Smart system for passive detection and classification of mines using feed-forward deep neural networks

Vasyl Lytvyn[1,†], Roman Peleshchak[1,*,†], Victoria Vysotska[1,†], Ivan Peleshchak[1,*,†], Lyubomyr Chyrun[2,†], Mariia Nazarkevych[1,†], Serhii Vladov[3,†], Olga Lozynska[1,†], Serhii Voloshyn[1,†], and Olena Nagachevska[1,†]

[1] *Lviv Polytechnic National University, Stepan Bandera 12, 79013 Lviv, Ukraine*

[2] *Ivan Franko National University of Lviv, University 1, 79000 Lviv, Ukraine*

[3] *Kremenchuk Flight College of Kharkiv National University of Internal Affairs, Peremohy Street 17/6 39605 Kremenchuk, Ukraine*

Abstract

The ongoing war waged by russia against Ukraine has accelerated the development of advanced technologies, including self-propelled artillery systems with integrated software, drones for enemy identification, and the widespread use of Starlink for internet connectivity in areas with limited access. A significant challenge facing Ukraine is demining territories liberated from temporary occupation. Official estimates indicate that over 30% of these areas are contaminated with explosive remnants of war, with 2.6 million hectares of agricultural land requiring urgent demining, severely disrupting the country's agrarian economy. Safely detecting, classifying, and neutralising these mines without risking human lives remains a pressing issue. Current detection methods rely on active sensors like ultra-wideband (UWB) radar. Although effective, these systems can inadvertently trigger mine explosions due to transmitted and reflected electromagnetic signals. In contrast, passive detection methods that do not activate detonating mechanisms offer a safer alternative. This research presents a sophisticated system for the passive detection and classification of mines using deep neural networks. Two models were developed: one with a single hidden layer and another with two hidden layers, achieving accuracies of 97.9% and 99.2%, respectively. The two-hidden-layer model demonstrated superior performance, surpassing a comparable k-NN heuristic algorithm by 1% in classification accuracy. Key advancements include reduced misclassifications, improved training efficiency, enhanced ROC curve performance, and an AUC exceeding 0.99, indicating exceptional efficacy in differentiating mine types. The F1 score of over 0.8 reflects the model's reliability, while loss metrics below 0.1 underscore the effectiveness of the training process. Recommendations for future work include developing datasets based on empirical data to enhance robustness and exploring parameter optimisation using more powerful hardware.

Keywords

mine detection, deep neural networks, passive sensors, demining technology, explosive remnants of war, classification accuracy, ROC curve, AUC value, F1 score, training efficiency.

1. Introduction

The detection of landmines remains a persistent and escalating global challenge, endangering millions of people due to the lethal threat posed by these explosive devices. In 2016, an average of 23 individuals per day worldwide were killed or severely injured by landmines or other explosive

CIAW-2024: Computational Intelligence Application Workshop, October 10-12, 2024, Lviv, Ukraine

* Corresponding author.

† These authors contributed equally.

Vasyl.V.Lytvyn@lpnu.ua (V. Lytvyn); roman.m.peleshchak@lpnu.ua (R. Peleshchak); Victoria.A.Vysotska@lpnu.ua (V. Vysotska); ivan.r.peleshchak@lpnu.ua (I. Peleshchak); Lyubomyr.Chyrun@lnu.edu.ua (L. Chyrun); mariia.a.nazarkevych@lpnu.ua (M. Nazarkevych); serhii.vladov@univd.edu.ua (S. Vladov); Olha.V.Lozynska@lpnu.ua (O. Lozynska); serhii.voloshyn.msaad.2022@lpnu.ua (S. Voloshyn); olena.o.nahachevska@lpnu.ua (O. Nagachevska)

0000-0002-9676-0180 (V. Lytvyn); 0000-0002-0536-3252 (R. Peleshchak); 0000-0001-6417-3689 (V. Vysotska); 0000-0002-7481-8628 (I. Peleshchak); 0000-0002-9448-1751 (L. Chyrun); 0000-0002-6528-9867 (M. Nazarkevych); 0000-0001-8009-5254 (S. Vladov); 0000-0002-5079-0544 (O. Lozynska); 0000-0002-5393-008X (S. Voloshyn); 0000-0002-5200-8085 (O. Nagachevska)

remnants of war. Currently, approximately 61 countries and territories remain contaminated by landmines, with thousands continuing to live under the daily threat of injury or death from these hidden dangers [1]. Traditional landmine detection and identification methods are no longer sufficiently reliable or efficient, necessitating the adoption of modern automated tools, such as neural networks. Developing a passive system for detecting and classifying landmines with high accuracy using neural networks and magnetic field sensors is an urgent priority. Russia's war against Ukraine has further accelerated the deployment of innovative technologies by Ukrainian forces, including:

- *Cyber Attacks*. Ukraine swiftly migrated its digital infrastructure to the public cloud, hosted across European data centres. Through collaborations with international tech companies such as Cloudflare and Microsoft, Ukraine has bolstered the resilience of its encryption and systems.
- *Satellites*. Approximately 25,000 Starlink terminals have been deployed, supporting military operations and Ukrainian civilians deprived of internet access. Other commercial space enterprises have contributed to Ukraine's military efforts through remote sensing and satellite communications. For instance, Ukrainian entrepreneur Serhiy Prytula facilitated the acquisition of a satellite and access to ICEYE's data repository.
- *Drones*. Unmanned aerial vehicles (UAVs), including Bayraktar, Furia, and Valkyrie, are used for reconnaissance and strike missions. Civilian drones have also been widely repurposed for reconnaissance.
- *Artificial Intelligence* (AI). Ukrainian AI firm Primer has adapted AI-powered speech transcription and translation services to process intercepted Russian communications, automatically highlighting intelligence on Ukrainian forces. AI-powered facial recognition technology from Clearview AI has also been employed to identify deceased Russian personnel through their social media profiles [2].

A critical issue for Ukraine is the demining of liberated territories formerly under occupation. The State Emergency Service of Ukraine reported via its official Telegram channel that approximately 175,000 square kilometres of territory remain potentially hazardous due to explosive remnants of war – equivalent to 30% of the country's total land area. Moreover, 2.6 million hectares of agricultural land require demining due to the Russian invasion, significantly threatening Ukraine's agricultural output. Landmines pose a substantial obstacle to Ukraine's post-conflict reconstruction. Even before the full-scale invasion, an estimated 1.8 million Ukrainians lived in mine-affected areas since 2014, according to the United Nations Office for the Coordination of Humanitarian Affairs in Ukraine [3].

The ongoing war has triggered a large-scale humanitarian crisis, including the widespread use of landmines and other explosive devices, resulting in the fastest-growing refugee population since World War II. Anti-personnel and anti-vehicle mines, along with unexploded ordnance in Ukraine, present a severe threat to millions of people. Clearing these mines will take years, hindering reconstruction efforts and endangering displaced persons returning to their homes. While large-scale demining is not feasible during the conflict, efforts are underway to coordinate support for Ukrainian authorities in identifying, locating, and removing explosive devices wherever possible [4].

Landmine detection uses various methods, many of which employ active sensors. However, active sensors may inadvertently trigger mine explosions because they rely on transmitted and reflected signals. Passive detection methods, which do not activate detonating mechanisms, offer a safer alternative. Nonetheless, passive detectors are typically less effective than their active counterparts. Studies indicate that machine learning algorithms can significantly enhance their performance.

This project envisions the safe detection and neutralisation of landmines. The system is designed to disarm mines during military operations and clear mined areas after they are liberated. On a global scale, this system could be utilised to clear mine-contaminated areas resulting from past conflicts, including those dating back to World War II. The primary objective of this work is the efficient detection and classification of landmines using a neural network. The system will classify mines

based on input data, which includes a vector of three values: voltage, height above ground, and soil type. The output will display the most likely mine type.

The work aims to develop an optimal neural network structure for effectively recognising different types of mines depending on the soil structure. The system's main task is the classification of mines using a deep artificial neural network of direct propagation. The object of research is the process of classifying mines of different types located in soils with other structures. The research subject is the methods and means of creating a neural network system for classifying mines during the execution of a combat mission or demining by public protection services. In particular, the mine classification process is investigated using a forward propagation artificial neural network with one and two hidden layers. The task of the work is to develop an optimal neural network system of direct propagation for recognising different types of mines located in various soils, using data from magnetic field sensors. For the successful development of the system, the following sequence of tasks was formed:

1. The first task involves the analysis of the current state and prospects in the field of detection and recognition of mines of various types by neural networks. It is necessary to analyse scientific publications that are freely available on the Internet and have similar functionality.
2. The second task involves system analysis and modelling of the neural network system. The result is a formalised unified description of business processes, project requirements, risks, objects of information, material, resource flows and other project components.
3. Project development involves formulation of the problem, construction of models to solve the problem, description of the methods used, selection of development tools, and direct development of the system. The result of the task is a fully or partially finished software product, which is supposed to solve the given problem.
4. Project testing involves the analysis of execution results, their evaluation, variation and validation. Deployment consists of developing a documented description of actions related to installing the system and its operation. The result of the implementation is a neural network system that has undergone verification and validation.

The innovative contribution of this research lies in developing a complete system that allows safe and accurate detection and classification of mines buried in the ground. The described approach involves detecting and recognising mines located in soils with different structures using magnetic field sensors and a deep neural network of direct propagation.

2. Related works

2.1. Analytical review of research and development in military technologies utilising artificial intelligence

Mine detection and classification systems are software solutions integrated into sensors, robotics, drones, or vehicles. Landmine detection and classification systems are software solutions increasingly integrated into advanced technological solutions, including sensors, robotics, unmanned aerial vehicles (UAVs), and autonomous ground vehicles. A modern approach to landmine detection challenges incorporates machine learning techniques such as computer vision, classification algorithms, and deep learning. These methods significantly enhance detection accuracy by improving upon traditional technologies, which would otherwise be inadequate without sophisticated software integration. This review concludes with a comparative analysis of existing systems alongside our proposed solution, followed by critical conclusions drawn from this comparison. One of the significant advancements in this field is using unmanned aerial vehicles (UAVs) for landmine detection. Recent progress in UAV-based remote sensing, employing lightweight multispectral and thermal infrared sensors, has rapidly detected landmine contamination across large areas, facilitating efficient mapping and detection efforts. Researchers at Binghamton University have focused on developing and testing automated remote detection techniques for anti-

personnel mines, particularly in identifying scattered anti-personnel landmines. Their study highlights the severe and long-lasting humanitarian and economic threats posed by the remnants of scattered plastic mines, such as PFM-1, which continue to affect communities in post-conflict regions. The methodology employed by these researchers is particularly suited for detecting plastic landmines containing liquid explosives encased in non-metallic materials such as polyethene or plastic. The system leverages multispectral and thermal datasets collected via an automated UAV imaging system, with PFM-1-type landmines serving as test subjects. The research team sought to automate landmine detection using supervised learning algorithms, precisely the Faster Regional-Convolutional Neural Network (Faster R-CNN). Their trials using RGB-visible light imaging combined with Faster R-CNN resulted in a detection accuracy of 99.3% for partially concealed landmines, while fully concealed landmines were detected with 71.5% accuracy.

In several test scenarios, combining centimetre-scale georeferencing datasets with the Faster R-CNN algorithm enabled the accurate autonomous detection of test PFM-1 landmines. The potential of this approach extends to humanitarian demining operations, with the capability to calibrate the method for detecting other types of scattered anti-personnel mines. It could be crucial in demining efforts in various post-conflict areas worldwide. The research collected field data under different environmental conditions to best model real-world scenarios. These conditions included sparse vegetation in Chenango Valley State Park, agricultural and pasture fields at Binghamton University, and snow-covered terrain after three inches of snowfall. The collected datasets are proxies for minefields under desert, farming, and winter conditions, respectively. While these environments may not perfectly replicate real minefields, they offer reliable spectral analogues, allowing for comprehensive testing of the detection system across various landscapes Data was collected using advanced sensor technology, including the FLIR Vue Pro thermal infrared sensor and the Parrot Sequoia multispectral sensor, mounted on a DJI Matrice 600 Pro UAV platform. Ground control points (GCPs) were strategically placed at grid intersections, and precise geospatial data was gathered using the Trimble Geo 7x handheld global navigation satellite system (GNSS). The UAV executed flight missions over simulated minefields, each containing 28–30 PFM-1 landmines scattered randomly within the grid to simulate real-world conditions. Multiple flight paths ensured the collection of extensive datasets later used for training and testing the CNN model. The convolutional neural network (CNN) employed for mine detection processed data with an average time of 1.87 seconds to detect PFM-1 landmines within a 10 × 20 m minefield. Scaling this to larger areas, the system could scan one square kilometre in approximately two hours and 36 minutes, achieving an overall mine detection accuracy of 71.5%. Each UAV flight covered a 10 × 20 m area in 3 minutes and 30 seconds. While the system demonstrated promising results, particularly with partially concealed landmines, the researchers acknowledged the need for further improvements to enhance detection accuracy for fully concealed mines. Detailed results of their findings are presented in Table 1 [5].

