"


# gprMax setup parameters

print("#domain: {} {} {}".format(Domain_x, Domain_y, Domain_z))
print("#dx_dy_dz: 0.001 0.001 0.001")
print("#time_window: 6e-9")

# Main logic


if (current_model_run - 1) % (scans_per_example + 1) ==
    scans_per_example:

    reset_scan_variables()

else:
    if (current_model_run - 1) % (scans_per_example + 1) == 0:

        setup_rock_variables()
        # Reset or initialize landmine and bullet properties
        configure_landmine_and_sand_properties(ground_depth,
    landmine_depth_variation, mean_bullet_count, antenna_z_variation
    )

# Creating the primary medium for the model
```

```
296
297 print("#material: {} {} 1 0 my_sand".format(sand_rel_permittivity,
        sand_rel_conductivity))
298 semi_empirical(T, S, C, rs, rb, fw)
299 print("#box: 0 0 {} {} {} {} my_sand".format(ground_depth, Domain_x
        , Domain_y, Domain_z))
300
301
302 # inserting bullet cases
303
304 if current_model_run <= Training_scans_TGT:
305     actual_bullets = 0
306
307 if Training_scans_TGT < current_model_run <= (Training_scans_TGT +
        Training_scans_MIX):
308     for i in range(actual_bullets):
309         if landmine_position >= (ground_depth + 0.045):
310
311             bullet_depth = 0.025
312
313             calculate_casing_positions(Domain_x, Domain_y,
        ground_depth, bullet_depth, length, b_rad)
314
315     # Log the geometry objects read
316
317
318 if (Training_scans_TGT + Training_scans_MIX) < current_model_run:
319     if rock_count >= 1:
320         process_and_save_data(input_file_path_2, output_file_path_2
        , rock_2_thetas, axes, rock_2_scale_factors,
321                               rock1_x_posA, rock1_y_posA,
        rock_1_pos_z, rock_1_type)
322         print("#geometry_objects_read: {} {} {} Rocks/rock_ro_{}.h5
         Rocks/rock{}_new_material.txt ".format(rock1_x_posA,
        rock1_y_posA, rock_1_pos_z, rock_1_type, rock_1_type))
323         rock_d1 = rock_1_pos_z
324
325         if rock_count == 2:
326             process_and_save_data(input_file_path_2,
        output_file_path_2, rock_2_thetas, axes, rock_2_scale_factors,
327                                   rock2_x_posA, rock2_y_posA,
        rock_2_pos_z, rock_2_type)
```

```
328             print("#geometry_objects_read: {} {} {} Rocks/rock_ro_
    {}.h5 Rocks/rock{}_new_material.txt ".format(rock2_x_posA,
    rock2_y_posA, rock_2_pos_z, rock_2_type, rock_2_type))
329             rock_d2 = rock_2_pos_z
330             print("rock two being printed")
331             print(rock_d2)
332
333
334     for i in range(actual_bullets):
335
336         bullet_depth = 0.045
337         calculate_casing_positions(Domain_x, Domain_y, ground_depth
    , bullet_depth, length, b_rad)
338 # Set landmine target
339
340 if current_model_run <= (Training_scans_TGT + Training_scans_MIX):
341
342     if landmine_status == 1:
343
344         print("#geometry_objects_read: {} {} {} PMN.h5
    PMN_materials.txt".format(((Domain_x/2) -0.05), ((Domain_y/2)
    -0.07), landmine_position))
345
346     else:
347         landmine_position = 0
348 else:
349
350     landmine_position = 0
351
352 # Load antenna model
353
354 GSSI_2(x_posA, y_posA, z_posA)
355
356 # print("#geometry_view: 0 0 0 {} {} {} 0.001 0.001 0.001
    TRAIN_SAND_ALL_ROCK_ n".format(Domain_x, Domain_y, Domain_z))
357
358 # Adding key parameters to a csv file for post analysis
359
360 if (current_model_run - 1) == 0:
361     clear_csv_file(csv_file_path)
362     append_to_csv(csv_file_path, ["current_model_run", "
    sand_rel_permittivity", "sand_rel_conductivity", "x_posA", "
```

```
    y_posA", "z_posA", "landmine_position", "actual_bullets","
    rock_d1","rock_d2"])
363     new_data = [current_model_run, sand_rel_permittivity,
    sand_rel_conductivity, x_posA, y_posA, z_posA, landmine_position
    , actual_bullets, rock_d1, rock_d2]
364     append_to_csv(csv_file_path, new_data)
365
366 else:
367     new_data = [current_model_run, sand_rel_permittivity,
    sand_rel_conductivity, x_posA, y_posA, z_posA, landmine_position
    , actual_bullets, rock_d1, rock_d2]
368     append_to_csv(csv_file_path, new_data)
369
370
371 #end_python:
```

Code/Model_Generate.txt

A.3.2 *Processing code*

```
1  # Synthetic dataset central definitions
2  # !!NOTE: Do not change any parameter names !!
3
4
5  Batch name =  'Batch_name'
6  Model_key =  18
7  runs =  4000
8  type = FA
9
10 Domain_x = 0.280
11 Domain_y = 0.280
12 Domain_z = 0.310
13
14 Scans per training example = 5
15 Mean bullet count in model = 3
16
17 Training examples Target = 1406
18 Training examples Mix = 665
19 Training examples False Alarm = 1400
20
21 Visualise A-scans = [0,1,2,3,4,2008,2009,2010,2011,2012]
```

```
22
23 PCA_variance = 0.999
24 PCA_scaling = 1.5
```

Code/Processing definitions.txt

```
1  import numpy as np
2  import pandas as pd
3  from sklearn.decomposition import PCA
4  import matplotlib.pyplot as plt
5  import ast
6  from scipy.signal import resample
7
8  # This version primarily focussed on creating sets with unprocessed
       A-scans
9  # It does this for singular A-scans
10
11 noSamples = 300
12
13 # Extracting key parameters and assigning variables
       -----------------
14 Dataset_definitions_file_path = "processing_dataset_definitions.txt
       "
15
16 def get_value(key, file_path):
17     with open(file_path, 'r') as file:
18         for line in file:
19             if line.startswith(key):
20                 # Splitting by '=' and stripping to remove any
       whitespace
21                 return line.split("=")[1].strip()
22     return None
23
24 # Convert these values to floats as they represent numerical data
25 Training_examples_TGT = int(get_value("Training examples Target",
       Dataset_definitions_file_path))
26 Training_examples_MIX = int(get_value("Training examples Mix",
       Dataset_definitions_file_path))
27 Training_examples_FA = int(get_value("Training examples False Alarm
       ", Dataset_definitions_file_path))
28 scans_per_example = int(get_value("Scans per training example",
       Dataset_definitions_file_path))
```

```
PCA_factor = float(get_value("PCA_scaling",
    Dataset_definitions_file_path))
PCA_variance = float(get_value("PCA_variance",
    Dataset_definitions_file_path))
Batch_name = get_value("Batch name", Dataset_definitions_file_path)

# Calculating key parameters
Training_scans_TGT = scans_per_example * Training_examples_TGT
Training_scans_MIX = scans_per_example * Training_examples_MIX
Training_scans_FA = scans_per_example * Training_examples_FA
total_scans = Training_scans_TGT + Training_scans_MIX +
    Training_scans_FA



# path = f'Current_code/Intermediate_datasets/NEW_TGT_FA.csv'

path = 'Current_code/Intermediate_datasets/SIMPLES_500S_5_13_V7.csv
    '
#path = 'Current_code/Intermediate_datasets/NBR_TGT_MIX_FA.csv's
data = pd.read_csv(path , header=None)

input_array = data.values

# Normalize input A-scans
maxVal = 8.08228  # Max amplitude in free space
input_array_normalized = input_array / maxVal
print("input array shape",input_array_normalized.shape)

# Apply time zero correction

corrected_rows = []  # To store the corrected rows
min_length = np.inf  # Initialize min_length to infinity

for row in input_array_normalized:
    # Step 2: Find the first index where the value passes 0.05
    time_zero_index = np.argmax(row > 0.05)  # np.argmax returns
    the first True (value exceeds 0.05)

    # Step 3: Remove all values before this index
    corrected_row = row[time_zero_index:]

```

```
66      # Append the corrected row to our list
67      corrected_rows.append(corrected_row)
68
69      # Update min_length if the current row is shorter
70      if len(corrected_row) < min_length:
71          min_length = len(corrected_row)
72
73  # Step 4: Truncate all rows to the length of the shortest row
74  truncated_rows = np.array([row[:min_length] for row in
        corrected_rows])
75  print("truncated array shape",truncated_rows.shape)
76  # np.savetxt(f'Geom/Repeats/First_Batch/Data_processing/
        Time_0_A_scans_{Batch_name}.csv', truncated_rows, delimiter=',')
77
78  def remove_background_and_reduce(a_scans):
79      """
80      Remove the background by subtracting the 6th row from the first
         5 rows in each group of 6.
81      Removes the 6th row from the dataset after subtraction.
82
83      Parameters:
84      a_scans (numpy.ndarray): The input array of A-scans.
85
86      Returns:
87      numpy.ndarray: The array after background removal and reduction
        .
88      """
89      # Ensure the array is a numpy array
90      a_scans = np.array(a_scans)
91
92      # Calculate the number of groups of 6
93      num_groups = a_scans.shape[0] // (scans_per_example+1)
94
95      # Initialize an empty list to hold the modified A-scans
96      modified_a_scans = []
97
98      # Process each group
99      for i in range(num_groups):
100         start_idx = i * (scans_per_example+1)
101         end_idx = start_idx + scans_per_example
102         background = a_scans[start_idx + scans_per_example]
103
```

```
        # Subtract the background from the first 5 rows and add them to the modified list
        for j in range(start_idx, end_idx):
            # modified_a_scan = a_scans[j] - background
            modified_a_scan = a_scans[j] - 0
            modified_a_scans.append(modified_a_scan)

    # Convert the list back to a numpy array
    modified_a_scans_array = np.array(modified_a_scans)
    return modified_a_scans_array

# Remove the background and reduce the array
a_scans_after_removal = remove_background_and_reduce(truncated_rows)
a_scans_after_removal2= resample(a_scans_after_removal , noSamples, axis=1)

print('resampled array shape:', a_scans_after_removal2.shape)

# Creating the classification values
ones = np.ones(((Training_scans_TGT + Training_scans_MIX), 1))
zeros = np.zeros((Training_scans_FA, 1))
classification_values = np.concatenate((ones, zeros), axis=0)

classification_values = np.concatenate((ones,  zeros), axis=0)
print('classification',classification_values .shape)

# Creating the classification values
twos = np.full((Training_scans_TGT, 1),2)
threes= np.full((Training_scans_MIX, 1), 3)  # This creates an array of twos
fours = np.full((Training_scans_FA, 1),4)

categorisation_values = np.concatenate((twos,threes,fours), axis=0)
print('categorisation_value shape:', categorisation_values.shape)

# Creating the training dataset
Training_dataset = np.hstack((a_scans_after_removal2, classification_values, categorisation_values))
units = Training_dataset.shape[1]

```

```
# Training_dataset = np.hstack((a_scans_after_removal,
    classification_values))
print("Training_dataset_shape:",Training_dataset.shape)
np.savetxt(f'ML_stuff/Training_Sets/Train_{units}_1Sc_{Batch_name}.
    csv', Training_dataset, delimiter=',')
print("Training dataset saved to:",f'ML_stuff/Training_Sets/Train_{
    units}_1Sc_{Batch_name}.csv')
```

Code/Process.txt

A.3.3 *ML code*

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.metrics import AUC
import autokeras as ak
from tensorflow.keras.models import load_model

# Define Batch Name for saving files
Batch_name = 'batch_name'

# Load synthetic dataset
path = 'Path_to_your_training_set'
data = pd.read_csv(path, header=None)
print('dataset shape', data.shape)

# Split data into features, labels, and types
X = data.iloc[:, :-2].values  # Features: All columns except the
    last two
y = data.iloc[:, -2].values   # Labels: The second last column
data_types = data.iloc[:, -1].values  # Data types: The last column

# Split synthetic data into training and temporary sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size
    =0.1, random_state=42)

# Further split the temporary set into test and auxiliary test sets
X_test, X_aux_test, y_test, y_aux_test = train_test_split(X_temp,
    y_temp, test_size=0.5, random_state=42)

```

```
# Configure the AutoKeras model search
clf = ak.StructuredDataClassifier(overwrite=True, max_trials=600)

# Train the model
clf.fit(X_train, y_train, epochs=750)

# Evaluate the model on synthetic test data
loss_synthetic, accuracy_synthetic = clf.evaluate(X_test, y_test)
print(f"Accuracy on Synthetic Test Data: {accuracy_synthetic*100:.2f}%")

# Access and summarize the best model
model = clf.export_model()
model.summary()

# Save the best model
model.save(f'ML_stuff/models/best_model_autokeras_{Batch_name}.tf', save_format='tf')
print('Model saved at:', f'ML_stuff/models/best_model_autokeras_{Batch_name}.tf')

# Saving test set information for reproducibility
np.savetxt(f'ML_stuff/Test_Sets/Synthetic_Test_Set_{Batch_name}.csv', np.hstack((X_test, y_test.reshape(-1, 1))), delimiter=',')
np.savetxt(f'ML_stuff/Test_Sets/Real_Test_Set_{Batch_name}.csv', real_data, delimiter=',')
print('Test sets saved to:')
print(f'ML_stuff/Test_Sets/Synthetic_Test_Set_{Batch_name}.csv')
print(f'ML_stuff/Test_Sets/Real_Test_Set_{Batch_name}.csv')
```