Table 1
Detailed Research Results

Training Sample	Training Time	Test Data	Test Time	Average Accuracy		
				PFM-1	KSF-Casing	Both Mines
Six flights, grass & rubble (Fall 2019)	37	One flight rubble (Fall 2017)	1.87	0.7030	0.7273	0.7152
Random 70% of 7 total flights	29	Random 30% of 7 total flights	5.47	0.9983	0.9879	0.9931

In the study [6], an experiment was carried out to generate data for passive mine detection and classify mines based on their magnetic field anomaly characteristics, soil depth, and soil type. This approach was designed to identify the specific type of mine by analysing variations in magnetic

anomalies. The method relies on three independent variables (input parameters): the type of soil (<*G*>) in which the mine is buried, the height of the detector above the ground (<*H*>), and the magnitude of the magnetic anomaly (<*V*>). A ferroprobe sensor was utilised to measure these magnetic anomalies. The detection process begins by determining whether the buried object is a mine. If a mine is identified, its type is then classified based on analysing the magnetic anomaly caused by the buried object. The classification considers the soil type and the distance between the sensor and the ground. Advanced machine learning algorithms are employed to classify the type of mine, where the mine type (<M_{type}>) is expressed as a function of the three independent variables: $M_{type} = f(V, H, S)$. The study further analysed the relationship between magnetic field anomalies and various environmental factors, including soil type (Figure 1) and depth (Figure 2), identifying specific magnetic anomalies associated with different mine types (Figure 3) [6]. This detailed analysis enhances the precision of mine classification and provides valuable insights into how magnetic field anomalies vary depending on the surrounding environmental conditions.

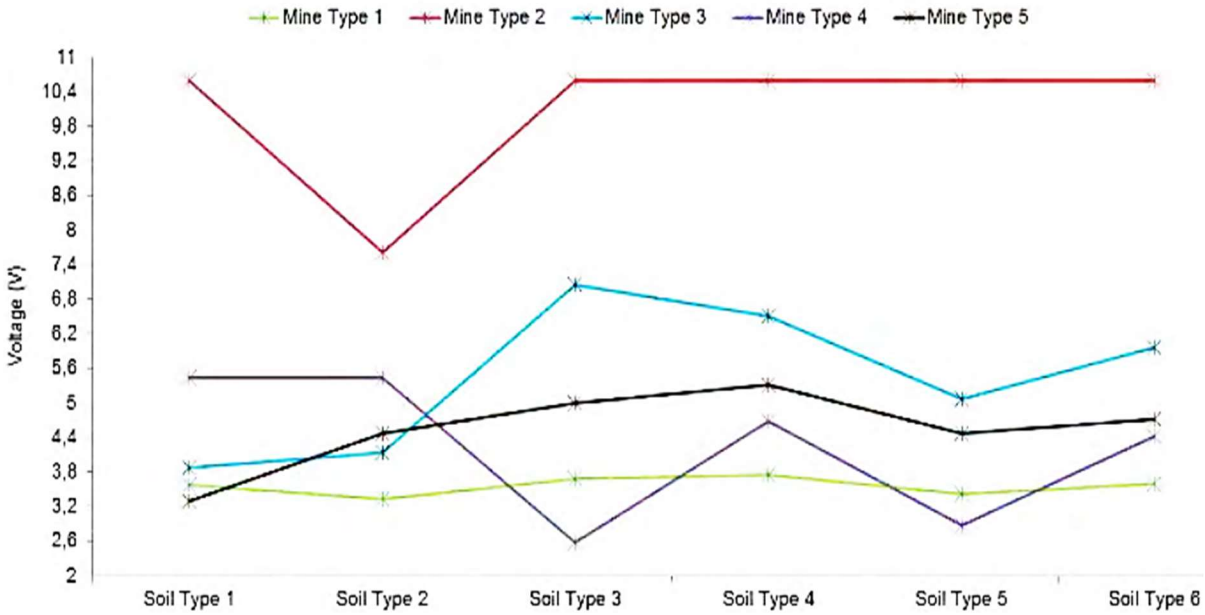

Figure 1: Magnetic field anomaly values relative to soil type for each mine type [6]

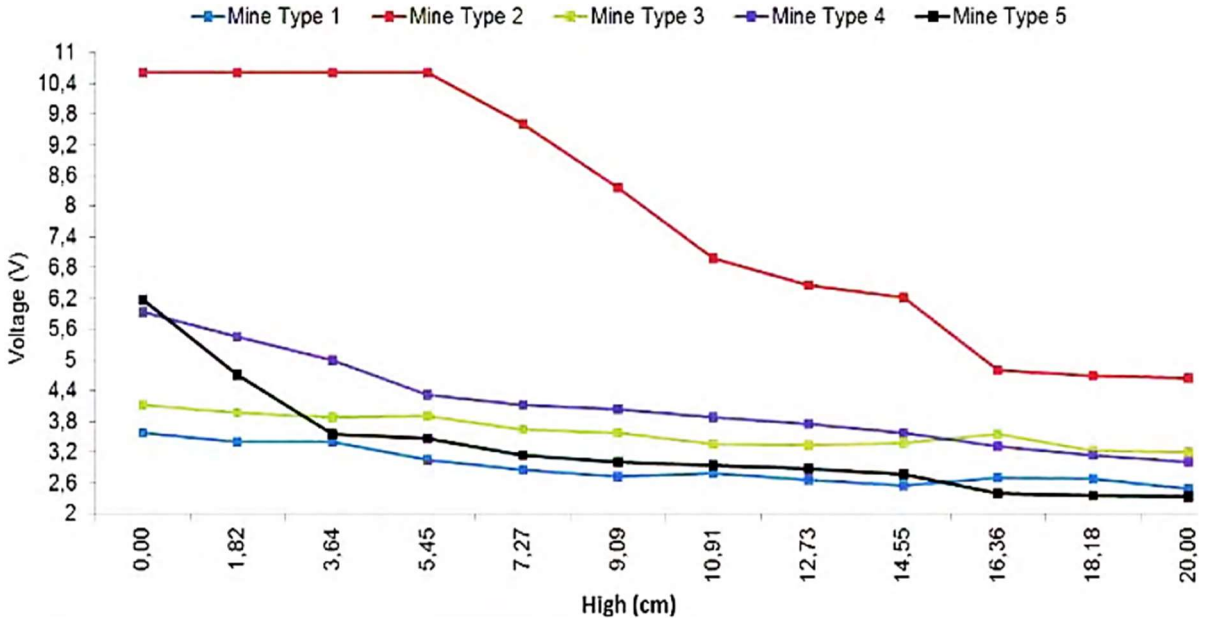

Figure 2: Magnetic field anomaly values relative to depth for each mine type [6]

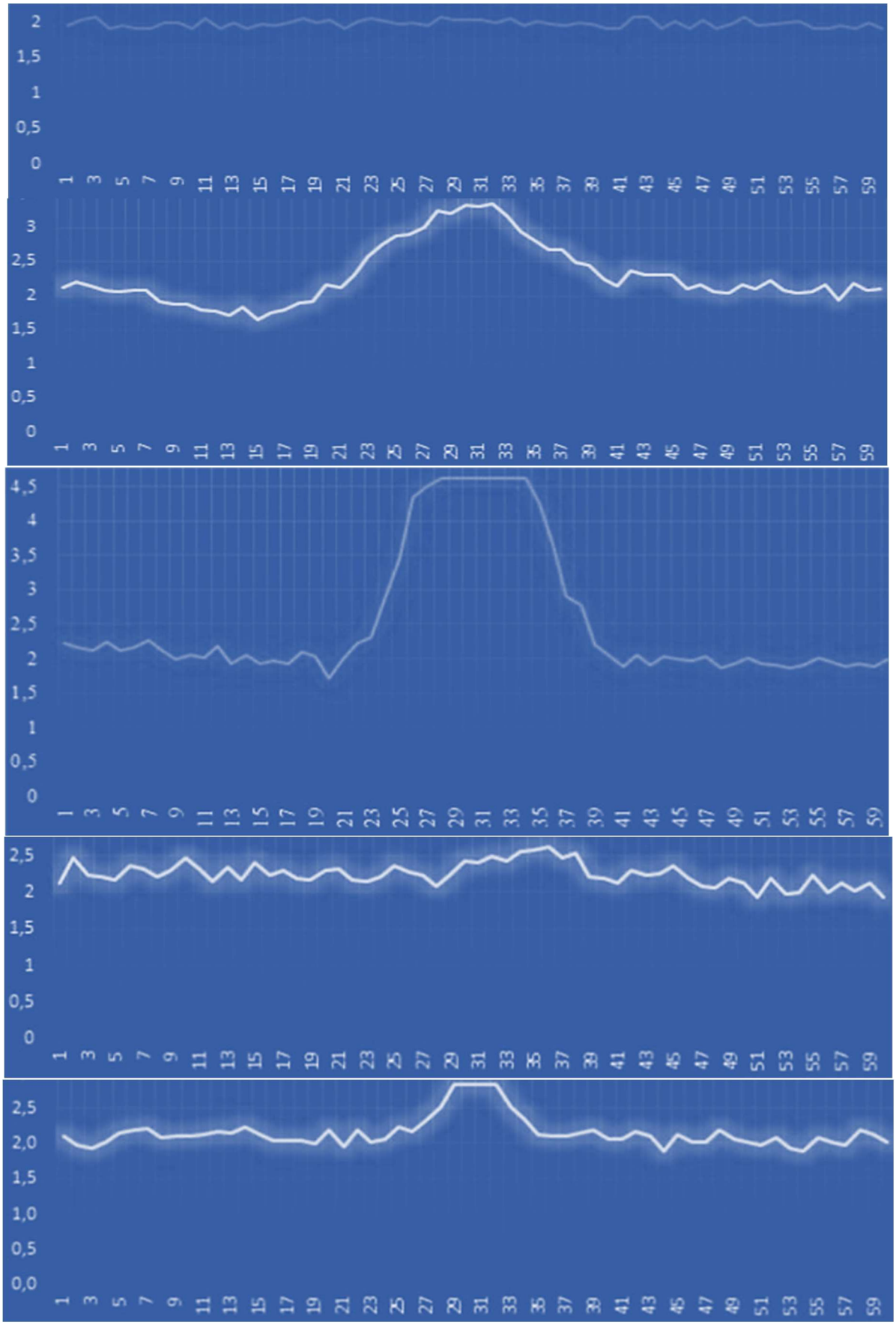

Figure 3: Graphs of the training dataset: a – Absence of a mine, b – Anti-tank mine, c – Anti-personnel mine, d – Booby trap mine, e – M14 mine, where x – sensor position (cm) and y – anomaly voltage (V) [6]

Having established the dataset, the researchers developed several machine learning algorithms, including models based on artificial neural networks (ANN) and k-nearest neighbours (k-NN), to address the problem of mine detection and classification. After conducting a series of experiments, they determined that the heuristic k-NN algorithm, enhanced with fuzzy metrics, was the most effective among the developed models. This approach achieved a mine detection efficiency of 98.2%, outperforming the other models. In contrast, the artificial neural network-based model demonstrated an average success rate of 95.6%, with an error rate of 4.4% [6].

2.2. Comparison of existing products

The "Landmine Detection Robot" is an advanced robotic system designed to identify landmines using integrated sensors and relay GPS coordinates of detected mines to a server, where an updated minefield map is maintained. Deploying multiple robots for mine detection enhances the identification of safe paths, a process conducted entirely autonomously without human involvement. The developers highlight that conventional mine clearance methods, such as manual tools, human-operated metal detectors, or machinery, are labour-intensive, costly, time-consuming, and pose substantial risks to personnel and equipment. The project aims to provide more efficient and sophisticated solutions for detecting, locating, and neutralising landmines, improving safety in affected areas. The system comprises three primary components: a web application, a robot, and a web server. The web application, deployed using Amplify Serverless methods, serves as a user access point within a designated user group. Upon logging in, users can access the main control page for managing a specific robot or the administrator page if they belong to an authorised user group. The robot's control page displays input data, control parameters, and a graphical representation of the search area's map, enabling comprehensive management of the mine detection process.

The robot is responsible for landmine detection, autonomous navigation, and data transmission at the project's core. The robot receives GPS coordinates from the server, stores search zone data and passively detects anti-personnel mines while navigating the search area. As it identifies mines, it updates the stored data and sends real-time information to the server, ensuring that the map and relevant data remain current.

Web servers facilitate communication between the hardware and users, handling tasks such as data storage, parameter calculations, and input/output operations. The initial data, entered by users through the interface, include GPS coordinates and the search area boundaries, which are sent to the servers for processing. Once the server's cloud functions are triggered, they calculate the search area's boundaries and parameters, which are then transmitted to the robot. These values are stored on the server and provided to the robot over a network connection.

Upon receiving the search parameters, the robot creates a data structure to track its path, detect mine locations, and determine the boundaries of the search area. It first retains this data locally and then periodically sends updates to the server via HTTP requests. These updates are stored on the server and accessible to the web application. As the data is processed, a virtual map displaying the mine locations and search progress is rendered in the user interface, providing users with visual and informative real-time data (Figure 4). This system architecture allows users to remotely monitor and control the mine detection process, facilitating more efficient and safer mine clearance operations [7]. The software developed for this project incorporates a diverse array of advanced tools and technologies:

- Python is the primary programming language for developing machine learning models, data preprocessing, and implementing auxiliary algorithms and methodologies.
- MATLAB is employed for comprehensive data analysis and signal processing throughout the project.
- C++ is utilised to establish hardware interfaces and manage the operation of the mine detection system.
- HTML/CSS/JavaScript is used to build the project's web-based user interface.

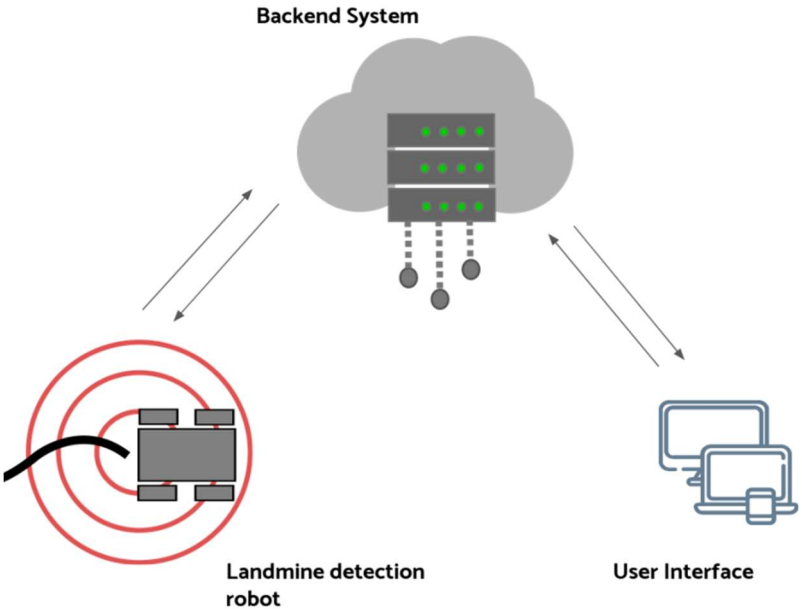

Figure 4: Data Flow in the Landmine Detector Project

The HOMARD project represents a cutting-edge research initiative to create an advanced system for detecting anti-personnel mines. This system integrates state-of-the-art robotics with sophisticated machine learning techniques, leveraging a combination of ground-penetrating radar sensors and advanced algorithms to detect landmines and other buried objects.

Within the HOMARD framework, various programming languages and tools are employed to address different aspects of the system. Machine learning algorithms are primarily implemented in Python, utilising well-known libraries such as TensorFlow and Keras. The robot control software and data collection system are developed through C++ and Python, ensuring efficient hardware-software integration.

To enhance detection accuracy, researchers in the HOMARD project have experimented with various machine learning models, including Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs). While specific data on the operational performance and accuracy of these models has not yet been publicly released, the system is described by its developers as "promising," with expectations of significant contributions to mine detection technologies [8].

Table 2
Comparative Analysis of the Developed Mine Detection and Classification System

Characteristics	Projects				
	Developed System	HOMARD	Landmine Detector	Faster R-CNN	Hybrid model
Functionality	Average	Average	High	Low	Low
Usability	Average	Average	High	Average	Low
Reliability	High	-	High	Average	Average
Performance	Average	-	High	High	Average
Utilises Python	Yes	Yes	Yes	-	-
Utilises C++	No	Yes	Yes	-	-
Employs Cloud Technologies	Yes	-	Yes	No	No
Integrates Robots or Drones (UAVs)	No	Yes	Yes	Yes	No
Accuracy of the Machine Learning Model	99.2%	-	-	98.2%	99.3%

This section provides an analysis of research from publicly available sources, as well as commercial projects. Several systems have been identified that use machine learning tools to detect mines. Among them are both commercial projects and research for creating such projects. As a result of the analysis, a table was formed, according to which it is possible to highlight the following trends in the creation and research of mine detection systems [9-12]:

- use of neural networks for accurate detection of mines such as convolutional, fully connected and their modifications;
- machine learning algorithms use data from sensors mounted on drones or robots to increase mine detection safety;
- systems additionally use a remote server for data collection, which has a positive effect on the speed of data processing;
- typical means of implementing machine learning algorithms are the Python and C++ programming languages.

3. Methods and means selection

3.1. Analysis of system functionality goals

The primary goal of the developed system is the high-precision detection and classification of landmines. A neural network model will be employed to achieve this, with continuous improvements driven by training on large datasets [12-18]. The system will offer remote access through a graphical user interface (GUI) and integrated sensors. Additionally, the system must be hosted on a cloud service to optimise performance. Objectives:

- Objective 1 is to ensure High Detection Accuracy of Mines. The system must achieve a high classification accuracy, measured by the Accuracy metric for the neural network model. It will be accomplished by designing and training a robust model with extensive and diverse datasets.
- Objective 2 is to provide a Remote User Interface. The system should allow users to interact with the software remotely without requiring direct sensor connections. Users will input data via the interface to receive classification results, ensuring ease of use and flexibility.
- Objective 3 is to develop a Graphical User Interface. The GUI will bridge the software and the sensors, displaying critical data in a user-friendly format, including changes in magnetic field anomalies and proximity to buried objects.
- Objective 4 is to facilitate Data Addition and Updating. Continuous improvement of the neural network model requires automatically expanding and updating the dataset. The software must support remote access to a cloud service for storing and managing classified data.
- Objective 5 is to Enable Interaction with the Neural Network Model. A user-friendly interface is necessary for interacting with the neural network, allowing users to train the model from scratch, continue its training, create network copies, and save modifications. These features will empower users to enhance the model's capabilities over time.
- Objective 6 is to Ensure Data and Model Security is paramount for the integrity of the dataset and the model. The system must implement robust usage restrictions, preventing unauthorised alterations to the neural network's architecture or the stored data.

The following risks must be anticipated:

- *False Classification of Mines.* The neural network may incorrectly classify objects as mines or non-mines, impacting the system's reliability.

- *Loss of Connection to the Remote Server*. Operators working in areas with limited connectivity may experience difficulty accessing the remote server. In such cases, it is recommended that sensors with local storage capabilities be employed.
- *Non-Operational Remote Server*. Server failure could disrupt system functionality. To mitigate this risk, the software should be hosted on multiple servers from different providers.
- *Interference with Database and System Code*. Data integrity may be compromised due to malicious actions or transmission errors. Safeguards must be implemented to protect against unauthorised access and system corruption.

As a result of identifying the system's objectives and risks, a detailed set of requirements has been established, summarised in Table 3 [18-27].

Table 3
Formation of Requirements

Type of Requirement	Business Requirements	User Requirements	Functional Requirements	Non-Functional Requirements
Purpose	They define the tasks and actions of users that the startup will support	They define the objectives of the business structure that the startup will achieve and the problems it will solve.	Description of what the system being developed in the innovative startup should do	Description of how the system being developed in the innovative startup should operate to perform its functions.
Content of the Requirement	Soldiers will have the ability to effectively and safely detect landmines. Project boundaries: mined areas worldwide. Effects: timely, secure, high-precision.	Users of the system are individuals who neutralise explosive devices. The system enables users to perform detection and classification tasks for landmines. The effect of task execution is providing a safer area post-demining compared to conventional methods.	The system can detect and classify landmines remotely and using sensors with specialised software.	Anyone can use the system. For successful classification, users must input specific data at given intervals. The system's quality attributes are accuracy, speed, and safety.

3.2. Modelling System Requirements

The modelling of requirements for the passive detection and classification system for landmines will be conducted using a use case diagram. Based on the defined objectives for system development, three primary actors have been identified: the soldier, the application, and the developer. The soldier is the leading actor in the system. The outcome of the requirement modelling process is creating a use case diagram that reflects both functional and non-functional requirements, as well as the interactions between the actors and the system. As an actor in the system, the soldier seeks to obtain

information regarding the presence of a landmine in a specific area of land. The prerequisites for this use case include successfully installing the software on the sensor, remote access to the user interface, the sensors' operational status, and network access [27-34]. If the software is connected remotely, the reliability of the cloud service is also required.

The use case diagram (Figure 5) illustrates the following critical actions of the soldier:

- Connect to the remote server.
- Input data from the sensor into the remote server.
- Receive classification results.

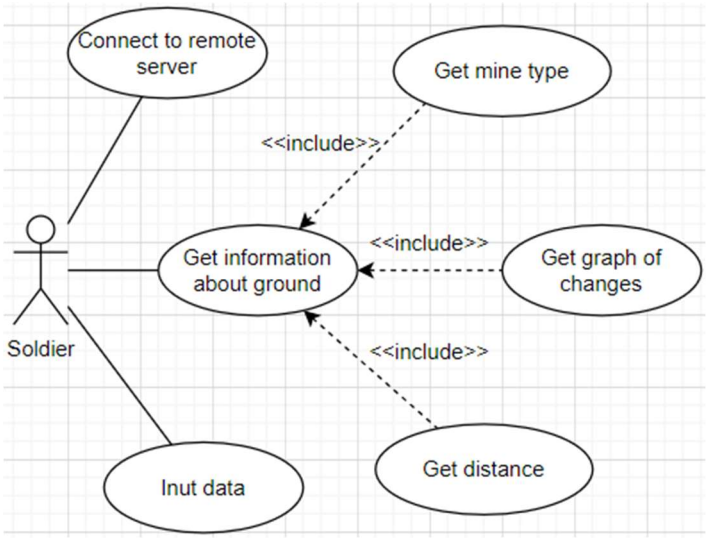

Figure 5: Use case diagram for the Soldier actor

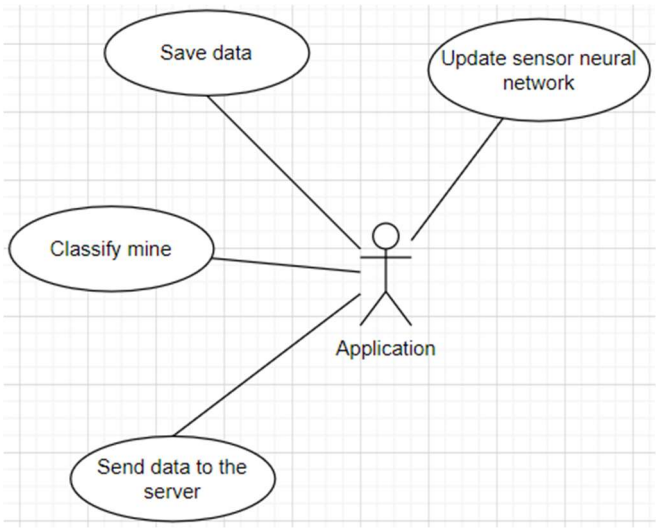

Figure 6: Use Case Diagram for the Application Actor

The soldier will receive a definitive result regarding the type of mine if the remote interface is utilised. Suppose the user employs a sensor already connected to the software. In that case, additional information will be displayed, including the distance to the ground surface and a graph depicting the changes in depth readings and magnetic field anomalies.

The software in the developed system will perform specific actions autonomously. Therefore, creating an actor that reflects this functionality, namely the Application, is appropriate. This actor represents the back-end component of the project, which will be hosted on a cloud service. The

prerequisites for this use case include the successful deployment of the project and the database on the cloud service. The diagram (Figure 6) illustrates the following main tasks of the Application:

- Data storage involves receiving and storing data that has been successfully classified in real-time.
- Mine classification occurs when a request is made by the soldier remotely.
- Data submission to the database.
- It updates the neural network model if installed on the local device.

The software, as well as the neural network model, requires technical support following successful deployment. In this case, the actor of the Programmer is necessary (Figure 7). The model should not be updated automatically, as this could decrease accuracy, necessitating a rollback to a previous version. The responsibilities of the Programmer include analysing the obtained data, training and retraining the neural network, updating the software, and interacting with the cloud service.

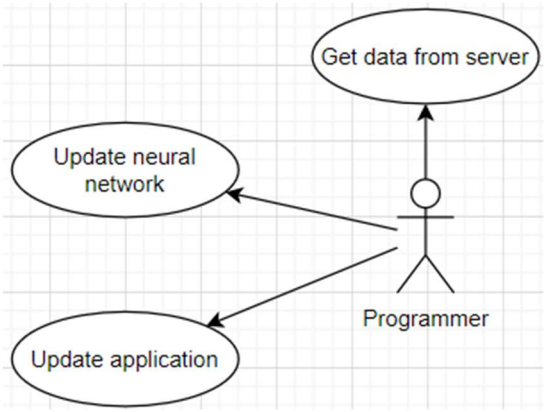

Figure 7: Use Case Diagram for the Programmer Actor

3.3. Modelling system objects

The modelling of system objects will be executed through the utilisation of class diagrams (Figure 8). Most of the described methods align with the use case diagram while incorporating methods for intermediate calculations. In addition to the previously identified actors, supplementary classes such as Interface, Dataset, and Data will be introduced to augment functionality. The relationships between these classes are delineated as follows:

- Soldier-Interface: an association link, signifying that the soldier employs the interface to classify mines;
- Application-Interface: an association link illustrating that the application for its operational functionality leverages the interface;
- Application-Dataset: a composition link of one-to-one, indicating that the dataset constitutes an integral component of the application;
- Dataset-Data: an aggregation link of one-to-many, demonstrating that data is an essential part of a singular dataset;
- Application-Programmer: an association link, as the programmer influences and possesses the capability to modify the application.

The soldier class represents the corresponding actor from the use case diagram. It includes attributes such as an identifier and a name, which are essential for storing the user in the database. The class also implements all use case scenarios and additional intermediary functions such as

retrieving distance, obtaining the mine classification, and generating graphs. A more detailed description of the methods and attributes can be found in Table 4.

The Class Interface is essential for establishing interaction between the soldier and the application. This class lacks attributes; however, it encompasses functions such as displaying analysis results, receiving input data, and visualising the type of mine, distance, and changing indicator graphs. A more detailed description of the methods and attributes is provided in Table 5.