Code/Grid_Search.txt

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import EarlyStopping
import autokeras as ak
import tensorflow as tf
import random as rd


```

```
path = 'training_set_path'


data = pd.read_csv(path, header=None)
print('New dataset shape:', data.shape)


Run_ID = 'Run_ID'

RS = int(rd.uniform(0,42))

input_components = data.values
X = input_components[:, :-2]  # Features: All columns except the last two
y = input_components[:, -2]   # Labels: The second last column
data_types = input_components[:, -1]  # Data types: The last column

# Split the dataset into features, labels, and a separate holdout test set
X_train_new, X_temp, y_train_new, y_temp, data_types_train, data_types_temp = train_test_split(X, y, data_types, test_size=0.1, random_state=RS)

# Further split the temp set into final test set and an auxiliary test set
X_test_new, X_aux_test, y_test_new, y_aux_test, data_types_test, data_types_aux_test = train_test_split(X_temp, y_temp, data_types_temp, test_size=0.5, random_state=RS)

# Now we construct the auxiliary test set with features, labels, and data types
Auxiliary_test = np.hstack((X_aux_test, y_aux_test.reshape(-1, 1), data_types_aux_test.reshape(-1, 1)))
np.savetxt(f'ML_stuff/Test_Sets/Aux_Set_{Run_ID}.csv', Auxiliary_test, delimiter=',')

# Load the previously trained AutoKeras model
#model_path = 'ML_stuff/models/best_model_autokeras_MIX_1Sc_302'
model_path = 'ML_stuff/models/updated_model_autokeras_PLS_WORKS_V3'
loaded_model = load_model(model_path, custom_objects=ak.CUSTOM_OBJECTS)

```

```
# Function to reset weights of specific layers
def reset_weights(model, layer_names):
    for layer in model.layers:
        if layer.name in layer_names:
            if hasattr(layer, 'kernel_initializer') and hasattr(layer, 'bias_initializer'):
                old_weights, old_biases = (None, None)
                if hasattr(layer, 'get_weights') and len(layer.get_weights()) > 0:
                    old_weights, old_biases = layer.get_weights()
                if old_weights is not None and old_biases is not None:
                    layer.set_weights([
                        layer.kernel_initializer(shape=old_weights.shape),
                        layer.bias_initializer(shape=old_biases.shape)
                    ])
                    print(f'Reset weights for layer: {layer.name}')

# Specify the layer names to reset (based on your model's architecture)
layer_names_to_reset = ['input_1','multi_category_encoding ','normalization','dense', 'dense_1','dense_2','classification_head_1']  # Example layer names, adjust based on model.summary()
reset_weights(loaded_model, layer_names_to_reset)

# Continue training with early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=25, restore_best_weights=True)
history = loaded_model.fit(
    X_train_new, y_train_new,
    epochs=150,
    validation_split=0.2,
    callbacks=[early_stopping],
    verbose=1
)

# Save the updated model
new_model_path = f'ML_stuff/models/updated_model_autokeras_{Run_ID}'
```

```
loaded_model.save(new_model_path, save_format='tf')
print(f"model_path = '{new_model_path}'")
print(f"test_data_path = 'ML_stuff/Test_Sets/Aux_Set_{Run_ID}.csv'")

# Evaluate the model on new test data
loss, accuracy = loaded_model.evaluate(X_test_new, y_test_new)
print(f"New test accuracy: {accuracy*100:.2f}%")

# Plotting the accuracy and loss
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.savefig(f'ML_stuff/figs/accuracy_loss_retrain_{Run_ID}.png')
plt.close()
```

Code/Retrain.txt

```
import numpy as np
from tensorflow.keras.models import load_model
import pandas as pd
import autokeras as ak
from sklearn.metrics import roc_curve, auc, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

model_path = 'Model_path'
test_data_path = 'test_data_path'
```

```
11
12 Run_ID = 'Run_ID'
13
14 # Load and prepare your unseen test data
15 # This assumes the last column is the label, the second to last is
       the data type
16 test_data = pd.read_csv(test_data_path, header = None)
17 X_test = test_data.iloc[:, :-2].values  # Features
18 y_test = test_data.iloc[:, -2].values   # Labels
19 data_types = test_data.iloc[:, -1].values  # Data types
20
21 # Real data Path !
22 path = 'Real_data_path'
23 Test_data = np.load(path)
24 X = Test_data[:, :-1]  # Features: All columns except the last
25 y = Test_data[:, -1]  # Labels: The last column
26
27 # Define the list of example indices
28 #example_no = list(range(45))
29 example_no = [0,1,2,3,4,10,11,12,13,14,20,
30     21,22,23,24,40,41,42,43,44,50,51,52,53,54]
31
32 model = load_model(model_path)
33
34 # Initialize a list to store the results
35 results = []
36
37 # Iterate over the example indices and make predictions
38 for index in example_no:
39     example_data = X[index].reshape(1, -1)
40     probability = model.predict(example_data)
41     prediction = (probability > 0.5).astype(int)[0][0]
42     confidence = np.max(probability)
43
44     # Format confidence as a decimal string
45     formatted_confidence = f"{confidence:.3f}"  # Format confidence
       to 3 decimal places
46
47     results.append({
48         "Index": index,
49         "Binary Prediction": prediction,
```

```
50          "Confidence": formatted_confidence,  # Use formatted
        confidence
51          "Actual Classification": y[index]
52      })
53
54  # Convert the results list to a DataFrame for neat presentation
55  results_df = pd.DataFrame(results)
56  results_df['Prediction Correct?'] = results_df['Binary Prediction']
         == results_df['Actual Classification']
57  correct_predictions_percentage = results_df['Prediction Correct?'].
        mean() * 100
58
59  # Print the updated results DataFrame with formatted confidence
60  print(results_df.to_string(index=False))
61
62  # Print the percentage of correct predictions
63  print(f"Percentage of correct predictions: {
        correct_predictions_percentage:.2f}%")
64
65
66  # Make predictions on the test data
67  y_pred_prob = model.predict(X_test).ravel()
68
69  # Calculate the ROC curve and AUC
70  fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
71  roc_auc = auc(fpr, tpr)
72
73  # Generate predicted labels based on the probability threshold of
        0.5
74  y_pred = (y_pred_prob > 0.5).astype(int)
75
76  # Create a figure for the ROC curve and overall confusion matrix
77  fig, ax = plt.subplots(1, 2, figsize=(16, 6))
78
79  # Plot ROC curve on the first subplot
80  ax[0].plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (
        area = {roc_auc:.2f})')
81  ax[0].plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
82  ax[0].set_xlim([0.0, 1.0])
83  ax[0].set_ylim([0.0, 1.05])
84  ax[0].set_xlabel('False Positive Rate')
85  ax[0].set_ylabel('True Positive Rate')
```