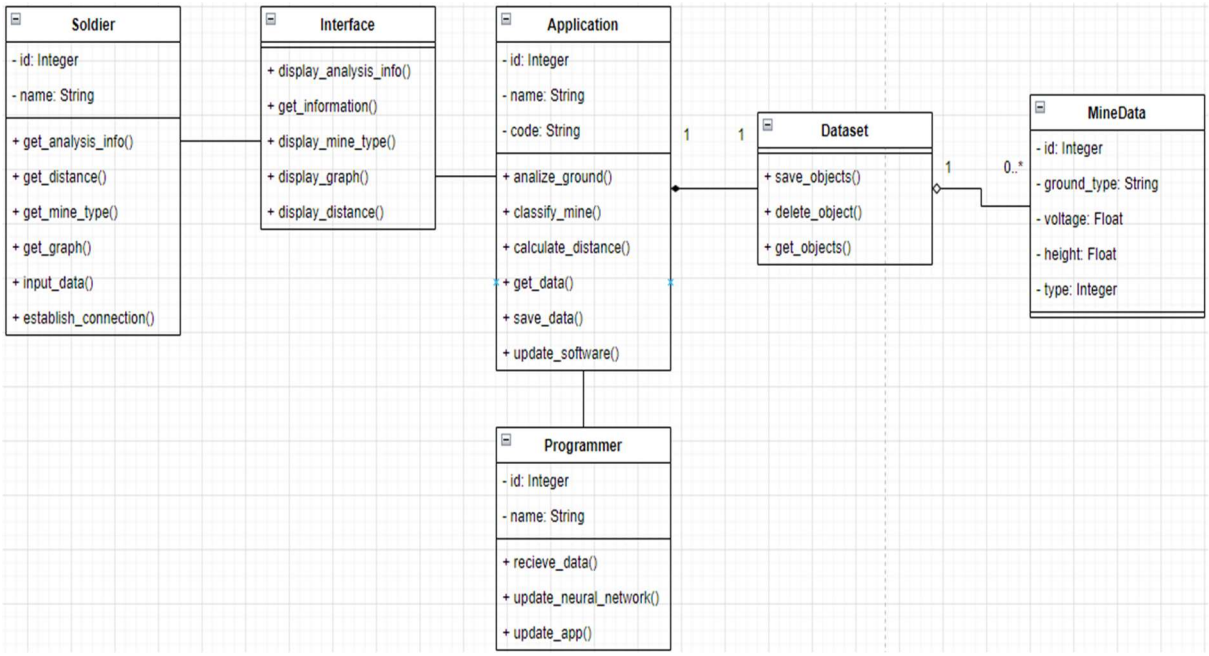

Figure 8: Class Diagram

Table 4
Description of the Soldier Class Objects

Classes of Objects		Class Attributes		Class Methods	
Class Name	Purpose of the Class	Attribute Name	Attribute Content	Method Name	Action Content
Soldier	The main actor of the system	id	Identifiers it in the database	get_analysis_info	Receive classification and additional information about the ground
		name	Soldier's full name	get_distance	Display the distance from the sensor to the ground
				get_mine_type	Display mine type
				get_graph	Display distance and anomaly changes as a curve
				input_data	Enter required data for further classification.
				establish_connection	Establish a connection with a remote server

The Application Class represents the corresponding actor within the system. It encompasses attributes such as an identifier, name, and code for its storage in the database. The application implements various methods, including conducting data analysis, classifying mines, calculating the

distance from the sensor to the surface, retrieving input data, storing data, and updating software on the local device. A more detailed description of the methods and attributes can be found in Table 6.

Table 5
Description of the Interface Class Objects

Object Classes		Class Attributes		Class Methods	
Class Name	Class Purpose	Attribute Name	Attribute Content	Method Name	Action Content
Interface	API between Soldier and Application			display_analysis_info	Display all information after successful classification
				get_information	Receive input data from soldiers.
				display_mine_type	Display mine type
				display_graph	Display distance and anomaly changes as a curve
				display_distance	Display the distance from the sensor to the ground.

Table 6
Description of the Application Class Objects

Object Classes		Class Attributes		Class Methods	
Class Name	Class Purpose	Attribute Name	Attribute Content	Method Name	Action Content
Application	Responsible for access to neural network model and manipulation of data	id	Identifies objects in the database	analyse_ground	Perform analysis of the ground, start the classification process
		name	Application name	classify_mine	Perform mine classification based on input data
		code	Unique application code within the system	calculate_distance	Calculate the distance between the sensor and the ground
				get_data	Receive input data
				save_data	Save data to the database.
				update_software	Update software in the local sensor

The Dataset class represents a data collection essential for training the neural network model. Its primary purpose is to facilitate data manipulation. As such, the class does not possess any attributes; however, it includes the following functions: save object, delete object, and retrieve object. A more detailed description of the methods and attributes can be found in Table 7. The Data class represents information about the mines with which the neural network model directly interacts. While the class does not include any methods, it encompasses the following attributes: identifier, voltage, distance, soil type, and mine type. A more detailed description of the methods and attributes can be found in

Table 8. The Programmer class is responsible for representing the corresponding actor in the system. This class includes the following attributes: identifier and name. Additionally, it contains the following methods: retrieve application data, update the neural network model, and update the software. A more detailed description of the methods and attributes can be found in Table 9.

Table 7
Description of the Dataset Class Objects

Object Classes		Class Attributes		Class Methods	
Class Name	Class Purpose	Attribute Name	Attribute Content	Method Name	Action Content
Dataset	Responsible for manipulation of data needed for neural network model			save_ object	Save the object to the database.
				delete_ object	Delete the object from the database.
				get_ object	Get an object from the database.

Table 8
Frequency of Special Characters

Object Classes		Class Attributes		Class Methods	
Class Name	Class Purpose	Attribute Name	Attribute Content	Method Name	Action Content
Dataset	Responsible for manipulation of data needed for neural network model	id	Identifies objects in the database		
		ground_ type	Represents one of the ground types		
		voltage	Display magnetic anomaly		
		heigh	Displays the distance between a ground and a sensor		
		mine_ type	Represents correct mine type for an object		

Table 9
Description of the Programmer Class Objects

Object Classes		Class Attributes		Class Methods	
Class Name	Class Purpose	Attribute Name	Attribute Content	Method Name	Action Content
Program-mer	Responsible for updating neural network and application	id	Identifies objects in the database	receive_data	Get data from the database and get code from the remote system
		name	Represents the name of the programmer	update_ neural_ network	Update neural network model after additional training
				update_ application	Update application

3.4. Modelling system processes

The modelling of system processes will be conducted using activity and sequence diagrams. These diagrams will illustrate the primary successful scenario of how the user interacts with the software product. The successful scenario represented in the activity diagram (Figure 9) comprises the following sequence of steps:

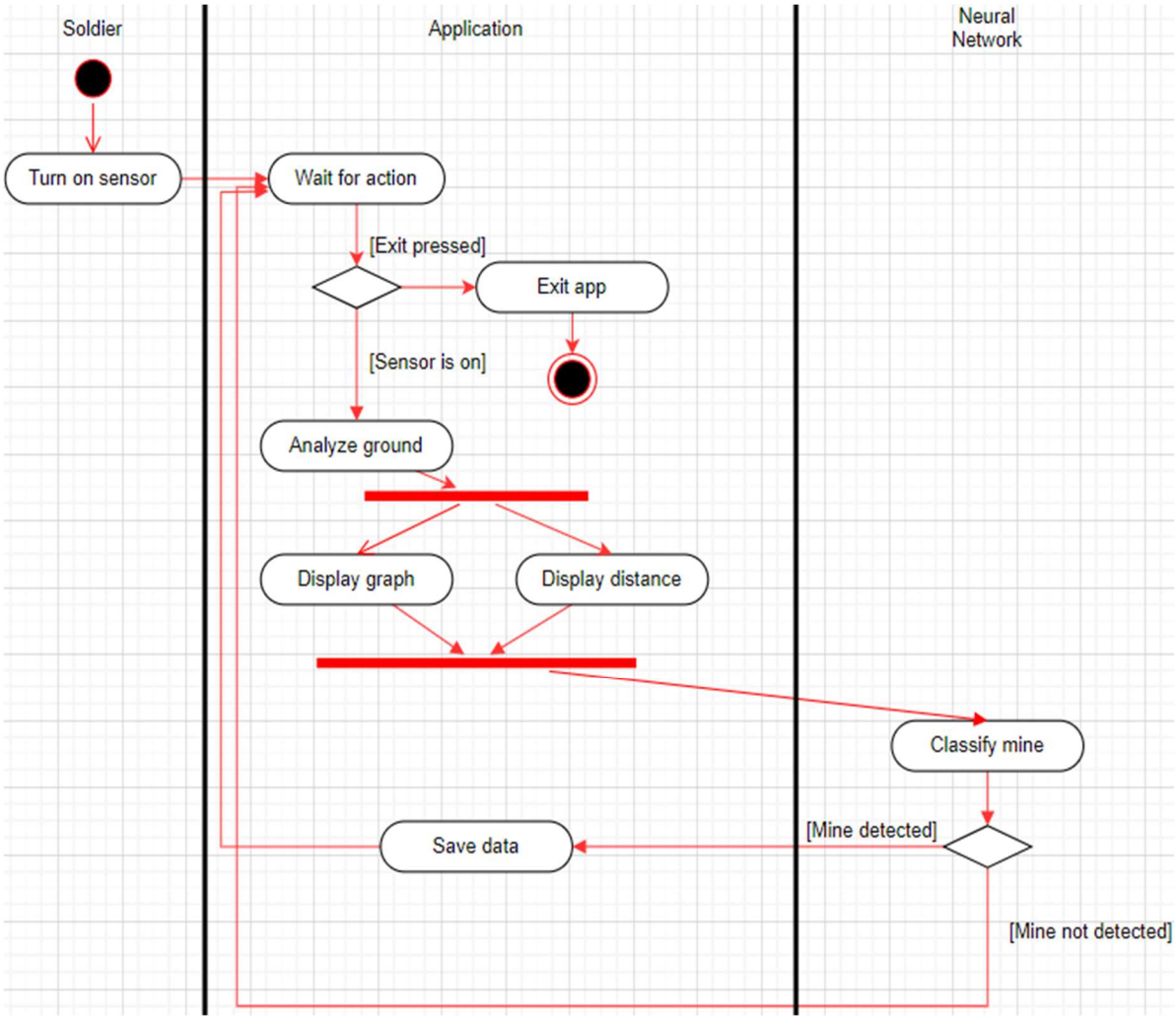

Figure 9: Activity Diagram

1. Activating the Sensor
2. Awaiting User Input
3. If the application remains active, soil analysis is conducted. Otherwise, the application terminates, and the user exits.
4. Concurrently, the system displays a graph illustrating variations in the magnetic field and the distance to obstacles.
5. Mine Classification
6. If a mine is detected, the data is stored in the application, and the program returns to step 2. If no mine is present, the program continues to operate from step 2.

The successful scenario illustrated in the sequence diagram (Figure 10) encompasses the following sequence of steps:

1. Conducting Soil Analysis
2. Displaying the Magnetic Field Variation Graph

3. Calculating the Distance to Obstacles
4. Classifying the Mine
5. Presenting the Classification Results
6. Storing Classification Data
7. Displaying the Soil Analysis Results