```
ax[0].set_title('Receiver Operating Characteristic (ROC) Curve')
ax[0].legend(loc="lower right")

# Plot overall confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", ax=ax[1])
ax[1].set_title('Overall Confusion Matrix')
ax[1].set_ylabel('Actual label')
ax[1].set_xlabel('Predicted label')

# Adjust layout for overall plots
plt.tight_layout()

# Save the figure containing both plots
plt.savefig(f'ML_stuff/figs/overall_roc_cm_{Run_ID}.png')
plt.close()

# Define function to safely predict if data is present
def safe_predict(model, X):
    if X.size == 0:
        return np.array([])  # Return an empty array if no data is present
    else:
        return model.predict(X).ravel()

# Split the test data based on data types
X_test_target = X_test[data_types == 2]
X_test_mix = X_test[data_types == 3]
X_test_false_alarm = X_test[data_types == 4]

# Predict probabilities for each subset using the safe predict function
predictions_target = safe_predict(model, X_test_target)
predictions_mix = safe_predict(model, X_test_mix)
predictions_false_alarm = safe_predict(model, X_test_false_alarm)

# Define actual labels
actual_target = np.ones(len(predictions_target))
actual_mix = np.ones(len(predictions_mix))
actual_false_alarm = np.zeros(len(predictions_false_alarm))

```

```
125 # Convert probabilities to binary predictions, safely handle empty
        arrays
126 binary_predict = lambda p: (p > 0.5).astype(int)
127 predictions_target = binary_predict(predictions_target)
128 predictions_mix = binary_predict(predictions_mix)
129 predictions_false_alarm = binary_predict(predictions_false_alarm)
130
131 # Combine actuals and predictions into a DataFrame
132 results = pd.DataFrame({
133     'Actual': np.concatenate([actual_target, actual_mix,
        actual_false_alarm]),
134     'Predicted': np.concatenate([predictions_target,
        predictions_mix, predictions_false_alarm]),
135     'Data Type': ['Target'] * len(actual_target) + ['Mix'] * len(
        actual_mix) + ['False Alarm'] * len(actual_false_alarm)
136 })
137
138 # Add an 'Outcome' column
139 results['Outcome'] = np.where(results['Actual'] == results['
        Predicted'], 'Correct', 'Incorrect')
140
141 # Calculate the number of correct predictions
142 results['Correct'] = results['Actual'] == results['Predicted']
143
144 # Calculate overall accuracy
145 accuracy = results['Correct'].mean() * 100
146
147 # Print the accuracy
148 print(f"Accuracy: {accuracy:.2f}%")
149
150 # Pivot the DataFrame to create a 3x2 matrix (3 data types x
        Correct/Incorrect outcome)
151 matrix = pd.pivot_table(results, index='Outcome', columns='Data
        Type', aggfunc='size', fill_value=0)
152
153 # Convert counts to percentages
154 for col in matrix.columns:
155     matrix[col] = (matrix[col] / matrix[col].sum()) * 100
156
157 # Plot the matrix
158 plt.figure(figsize=(9, 6))
```