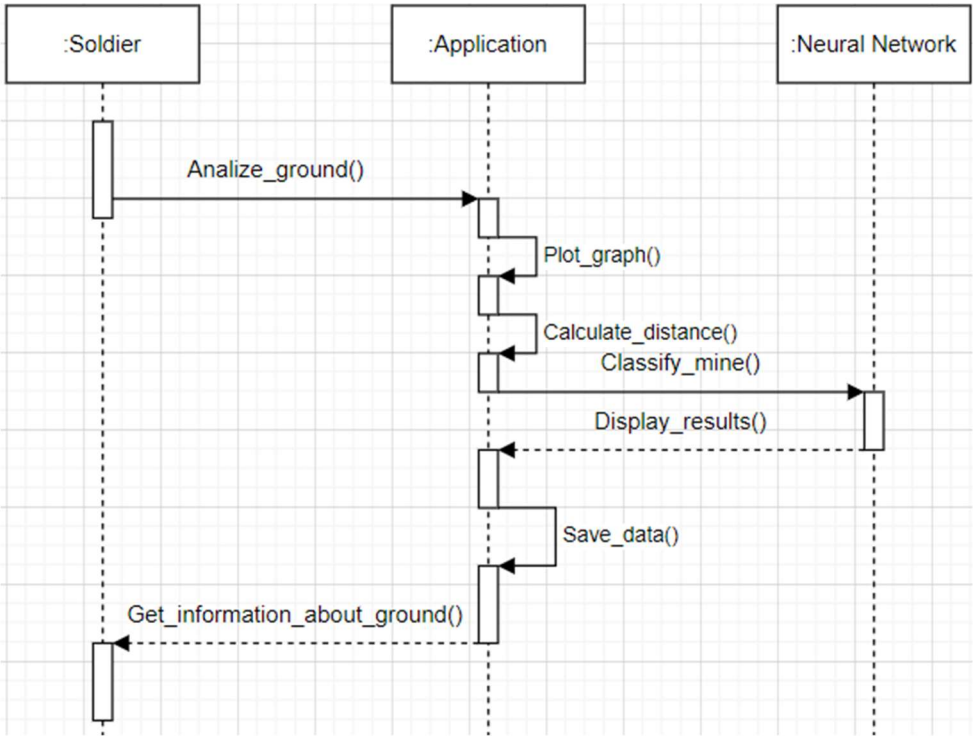

Figure 10: Sequence Diagram

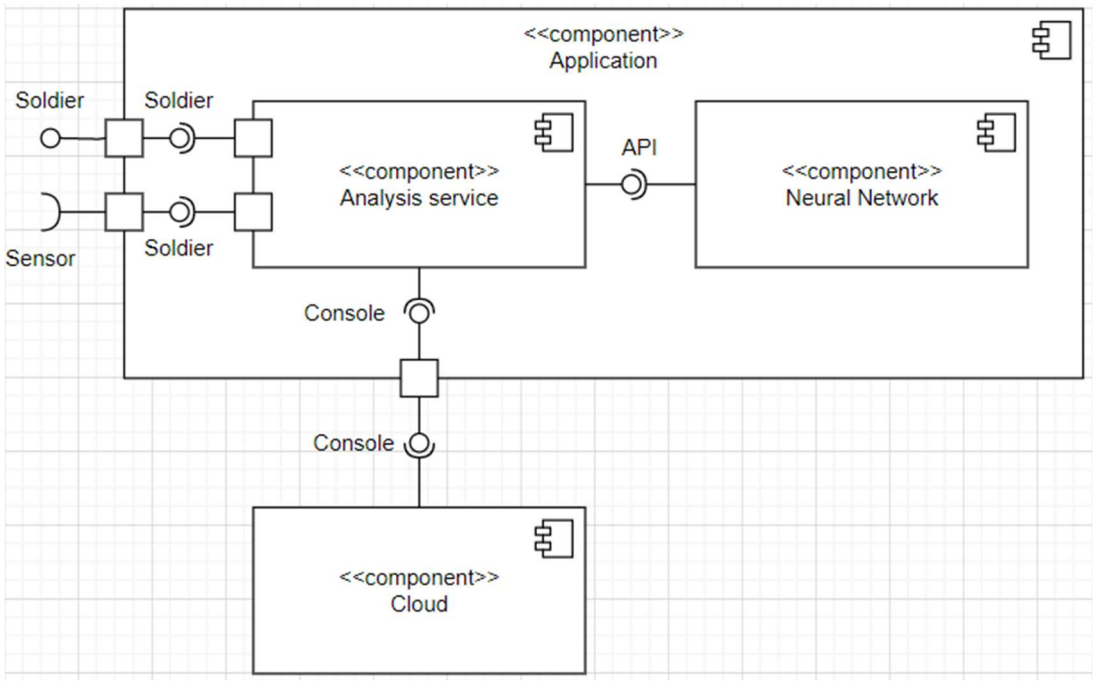

Figure 11: Component Diagram

The component diagram (Figure 11) illustrates two primary components: the software and the cloud service. The software encompasses elements such as the soil analysis service and the neural

network for mine classification. Access to the service is facilitated through a user interface and sensor integration, while interaction with the neural network occurs via an API. The cloud service operates independently of the software and is designed for remote data storage. Access to the service is conducted through console commands.

The operational objectives of the system have been meticulously articulated, encompassing the identification of principal users and a comprehensive delineation of functional goals. The requirements have been systematically formulated and modelled using a use case diagram. Object modelling has been executed via class diagrams, which offer an in-depth exposition of the class structures. Furthermore, the modelling of system processes has been successfully carried out, with a particular emphasis on the primary successful use case scenario, employing both activity and sequence diagrams.

This section describes the purpose of the system's operation, the primary users, and the operation's objectives in detail. The requirements were formulated and modelled using a usage diagram. Object modelling was done using a class diagram and described in detail using class diagrams. Successfully modelled system processes, namely the main successful use case, using activity and sequence diagrams.

4. Development of the project solution

4.1. Problem formulation and justification

The primary objective of this research is to devise a comprehensive conceptual framework for a software system aimed at the passive detection and classification of landmines. The system's core functionality categorises landmines by applying a deep artificial neural network, which delineates classifications into five distinct categories.

The focus of this investigation is the classification process itself. The subject matter encompasses the methodologies and tools utilised in developing a landmine classification system, particularly within combat operations or demining efforts conducted by civil defence agencies. Specifically, this study explores the classification process by implementing a deep artificial neural network featuring configurations of one and two hidden layers and convolutional neural networks. Before the classification process, it is essential to address the challenge of generating additional values within the dataset, given that the current sample size is limited to 45 (representing the magnetic anomaly values at depths ranging from 26 to 34 centimetres for the "Dry and Humus" soil type associated with each of the five mine classifications). However, neural networks require training datasets that encompass several thousand instances. To tackle this subproblem effectively, the proposed approach employs a normal distribution function to facilitate data augmentation.

4.2. 3.2. Model construction for problem resolution

To illustrate the advantages of a deep neural network architecture, an alternative neural network with a single hidden layer has also been developed [33-35]. This model consists of an input layer comprising three nodes, followed by a first hidden layer containing seven nodes that utilise the ReLU activation function. The output layer consists of five nodes that employ the Softmax activation function. The Adam optimiser has been used with categorical cross-entropy as the loss function, while accuracy is the principal performance metric (Figure 12). The output of this neural network is quantitatively expressed by the relationship delineated in equation (1).

$$y_i = f_{softmax}(\sum_{j=1}^{7} w_{ij} f_{relu}\,(\sum_{k=1}^{3} w_{jk} x_k)), \quad i \in \{\overline{1;5}\}, \tag{1}$$

where y_i – represents the element of the output vector of probabilities corresponding to the association of the object with each class of mines, while $f_{softmax}$ – denotes the Softmax activation function, w_{ij} – refers to an element of the weight matrix between the first and second hidden layers and w_{jk} – represents an element of the weight matrix connecting the input layer to the first hidden

layer. Additionally, x_k – signifies an aspect of the input feature vector associated with the mine [33-35].

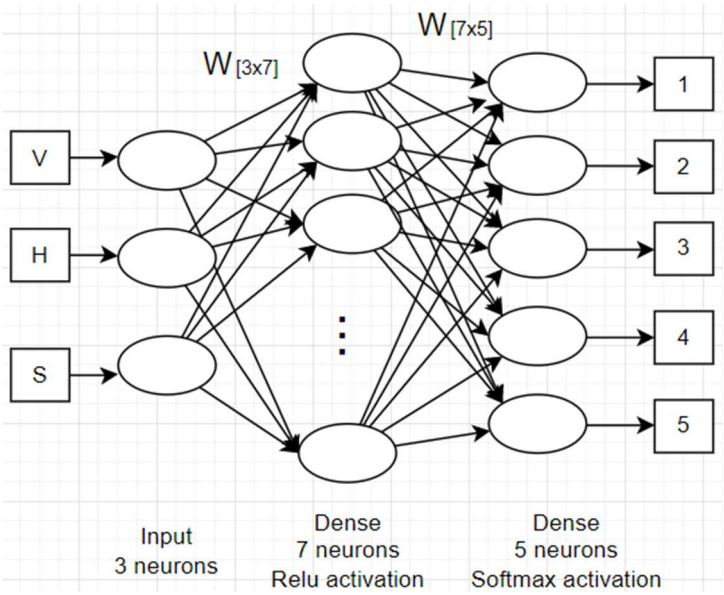

Figure 12: Architecture of the Neural Network Featuring a Single Hidden Layer

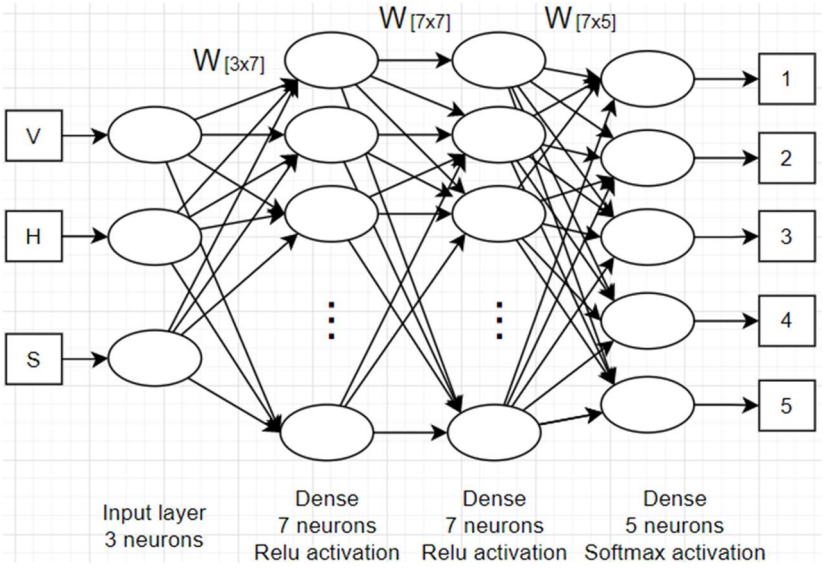

Figure 13: Architecture of the Neural Network with Two Hidden Layers

The neural network architecture with two hidden layers features an input layer size of 3, with the first layer comprising seven neurons and employing the ReLU activation function. The second hidden layer possesses identical characteristics to the first, while the output layer consists of 5 neurons utilising the Softmax activation function. The Adam optimiser is employed, with categorical cross-entropy as the loss function, and accuracy is utilised as the performance metric (Figure 13). The output of this neural network will be described by the equation (2).

$$y_i = f_{softmax}\left(\sum_{l=1}^{7} w_{li}\, f_{relu}\left(\sum_{j=1}^{7} w_{ij} f_{relu}\left(\sum_{k=1}^{3} w_{jk} x_k\right)\right)\right),\ i \in \{\overline{1;5}\}, \quad (2)$$

where y_i – represents an element of the output probability vector, indicating the relationship of the object to each class of mines; $f_{softmax}$ – denotes the Softmax activation function; f_{relu} – refers to the ReLU activation function; w_{li} – is an element of the weight matrix connecting the second hidden layer to the output layer; w_{ij} – represents an element of the weight matrix between the first and second hidden layers; w_{jk} – is an aspect of the weight matrix linking the input layer to the first

hidden layer; x_k – signifies an aspect of the input vector representing the characteristics of the mine [33-35].

4.3. Selection and justification of problem-solving methods

In alignment with the established objectives, ensuring high accuracy in mine detection is imperative. The primary challenge to be addressed is classification. In machine learning, classification pertains to training a model to categorise or classify input data into predefined classes or categories. It involves learning the decision boundary or mapping function that correlates the input features to the corresponding output labels. The goal is to accurately predict the class of unseen instances based on the patterns and relationships derived from the training data [36]. Several classification models are discussed below.

- Logistic Regression is a simple and widely utilised classifier that models the probability of membership in a specific class based on input features. It estimates coefficients for each feature and applies a logistic function for prediction. Logistic Regression is typically employed for binary classification tasks or scenarios where the outcome variable is categorical.
- Bayes Classifier is grounded in Bayes' theorem, operating under the assumption of independence among features. It computes the posterior probability for each class and selects the class with the highest probability. The Bayes classifier is often applied in text classification, spam filtering, and other high-dimensional data tasks.
- Decision Tree models construct a tree-like structure by recursively partitioning data based on feature values. Each internal node represents a feature test, while each leaf node corresponds to a class label. Decision trees are beneficial for classification and regression tasks and are recognised for their interpretability and ability to handle numerical and categorical data.
- The Random Forest method integrates multiple decision trees. Each tree is trained on a random subset of the data, and the final prediction is determined by majority voting or averaging the predictions from individual trees. Random forests are robust and efficient for classification and Regression, often employed in tasks with complex datasets.
- The Support Vector Machine (SVM) Classifier identifies the optimal hyperplane that separates classes by maximising the margin between them. It maps data into a higher-dimensional space to find a linear or nonlinear decision boundary. SVMs are commonly utilised for binary classification tasks but can be extended to multiclass problems. They are effective in high-dimensional spaces and can handle linear and nonlinear relationships.
- Neural Network models are structurally and functionally akin to biological neurons. Comprising interconnected layers of artificial neurons, neural networks learn from data through forward and backward signal propagation. They excel in tasks involving large datasets, complex patterns, and nonlinear relationships, finding widespread applications in image classification, natural language processing, and various other fields [37].

At this juncture, the objective of providing a graphical user interface will not be implemented, as it entails the development of a user sensor capable of detecting anomalies in the Earth's magnetic field. Functions stipulated by other objectives may be realised within the back-end portion of the system using cloud services.

4.4. Development of problem-solving algorithms

The generation based on the normal distribution entails the application of the standard distribution formula [38] and the generation of pseudorandom values [39].

$$p(V) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(V-\mu)^2}{2\sigma^2}}, \quad (3)$$

$$V_i^* = rand(V_i, \sigma_з), \quad (4)$$

where V_i – represents the magnetic field anomaly value at a depth H_i and soil type S_i; V_i^* – denotes the new magnetic field anomaly value at a depth H_i and soil type S_i; $rand(V_i, \sigma_з)$ is a function that generates pseudorandom values based on the normal distribution function, where μ – is the arithmetic mean and σ is the standard deviation [39].

Consequently, the new values will be generated closer to those in the training dataset, approximating the arithmetic mean and considering the standard deviation of the Earth's magnetic field anomaly. The standard deviation is computed as follows:

$$\sigma_з = avg(\sigma_i), i \in \{\overline{1; 15}\}, \quad (5)$$

where $\sigma_з$ – represents the standard deviation of the Earth's magnetic field anomaly; σ_i – denotes the standard deviation of the magnetic field anomaly at a depth H_i.

The index i – takes values from 1 to 15, as the study [6] indicated that this range corresponds to the distances at which the intensity of the magnetic anomaly of the mine is not detected.

The next phase involves data preparation. The following steps have been established for data preparation:

1. Normalise magnetic field anomaly data based on the mean value and standard deviation [40].

$$V' = \frac{V-\bar{V}}{\sigma_v}, \quad (6)$$

where V' denotes the normalised value of the anomaly; V represents the initial value of the anomaly; $\bar{V}$ signifies the mean value of the anomaly; σ_v indicates the standard deviation of the anomaly.

2. Additionally, encoding is applied to the soil type values, as these data are recorded as categorical variables.

The primary challenge to address is the classification of mines using a deep neural network. A classifier functions to establish a correspondence for each pair of object characteristics and their respective classes. In this study, the characteristics of the object (the mine) are represented by a vector comprising three values <V, H, S>, corresponding to the magnetic field anomaly value of the mine and the Earth in volts, depth in centimetres, and soil type, respectively. The classes of objects constitute a set of five values C = {0, 1, 2, 3, 4}, representing the types of mines: "no mine," "anti-tank mine," "anti-personnel mine," "booby trap," and "M14." The classification task is framed as the identification of a classifier that ensures the minimum norms in Euclidean space:

$$\min \|f - f_0\|, \quad (7)$$

where: f denotes the target classifier; f_0 represents the deep neural network [41].

The mapping operator is known solely for objects in a finite training sample:

$$X_m = \{(x_1, y_1), \ldots, (x_m, y_m)\}, \quad (8)$$

where X_m is the set of elements in the training sample with a size of m, the objective is to construct an algorithm capable of determining the membership of any object $x \in X$ to the class $y \in Y$ [41-42].

5. Experiments

5.1. Selection and justification of development tools

The central objective of the system is the classification task, with the primary focus of development being the design and implementation of an accurate neural network model. Neural networks are

typically developed using programming languages or specialised libraries. Below is a summary of the most widely employed programming languages in the realm of machine learning:

- Python is a widely utilised language for machine learning and neural network applications. It offers a robust ecosystem of libraries and frameworks such as TensorFlow, Keras, and PyTorch, which provide high-level abstractions and optimised implementations for neural networks. Python's simplicity and readability make it an ideal choice for novice and experienced developers, enabling rapid prototyping and seamless learning. The extensive and active Python community of data scientists and machine learning practitioners provides many resources, tutorials, and sample codes. Despite its numerous advantages, Python can perform slower than lower-level languages like C++, particularly in computationally intensive operations involving large-scale neural networks or tasks requiring maximal efficiency [43].
- R is traditionally employed for statistical computing and data analysis. Still, it also features several machine learning libraries, including TensorFlow, Keras, and MXNet, which facilitate the creation and training of neural networks. R's vast array of data analysis libraries makes it well-suited for statistical modelling and exploratory data analysis. Its data visualisation capabilities, exemplified by packages such as ggplot2, are highly effective for visualising neural network results. However, R's interpreted nature can lead to slower execution times when compared to languages like Python or C++, which may affect the performance of large-scale neural networks and other computationally intensive tasks. Moreover, R's automatic memory management could result in inefficiencies when handling memory-intensive processes [44].
- MATLAB is a programming language widely recognised in scientific and engineering domains. It offers toolkits like the Neural Network Toolbox and Deep Learning Toolbox, which provide a comprehensive suite of functions and algorithms for designing and training neural networks. MATLAB's interactive environment enables rapid prototyping, visualisation, and experimentation, facilitating development. However, MATLAB is commercial software that requires a paid license, rendering it less accessible compared to open-source alternatives such as Python. Additionally, its proprietary nature may limit flexibility and customizability relative to open-source languages [45].
- C++ is a high-performance programming language frequently used for systems programming and computationally demanding tasks. Libraries such as TensorFlow, Caffe, and Torch provide C++ APIs that enable the construction and training of neural networks. C++ offers superior performance to interpreted languages like Python, making it suitable for resource-intensive computations or large-scale neural networks. Its explicit memory management gives developers granular control over memory allocation and deallocation. However, C++ has a steeper learning curve and may be more challenging than higher-level languages like Python. The development process in C++ can be more time-consuming owing to its low-level nature and the necessity for manual memory management [46].

To achieve the additional goals of the system, it is necessary to develop the back-end component. The back-end is created using frameworks from various programming languages. Below is a description of the most commonly used programming languages for back-end development:

- Python boasts a robust ecosystem of libraries and frameworks, making it well-suited for web development, data analysis, and scientific computing. Python is often chosen for its ease of use, rapid growth, and strong community support. It is widely used for server-side web development, creating APIs, data processing, and scripting.
- JavaScript is a versatile scripting language primarily used for front-end web development. However, with the advent of Node.js, JavaScript can now also be employed as a server-side language. Node.js enables JavaScript development on the server side, making it an excellent

choice for building scalable, high-performance web applications. JavaScript (Node.js) is frequently used to create real-time applications, render server-side, and handle many simultaneous connections.

- Java is a robust object-oriented programming language known for its platform independence and scalability. It offers a wide range of libraries and frameworks for developing enterprise-level applications. Its extensive toolset, performance, and strong tool support make Java popular for large-scale server-side systems. Java is commonly used for enterprise application development, distributed systems, and server-side components.
- Ruby is a dynamic object-oriented programming language with an elegant and readable syntax. It emphasises simplicity and developer productivity and is known for its focus on developer satisfaction. Ruby features a mature web framework called Ruby on Rails, which promotes rapid application development and follows the convention-over-configuration principle. Ruby is often used for web development, prototyping, and building scalable web applications through the Ruby on Rails framework.
- PHP is a server-side scripting language designed specifically for web development. It is widely used for building dynamic websites and web applications. PHP has a large ecosystem of frameworks, such as Laravel and Symfony, which provide powerful tools and libraries for web development. PHP is commonly used for website development, content management systems (CMS), and e-commerce platforms.

5.2. Description of the developed project tools

The development of project tools includes creating the back-end component, a neural network model, and their deployment on a cloud service. During development, a neural network, an object generator, a graphical representation of the neural network's performance, the back-end component, and the necessary file for container deployment were created.

The software for the neural network consists of tools for dataset generation, data preprocessing, and the neural network model itself. As previously mentioned, number generation was performed using pseudorandom number generation methods available in Python libraries. Data preprocessing was handled by Panda's library, which is designed to work with large datasets. The steps carried out by this script were outlined earlier. The neural network models were developed using the TensorFlow library. The final product will use only the model with two hidden layers.

The back-end follows the basic structure of the Django framework, adhering to the Model-View-Template (MVT) architecture. All required functions were developed in the views file, while database tables were defined in the models file. The graphical interface is integrated using the Django Rest Framework. To implement the required functionality, the neural network model was added to the back-end files. As a result, when a user submits a classification request, the model will be loaded and used to classify the data.

In this section, the development tasks are formulated and substantiated. A mathematical model of direct propagation neural networks with one and two hidden layers was created, and their architecture was implemented. As a result of the analysis of development tools, the following basic functionality was chosen:

- The neural network will be implemented using Python, the programming language in particular, the TensorFlow library.
- The back-end part of the system will be implemented using the Python programming language and the Django framework.
- System deployment will be done using Docker software.

6. Results and discussion

6.1. System testing

The neural network's performance was rigorously assessed using key metrics, including Accuracy and AUC, alongside a series of visualisations that effectively capture the system's training and testing outcomes. The results of the neural network classification are illustrated through the following diagrams (Figure 14-17):

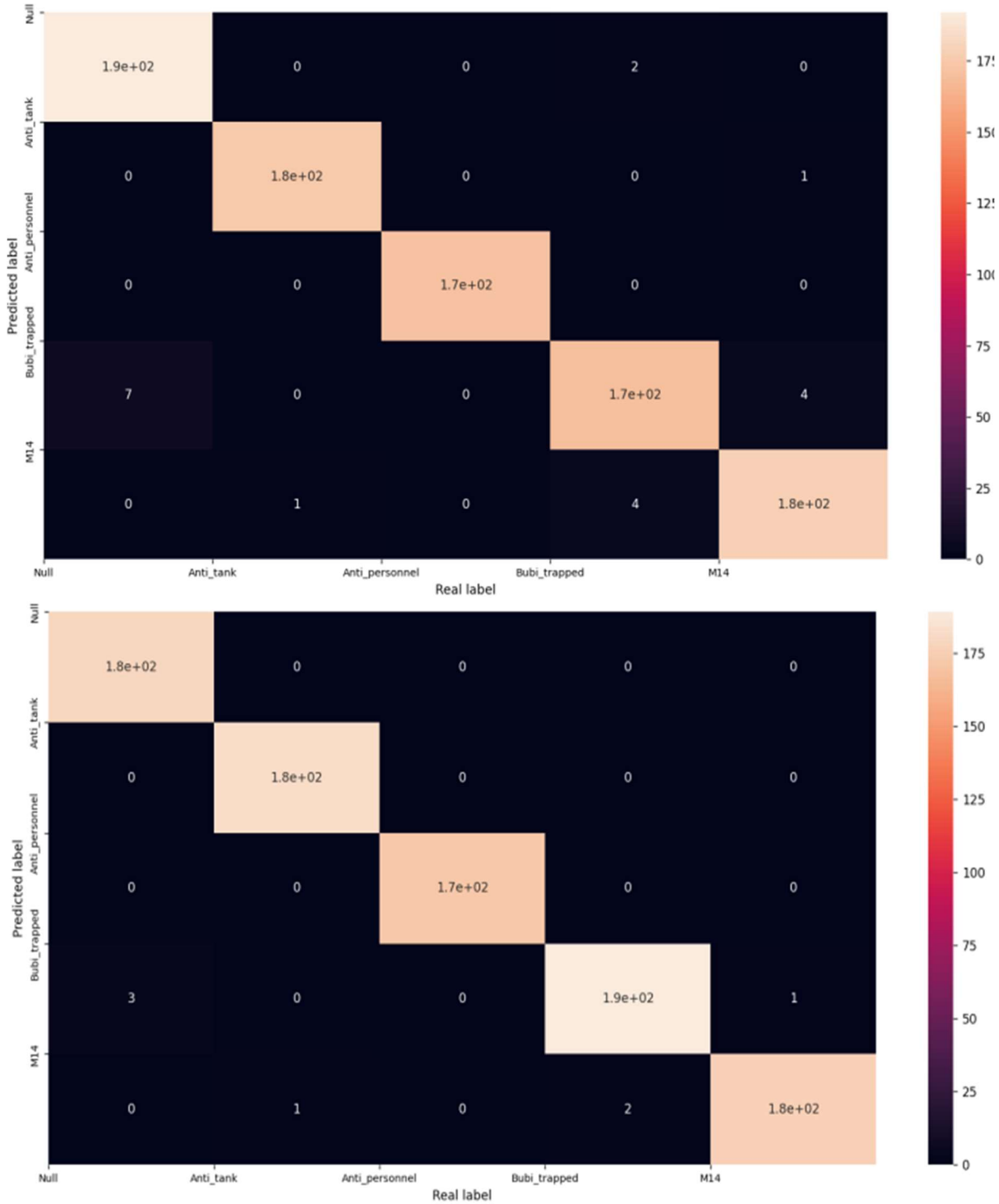

Figure 14: Confusion Matrix, where a – Neural network with one hidden layer, b – Neural network with two hidden layers