```
159 sns.heatmap(matrix, annot=True, fmt=".2f", cmap="Greens", cbar_kws
        ={'label': 'Percentage (%)'})
160 plt.title('Classification Results by Data Type and Outcome (
        Percentages)')
161 plt.ylabel('Outcome')
162 plt.xlabel('Data Type')
163 plt.tight_layout()  # Adjust layout to prevent clipping of ylabel
164 plt.show()
165
166 plt.savefig(f'ML_stuff/figs/cm_data_type_{Run_ID}.png')
167 plt.close()
168
169 print("All plots have been saved.")
```

Code/Evaluate.txt

A.4 GRAPHICAL USER INTERFACE

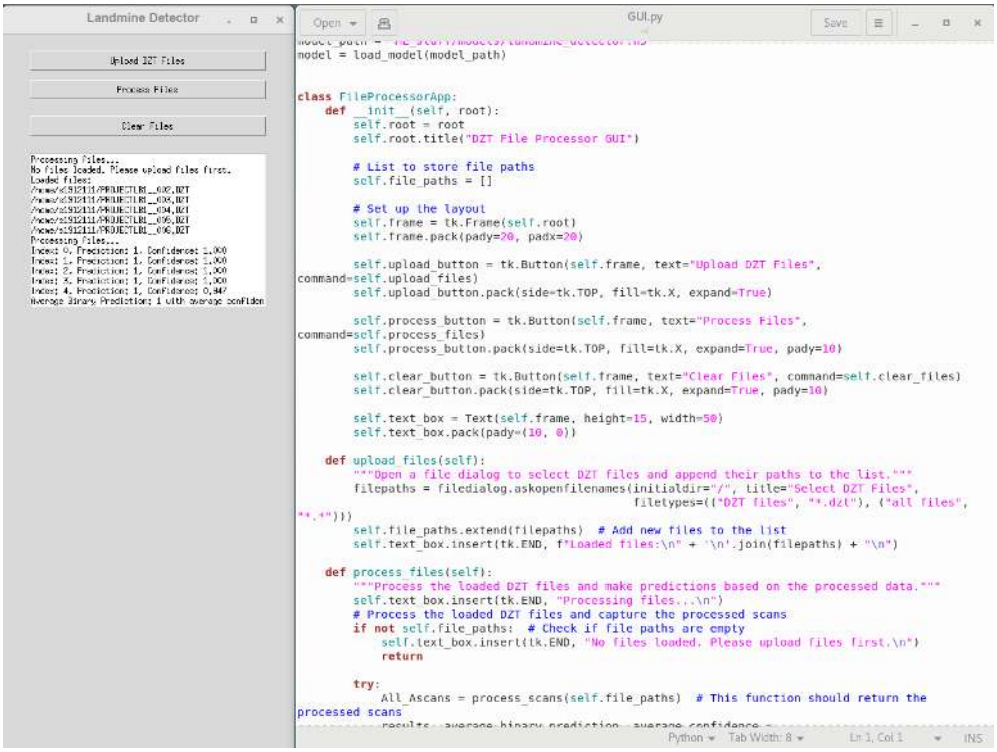


Figure A.1: Screenshot of the GUI interface (left) and the a snippet of the relavant code (right).

A.5 SUPPLEMENTARY FIGURES

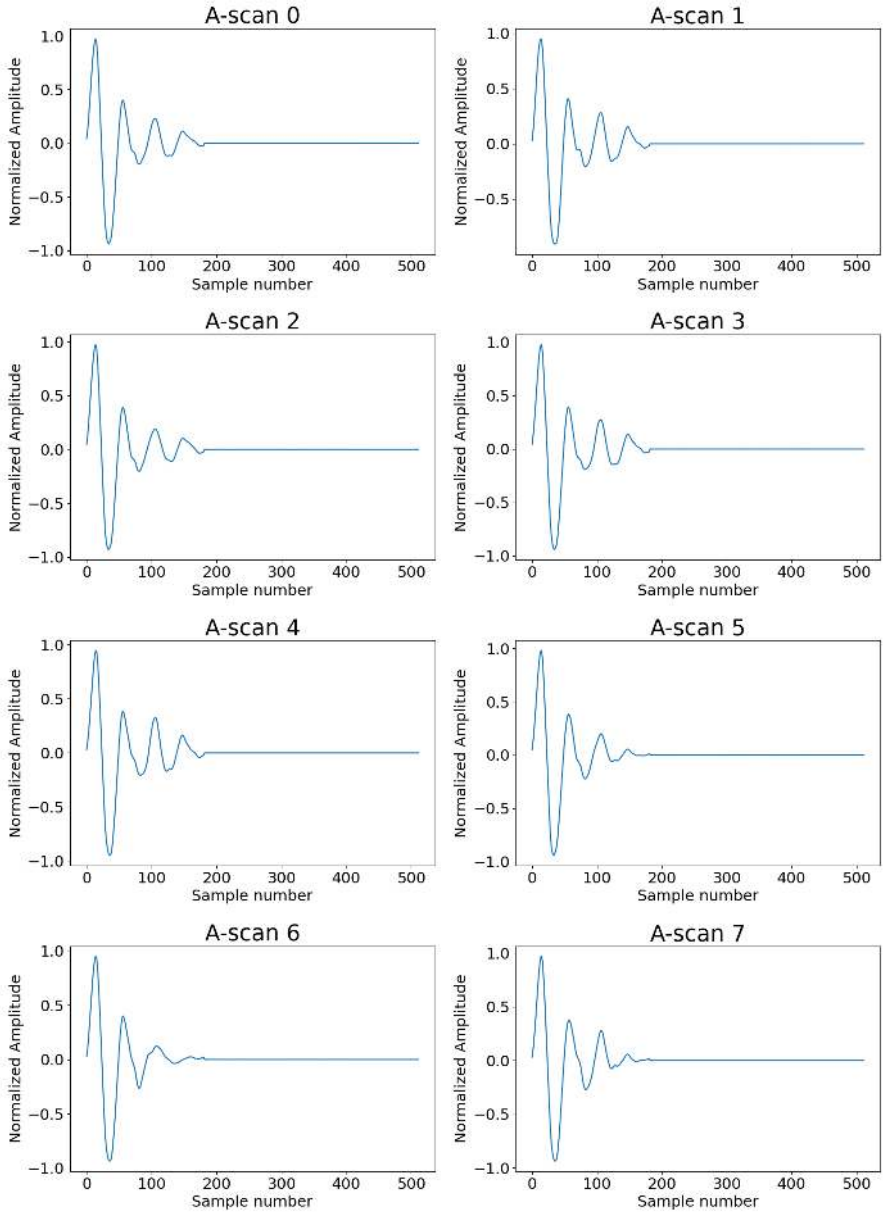


Figure A.2: The A-scan traces of all the real data examples (indices 0-7).

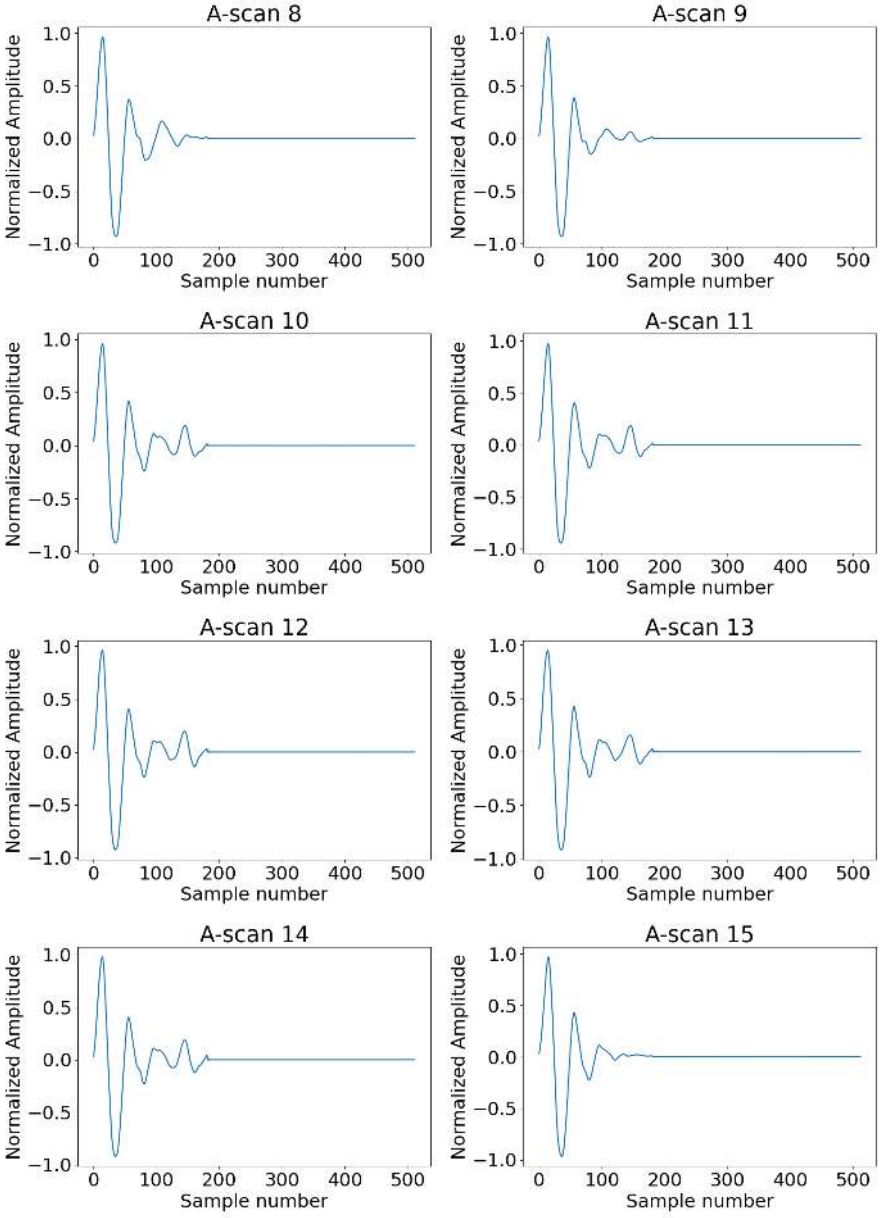


Figure A.3: The A-scan traces of all the real data examples (indices 8-15).

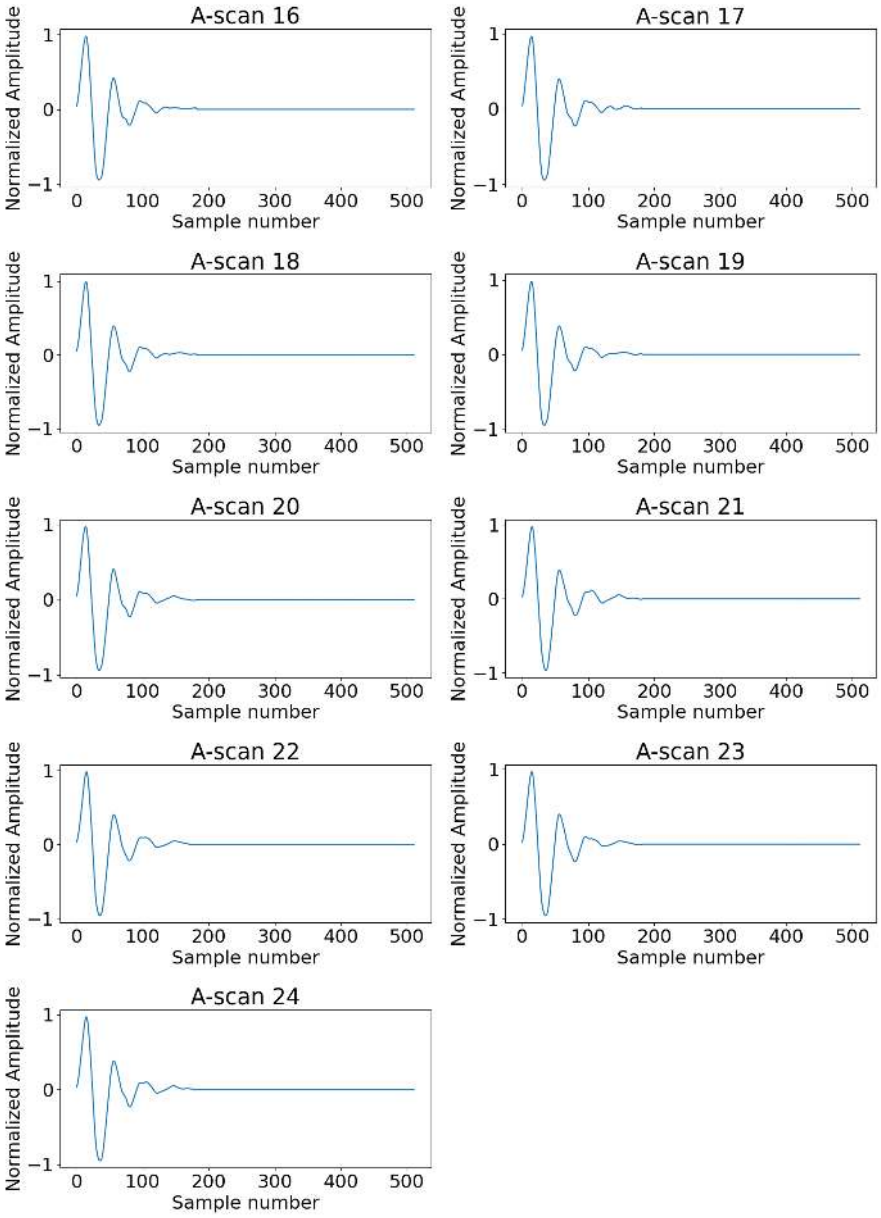


Figure A.4: The A-scan traces of all the real data examples (indices 16-24).

REFERENCES

[1] N. J. Cassidy. A review of practical numerical modelling methods for the advanced interpretation of ground-penetrating radar in near-surface environments. *Near Surface Geophysics*, 5:138–148, 2007.

[2] A. P. Annan. Ground-penetrating radar. In *Near-Surface Geophysics*, volume 13. Society of Exploration Geophysicists, 2005.

[3] O. Patsia. *Deep learning processing and interpretation of ground penetrating radar data using a numerical equivalent of a real GPR transducer.* thesis report, The University of Edinburgh, 2023.

[4] I. Giannakis, A. Giannopoulos, and C. Warren. A realistic fdtd numerical modeling framework of ground penetrating radar for landmine detection. *IEEE journal of selected topics in applied earth observations and remote sensing*, 9(1):37–51, 2015.

[5] I. Giannakis. *Realistic numerical modelling of ground penetrating radar for landmine detection.* thesis report, The University of Edinburgh, 2016.

[6] O. Patsia, A. Giannopoulos, and I. Giannakis. Background removal, velocity estimation, and reverse-time migration: A complete gpr processing pipeline based on machine learning. *IEEE Transactions on Geoscience and Remote Sensing*, 61, 2023.

[7] N. Cassidy. Introduction to gpr. In *Workshop at the 12th International Conference on Ground Penetrating Radar*, volume 14, pages 12–19, 2008.

[8] A. M. Zoubir, I. J. Chant, C. L. Brown, B. Barkat, and C. Abeynayake. Signal processing techniques for landmine detection using impulse ground penetrating radar. *IEEE sensors journal*, 2(1):41–51, 2002.

[9] D. J. Daniels. *Ground penetrating radar*, volume 1. Iet, 2004.

[10] O. Patsia, A. Giannopoulos, and I. Giannakis. Developing a realistic numerical equivalent of a gpr antenna transducer using global optimizers. *Near Surface Geophysics*, 2023.

[11] C. Warren, A. Giannopoulos, and I. Giannakis. gprmax: Open source software to simulate electromagnetic wave propagation for ground penetrating radar. *Computer Physics Communications*, 209:163–170, 2016.

[12] R. C. Benson, R. A. Glaccum, and M. R. Noel. *Geophysical techniques for sensing buried wastes and waste migration*. National Water Well Association, 1983.

[13] C. Balanis. *Advanced Engineering Electromagnetics*. Wiley, New York, 1989.

[14] J. R. Wang and T. J. Schmugge. An empirical model for the complex dielectric permittivity of soils as a function of water content. *IEEE Transactions on Geoscience and remote sensing*, (4):288–295, 1980.

[15] H. M. Jol. *Ground penetrating radar theory and applications*. Elsevier, 2008.

[16] Line Surveying. Gssi 2000 mhz palm antenna. https://line-surveying.com/ground-penetrating-radar/gssi-2000-mhz-palm-antenna, 2024. Accessed: 2024-03-29.

[17] I. Giannakis, A. Giannopoulos, C. Warren, and N. Davidson. Numerical modelling and neural networks for landmine detection using ground penetrating radar. In *2015 8th International Workshop on Advanced Ground Penetrating Radar (IWAGPR)*, pages 1–4. IEEE, 2015.

[18] A. F. Peterson, S. L. Ray, and R. Mittra. *Computational Methods for Electromagnetics*. IEEE Press, New York, 1998.

[19] J. M. Jin. *The Finite Element Method in Electromagnetics*. John Wiley and Sons, New York, 2nd edition, 2002.

[20] K. Yee. Numerical solution of initial boundary value problems in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14:302–307, 1966.

[21] J. Li, L. X. Guo, Y. C. Jiao, and R. Wang. Composite scattering of a plasma-coated target above dispersive sea surface by the ade-fdtd method. *IEEE geoscience and remote sensing letters*, 10(1):4–8, 2012.

[22] J. M. Bourgeois and G. S. Smith. A complete electromagnetic simulation of a ground penetrating radar for mine detection: theory and experiment. In *IEEE Antennas and Propagation Society International Symposium*, volume 2, 1997.

[23] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Proceedings of the Cambridge Philosophical Society*, 1947.

[24] S. Lameri, F. Lombardi, P. Bestagini, M. Lualdi, and S. Tubaro. Landmine detection from gpr data using convolutional neural networks. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 508–512. IEEE, 2017.

[25] F. Hollender and S. Tillard. Modeling ground-penetrating radar wave propagation and reflection with the jonscher parameterization. *geophysics*, 63(6):1933–1942, 1998.

[26] J. H. Bradford. Frequency-dependent attenuation analysis of ground-penetrating radar data. *Geophysics*, 72(3):J7–J16, 2007.

[27] M. R. Taherian, W. E. Kenyon, and K. A. Safinya. Measurements of dielectric response of water-saturated rocks. *Geophysics*, 55(12):1530–1541, 1990.

[28] J. Clegg and M. P. Robinson. A genetic algorithm used to fit debye functions to the dielectric properties of tissues. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[29] I. Giannakis, A. Giannopoulos, and N. Davidson. Incorporating dispersive electrical properties in fdtd gpr models using a general cole-cole dispersion function. In *2012 14th International Conference on Ground Penetrating Radar (GPR)*, pages 232–236. IEEE, 2012.

[30] F. L. Teixeira, W. C. Chew, M. Straka, M. L. Oristaglio, and T. Wang. Finite-difference time-domain simulation of ground penetrating radar on dispersive, inhomogeneous, and conductive soils. *IEEE Transactions on Geoscience and remote sensing*, 36(6):1928–1937, 1998.

[31] G. E. Atteia and K. F. A. Hussein. Realistic model of dispersive soils using plrc-fdtd with applications to gpr systems. *Progress In Electromagnetics Research B*, 26:335–359, 2010.

[32] M. C. Dobson, F. T. Ulaby, M. T. Hallikainen, and M. A. El-Rayes. Microwave dielectric behavior of wet soil—part ii: Dielectric mixing models. *IEEE Transactions on geoscience and remote sensing*, (1):35–46, 1985.

[33] N. R. Peplinski, F. T. Ulaby, and M. C. Dobson. Corrections to dielectric properties of soils in the 0.3–1.3-ghz range. *IEEE Transactions on Geoscience and Remote Sensing*, 33(6):1340, 1995.

[34] A. Giannopoulos and N. Diamanti. Numerical modelling of ground-penetrating radar response from rough subsurface interfaces. *Near Surface Geophysics*, 6(6):357–369, 2008.

[35] D. L. Turcotte. *Fractals and Chaos in Geology and Geophysics*. The Press Syndicate of the University of Cambridge, Cambridge, U.K., 1992.

[36] J. Rea and R. Knight. Geostatistical analysis of ground-penetrating radar data: A means of describing spatial variation in the subsurface. *Water Resources Research*, 34(3):329–339, 1998.

[37] J. Tabony, D. O. Carlson, H. A. Duvoisin III, and J. Torres-Rosario. Detection of bulk explosives using the gpr only portion of the hstamids system. In *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XV*, volume 7664, pages 731–736. SPIE, 2010.

[38] O. A. Pryshchenko, V. Plakhtii, O. M. Dumin, G. P. Pochanin, V. P. Ruban, L. Capineri, and F. Crawford. Implementation of an artificial intelligence approach to gpr systems for landmine detection. *Remote Sensing*, 14(17):4421, 2022.

[39] T. Counts, A. C. Gurbuz, W. R. Scott, J. H. McClellan, and K. Kim. Multistatic ground-penetrating radar experiments. *IEEE transactions on geoscience and remote sensing*, 45(8):2544–2553, 2007.

[40] C. Warren and A. Giannopoulos. Creating finite-difference time-domain models of commercial ground-penetrating radar antennas using taguchi's optimisation method. *Geophysics*, 76(2):G37–G47, 2011.

[41] D. J. Daniels. A review of gpr for landmine detection. *Sensing and imaging*, 7(3):90, 2006.