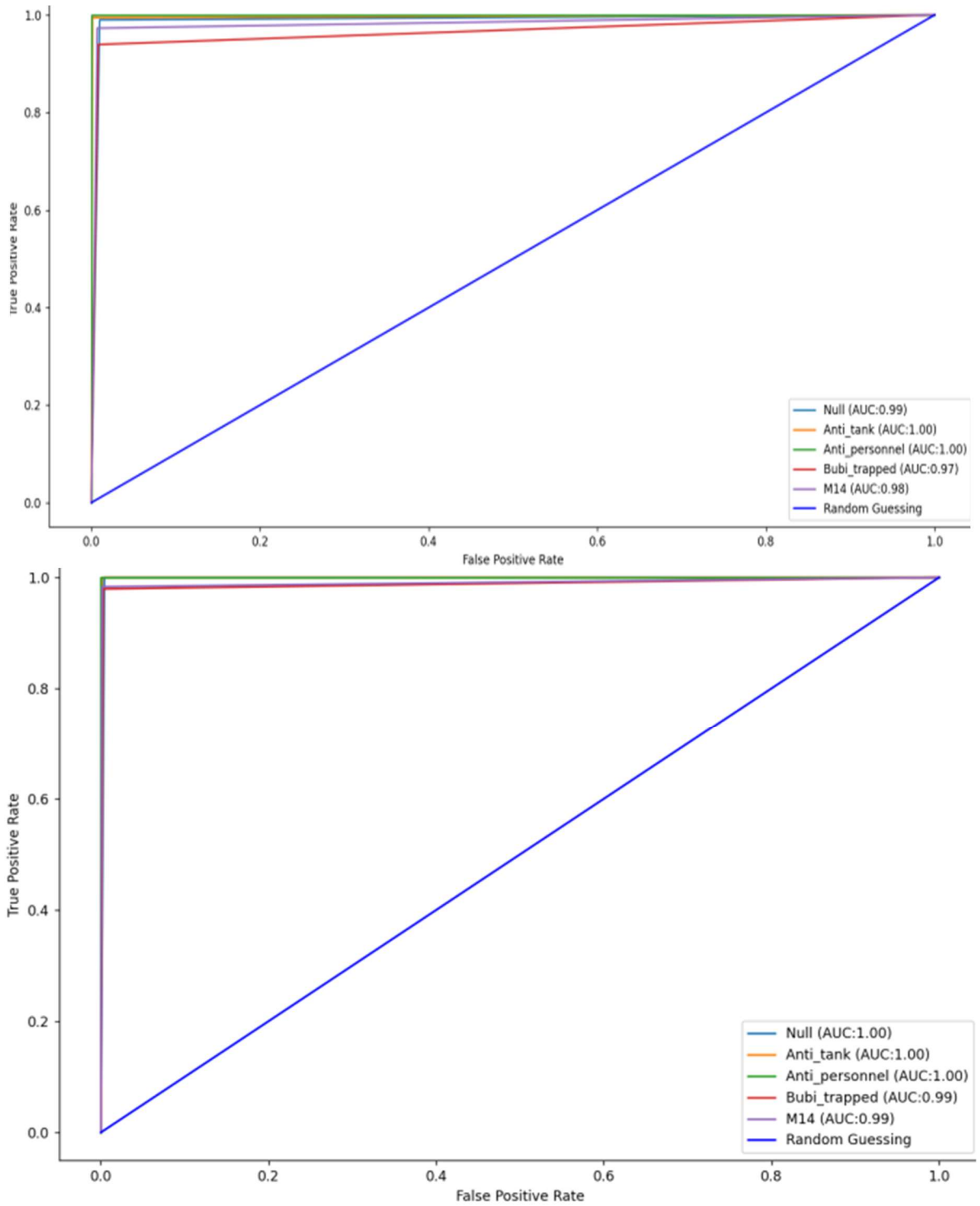

Figure 15: ROC Curves for Each Type of Mine, where a – Neural network with one hidden layer, b – Neural network with two hidden layers.

1. **Confusion Matrix Heatmap** visualises the confusion matrix of the classification results, where the X-axis corresponds to the predicted classes, and the Y-axis represents the actual classes (Figure 14). For clarity in presenting large values, scientific notation (e.g., "e+02") is employed, indicating that the given number should be multiplied by 10^2. The heatmap reveals a high classification accuracy, with most classes correctly classified. More than 180 samples per class were accurately predicted, while the total number of misclassified samples remains below 10, underscoring the model's robust performance [47].

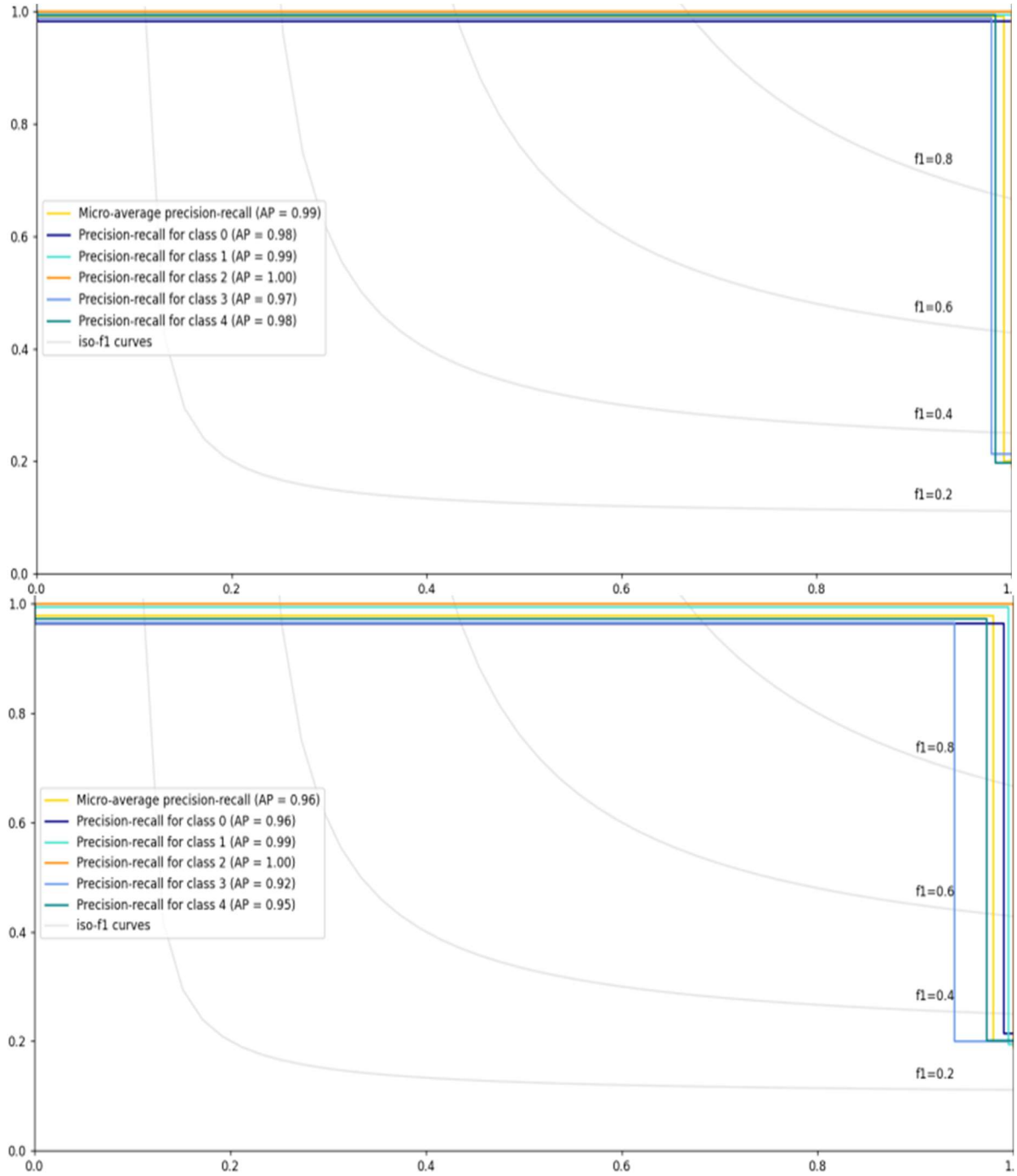

Figure 16: Precision-Recall Curve, where a – Neural network with one hidden layer, b – Neural network with two hidden layers.

2. **The Receiver Operating Characteristic (ROC) curve** provides a detailed assessment of the classification quality across each class. The X-axis tracks the growth in actual positive classifications, while the Y-axis measures the increase in *false positive classifications* (Figure 15) [48]. This visualisation enables a precise evaluation of the model's discriminatory power for individual classes, highlighting its effectiveness in distinguishing between them.
3. **Precision-Recall curves** offer additional insight into the model's classification performance and incremental improvements. The variation in recall is represented on the X-axis, while the Y-axis displays the corresponding changes in Precision (Figure 16). Additionally, the F-score, a harmonic mean of Precision (P) and Recall (R) [49], is depicted, serving as a comprehensive measure of the model's overall performance. The F-score is calculated as follows:

$$F = \frac{2PR}{P+R}. \quad (9)$$

4. The graph includes **iso-F1 curves**, representing lines where the values of precision and recall yield a corresponding F1 score [50]. These curves provide an intuitive understanding of the balance between precision and recall and how this affects the F1 criterion.
5. **Accuracy and Loss Curves** illustrate the accuracy metric and loss function after each training epoch. The values of accuracy and loss are shown on the Y-axis, while the X-axis represents the progression of training epochs (Figure 17) [51].

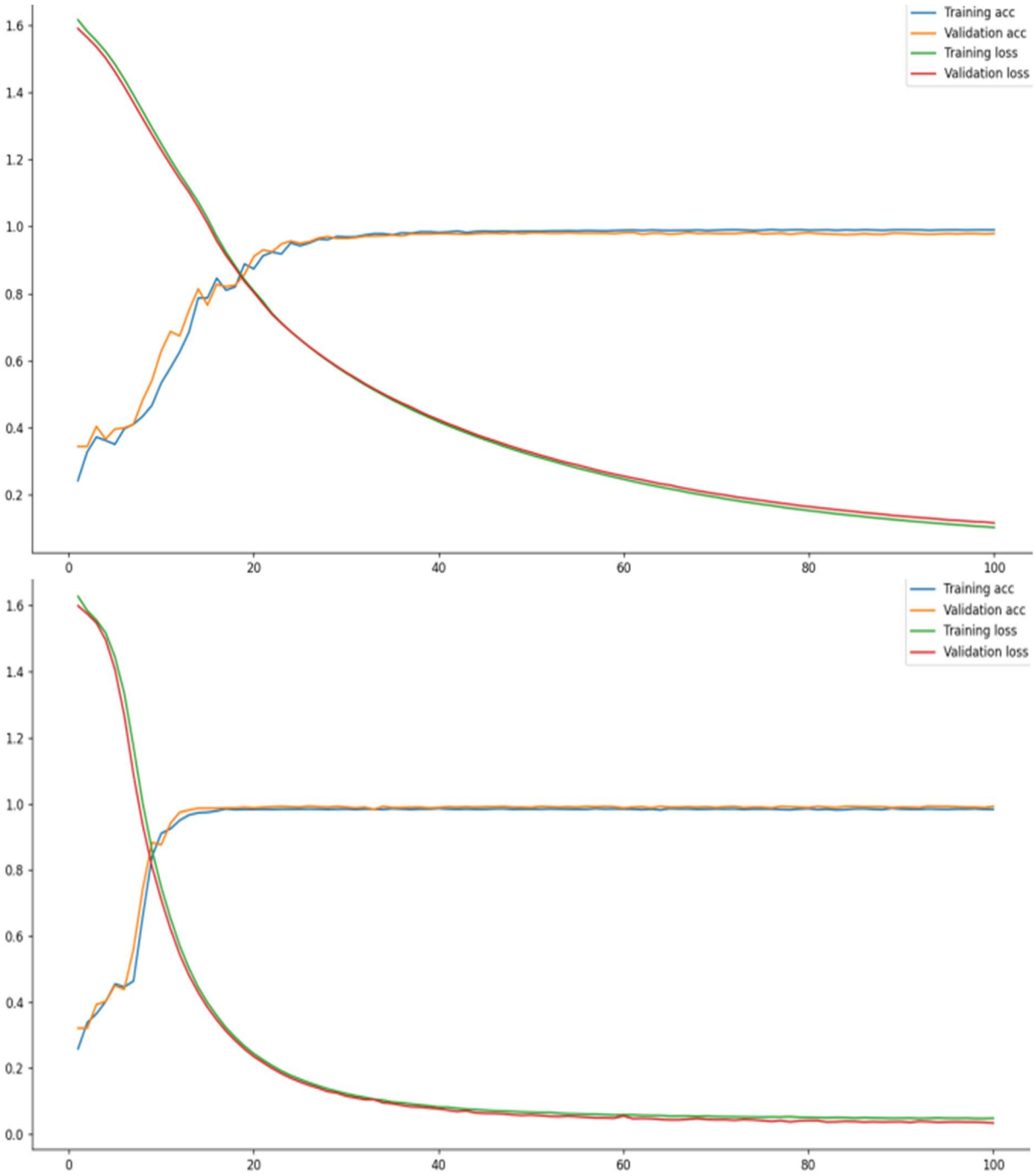

Figure 17: Accuracy-Loss Curves, where a – Neural network with one hidden layer, b – Neural network with two hidden layers.

The ROC curves for the two-hidden-layer model exhibit superior performance compared to a similar neural network with a single hidden layer. The area under the curves (AUC) in Figure 15 is

larger across all mine classes. An AUC value exceeding 0.99 was achieved, demonstrating exceptional classification performance for each mine class. It indicates that the neural network model reliably distinguishes between different types of mines with remarkable precision.