[42] O. Patsia, A. Giannopoulos, and I. Giannakis. A digital twin of the gssi 2000 mhz palm antenna developed using multi-parametric optimisation. In *2021 11th International Workshop on Advanced Ground Penetrating Radar (IWAGPR)*, pages 1–5. IEEE, 2021.

[43] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[44] W. C. Roberts. *Landmines in Cambodia*. Cambria Press, 2011.

[45] I. Rudakevych. Geographical aspects of pollution of the territory of ukraine of explosive objects. *The scientific issues of Ternopil Volodymyr Hnatiuk National Pedagogical University. series: geography*, 55:81–88, 2023.

[46] M. C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

[47] O. Patsia, A. Giannopoulos, and I. Giannakis. Gpr full-waveform inversion with deep-learning forward modeling: A case study from non-destructive testing. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–10, 2023.

[48] S. Haykin. *Neural Networks and Learning Machines*. Pearson Education, 3 edition, 2009.

[49] P. Gamba and S. Lossani. Neural detection of pipe signatures in ground penetrating radar images. *IEEE Transactions on Geoscience and Remote Sensing*, 38(2):790–797, 2000.

[50] N. P. Singh and M. J. Nene. Buried object detection and analysis of gpr images: Using neural network and curve fitting. In *2013 Annual International Conference on Emerging Research Areas and 2013 International Conference on Microelectronics, Communications and Renewable Energy*, pages 1–6. IEEE, 2013.

[51] X. Zhang, L. Han, M. Robinson, and A. Gallagher. A gans-based deep learning framework for automatic subsurface object recognition from ground penetrating radar data. *IEEE Access*, 9:39009–39018, 2021.

[52] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[53] Q. Dai, Y. H. Lee, H. Sun, J. Qian, G. Ow, M. L. M. Yusof, and A. Yucel. A deep learning-based gpr forward solver for predicting b-scans of subsurface objects. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022.

[54] S. S. Todkar, C. Le Bastard, A. Ihamouten, V. Baltazart, X. Dérobert, C. Fauchard, D. Guilbert, and F. Bosc. Detection of debondings with ground penetrating radar using a machine learning method. In *2017 9th International Workshop on Advanced Ground Penetrating Radar (IWAGPR)*, pages 1–6. IEEE, 2017.

[55] M.-T. Pham and S. Lefèvre. Buried object detection from b-scan ground penetrating radar data using faster-rcnn. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 6804–6807. IEEE, 2018.

[56] B. Cagnoli and T. J. Ulrych. Singular value decomposition and wavy reflections in ground-penetrating radar images of base surge deposits. *Journal of applied geophysics*, 48(3):175–182, 2001.

[57] Z. Tong, J. Gao, and D. Yuan. Advances of deep learning applications in ground-penetrating radar: A survey. *Construction and Building Materials*, 258:120371, 2020.

[58] R. Solimene, A. Cuccaro, A. Dell'Aversano, I. Catapano, and F. Soldovieri. Ground clutter removal in gpr surveys. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(3):792–798, 2013.

[59] M. A. González-Huici, I. Catapano, and F. Soldovieri. A comparative study of gpr reconstruction approaches for landmine detection. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(12):4869–4878, 2014.

[60] R. Persico, F. Soldovieri, and E. Utsi. Microwave tomography for processing of gpr data at ballachulish. *Journal of Geophysics and Engineering*, 7(2):164, 2010.

[61] S. E. Hamran, D. T. Gjessing, J. Hjelmstad, and E. Aarholt. Ground penetrating synthetic pulse radar: Dynamic range and modes of operation. *Journal of Applied Geophysics*, 33(1-3):7–14, 1995.

[62] E. Tebchrany, F. Sagnard, V. Baltazart, J.-P. Tarel, and X. Dérobert. Assessment of statistical-based clutter reduction techniques on ground-coupled gpr data for the detection of buried objects in soils. In *Proceedings of the 15th International Conference on Ground Penetrating Radar*, pages 604–609. IEEE, 2014.

[63] X. Song, D. Xiang, K. Zhou, and Y. Su. Improving rpca-based clutter suppression in gpr detection of antipersonnel mines. *IEEE Geoscience and Remote Sensing Letters*, 14(8):1338–1342, 2017.

[64] M P Masarik, J Burns, B T Thelen, J Kelly, and T C Havens. Gpr anomaly detection with robust principal component analysis. In *Detection and Sensing of Mines, Explosive objects, and Obscured targets XX*, volume 9454, pages 395–405. SPIE, 2015.

[65] C. Liu, C. Song, and Q. Lu. Random noise de-noising and direct wave eliminating based on svd method for ground penetrating radar signals. *Journal of Applied Geophysics*, 144:125–133, 2017.

[66] W. Bi, Y. Zhao, C. An, and S. Hu. Clutter elimination and random-noise denoising of gpr signals using an svd method based on the hankel matrix in the local frequency domain. *Sensors*, 18(10):3422, 2018.

[67] M. Garcia-Fernandez, Y. Alvarez-Lopez, A. Arboleya-Arboleya, F. Las-Heras, Y. Rodriguez-Vaqueiro, B. Gonzalez-Valdes, and A. Pino-Garcia. Svd-based clutter removal technique for gpr. In *2017 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, pages 2369–2370. IEEE, 2017.

[68] G. Borgioli, L. Capineri, P. Falorni, S. Matucci, and C. G. Windsor. The detection of buried pipes from time-of-flight radar data. *IEEE Transactions on Geoscience and Remote Sensing*, 46(8):2254–2266, 2008.

[69] G. L. Plett, T. Doi, and D. Torrieri. Mine detection using scattering parameters and an artificial neural network. *IEEE Transactions on Neural Networks*, 8(6):1456–1467, 1997.

[70] M. R. Azimi-Sadjadi, D. E. Poole, S. Sheedvash, and K. D. Sherbondy. Detection and classification of buried dielectric anomalies using a separated aperture sensor and a neural network discriminator. *IEEE transactions on instrumentation and measurement*, 41(1):137–143, 1992.

[71] C.-C. Yang and N. K. Bose. Landmine detection and classification with complex-valued hybrid neural network using scattering parameters dataset. *IEEE transactions on neural networks*, 16(3):743–753, 2005.

[72] Arms Project (Human Rights Watch) and Physicians for Human Rights (US). *Landmines: a deadly legacy*, volume 2156. Human Rights Watch, 1993.

[73] R. Dolgov. Landmines in russia and the former soviet union: A lethal epidemic. *Medicine & Global Survival*, 7(1):38–42, 2001.

[74] B. Sai, I. Morrow, and P. Van Genderen. Limits of detection of buried landmines based on local echo contrasts. In *Proc. 28th EuMc Workshop*, pages 121–125, 1998.

[75] R. Keeley. Understanding landmines and mine action. *Mines Action Canada*, 2003.

[76] Y. L. D. Zhou. *NATO Infantry Weapons Standardization: Ideal or Possibility?* PhD thesis, University of Calgary, 2016.

[77] S. Wong, A. Gatt, V. Stamatescu, and M. McDonnell. Understanding data augmentation for classification: When to warp? *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–6, 2016.

[78] C Shorten and T M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[79] S. Tantum, K. Morton, L. Collins, and P. Torrione. Investigation of the effects of operator technique on handheld sensor data for landmine detection. In *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XVII*, volume 8357, pages 389–396. SPIE, 2012.

[80] J. Baili, S. Lahouar, M. Hergli, A. Amimi, and K. Besbes. Application of the discrete wavelet transform to denoise gpr signals. In *2nd International Symposium on Communications, Control and Signal Processing, Marrakech, Morocco*, page 11, 2006.

[81] J. Baili, S. Lahouar, M. Hergli, I. L. Al-Qadi, and K. Besbes. Gpr signal denoising by discrete wavelet transform. *Ndt & E International*, 42(8):696–703, 2009.

[82] P. Melville and R. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99–111, 2005.

[83] H. P. Chan, B. Sahiner, R. F. Wagner, and N. Petrick. Classifier design for computer-aided diagnosis: effects of finite sample size on the mean performance of classical and neural network classifiers. *Medical physics*, 26(12):2654–2668, 1999.

[84] P. Klęsk, A. Godziuk, M. Kapruziak, and B. Olech. Fast analysis of c-scans from ground penetrating radar via 3-d haar-like features with application to landmine detection. *IEEE Transactions on Geoscience and Remote Sensing*, 53(7):3996–4009, 2015.

[85] A. Manandhar, K. Morton, L. Collins, and P. Torrione. Multiple instance learning framework for landmine detection using ground-penetrating radar. 8017:618–627, 2011.

[86] X. Núñez-Nieto, M. Solla, P. Gómez-Pérez, and H. Lorenzo. Gpr signal characterization for automated landmine and uxo detection based on machine learning techniques. *Remote sensing*, 6(10):9729–9748, 2014.

[87] H. Jin, Q. Song, and X. Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1946–1956, 2019.