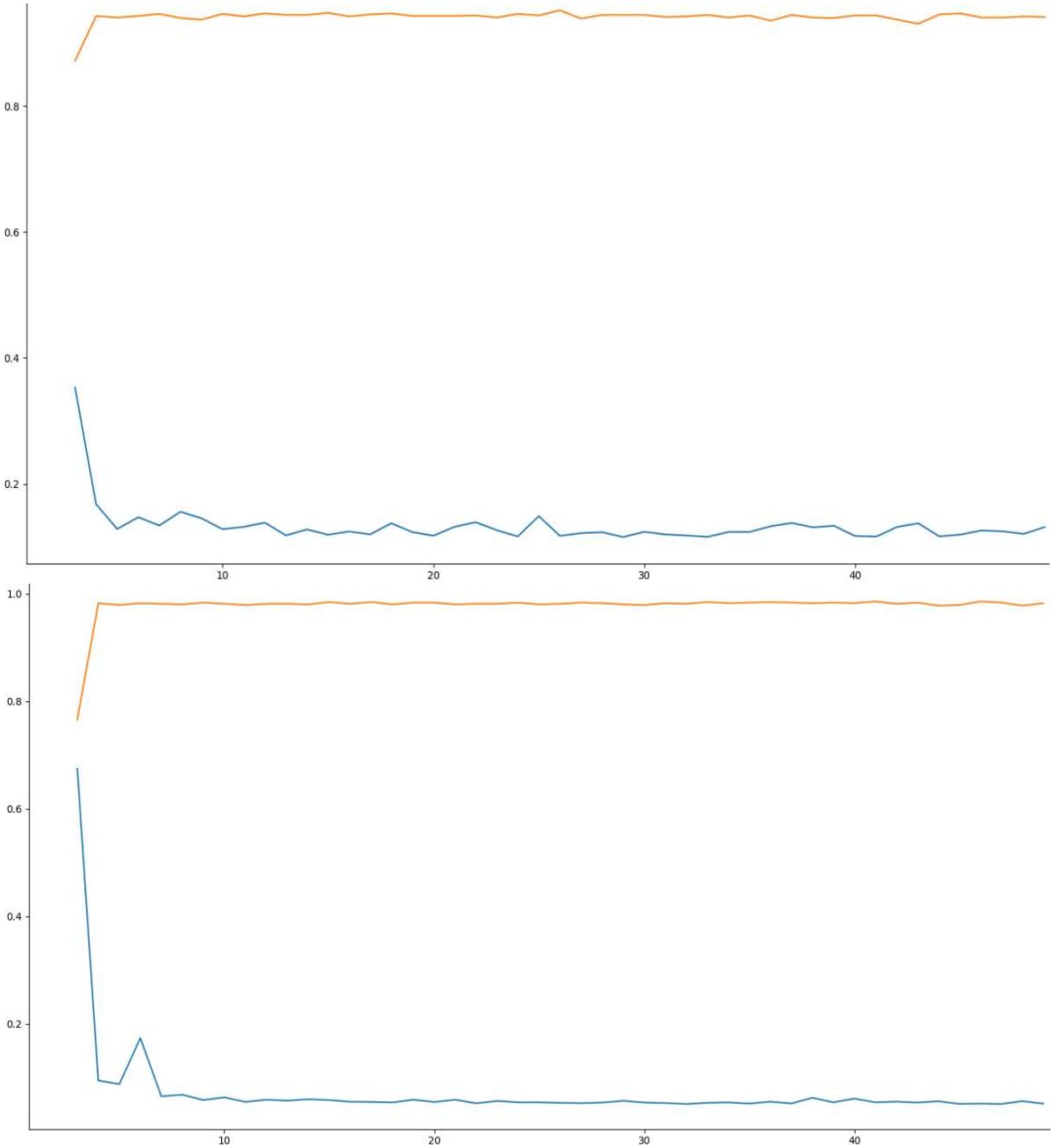

Figure 18: The relationship between accuracy and loss metrics relative to the number of neurons in the first (a) and second (b) hidden layers.

Table 10
Results of Neural Networks by Accuracy Metric Relative to Optimizers

Optimisers	1-layer NN	2-layer NN
Adam	0.9790	0.9923
RMSprop	0.9790	0.9856
SGD	0.9695	0.9812

Table 11
Results of Neural Networks by Accuracy and AUC Score Metrics

Metrics	1-layer NN	2-layer NN
Accuracy	0.9790	0.9923
AUC score	0.9870	0.9953

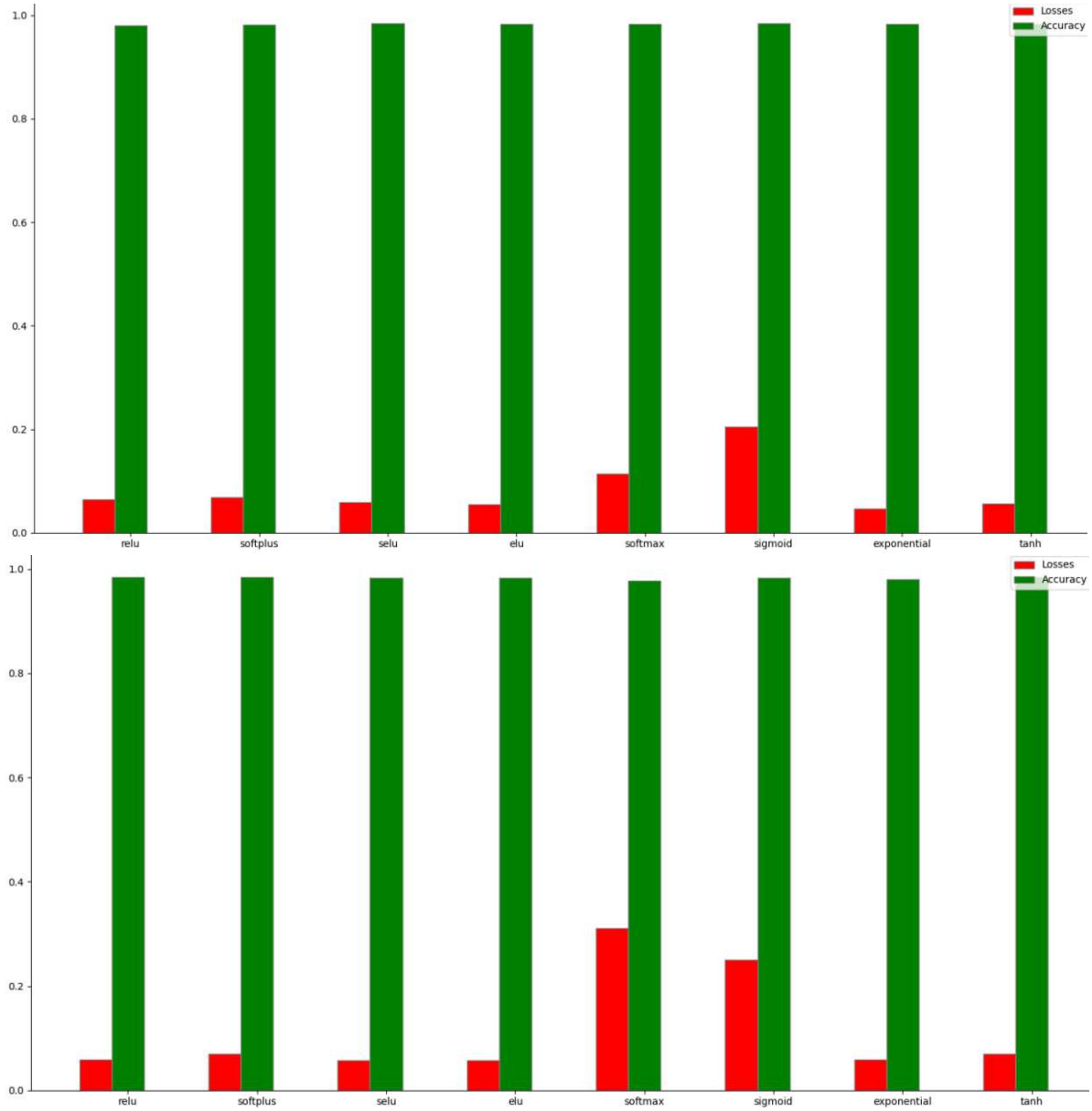

Figure 19: Correlation between accuracy and loss metrics as influenced by the activation function in the first (a) and second (b) hidden layers, each comprising seven neurons

The heatmap demonstrates that the number of misclassified classes is significantly reduced, with over 180 correctly classified samples for each class and fewer than ten misclassifications overall. Training a neural network with two hidden layers accelerates learning, as shown in Figure 16. Consequently, the model is capable of achieving higher accuracy at a faster rate compared to a neural network with only one hidden layer.

The F1-score is consistently high, exceeding 0.8 (Figure 16), which indicates a low number of false negatives and false positives, thus reflecting the model's accuracy in classification.

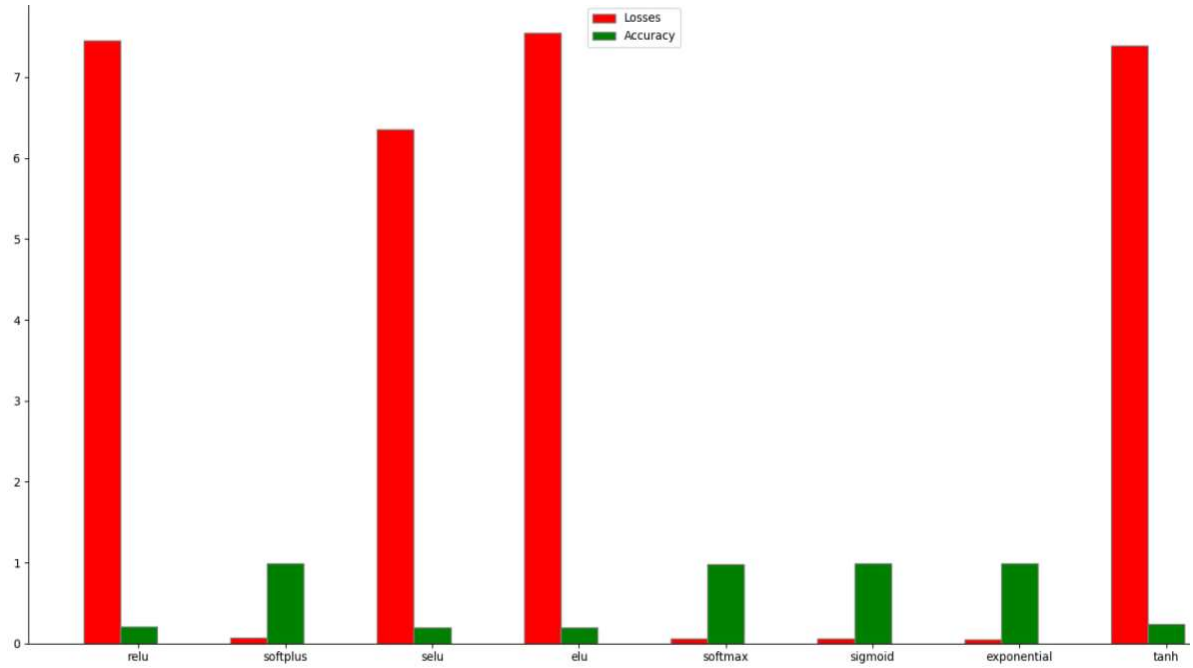

Figure 20: Correlation between accuracy and loss metrics on the activation function in the output layer

The loss values for the neural network are impressively low, falling below 0.1, in contrast to similar studies where loss values peaked at 0.1. It suggests that the neural network training process is highly efficient, resulting in better overall performance and convergence.

Experiments were conducted on neural networks with one and two hidden layers. The networks' effectiveness was evaluated using the metrics of Accuracy and AUC score. The optimisers Adam, RMSprop, and SGD were selected for optimal performance. The assessment of these optimisers was carried out using the Accuracy metric. The results are presented in Tables 10 and 11. To enhance the performance of the neural network, a comprehensive investigation into its symmetry was undertaken. Symmetry within the neural network architecture is defined by an equal distribution of neurons across all layers [52] and the uniformity of activation functions [53] concerning the established symmetry plane. In this study, the plane of symmetry was delineated between the first and second hidden layers.

The following experimental procedures were implemented:

1. Variation of Neuron Count: Systematically altering the number of neurons in the first and second hidden layers, ranging from 3 to 49.
2. Modification of Activation Functions: Adjusting the activation functions employed in the first and second hidden layers.
3. Assessment of Output Layer Performance: Evaluating the performance outcomes of the model upon varying the activation function within the output layer.

Figure 18 elucidates the results, wherein the orange line denotes the accuracy metric, while the blue line represents the loss function. The Y-axis encapsulates the metrics of accuracy and loss, whereas the X-axis indicates the neuron count in both the first and second hidden layers.

The findings indicate that asymmetry between the first and second hidden layers exerts negligible influence on the model's accuracy (≤ 1%); however, a significant escalation of up to 75% in the loss function occurs when symmetry is compromised. Notably, the choice of activation functions within the output layer of the neural network yields profound implications for performance. Specifically, substituting the ReLU function in the output layer with alternatives such as softmax, softplus, sigmoid, or exponential functions results in a remarkable increase in accuracy, rising from 21% to an impressive 98%. In stark contrast, the loss function attains its apex (exceeding 6) when utilising

activation functions such as ReLU, SELU, ELU, and tanh. In contrast, it exhibits minimal values when employing softplus, softmax, sigmoid, and exponential functions.

The observed metrics for accuracy and loss can be elucidated through gradient vanishing and explosion. In the case of the ReLU function, as depicted in Figure 19, both vanishing and exploding gradients are absent, given that the derivative of the ReLU function remains constant for positive input values. Conversely, all other activation functions demonstrate vulnerability to the issues of gradient vanishing and explosion, attributed to their variable derivatives within their respective domains. Figure 20 further exemplifies this inverse relationship.

6.2. System deployment

The system's initial deployment will be orchestrated using Docker software, which will also encompass the database within the container. The following steps elucidate the process of creating and launching the Docker container:

- The creation of the Dockerfile step entails the formulation of a specialised file titled "Dockerfile" located in the application's root directory. This file articulates a comprehensive sequence of instructions for initialising the application and the database within the container. Specifically, it designates the official Python 3.9 image as the base image, establishes the root directory as /app, copies the requirements.txt file into the container, installs the requisite dependencies, transfers the application code to the container, opens port 8000, and initiates the server.
- The creation of the requirements.txt File process involves generating a dedicated file that enumerates all essential packages and libraries, along with their respective versions, which are imperative for the successful execution of the application within the container.
- The Docker Image command from the application's root directory is executed in the terminal.
- Launching the Docker Container e container must be instantiated using a specific command. Upon successful initiation, the container will be visible in the Docker registry, with the back-end and database fully operational [52].

The procedures for testing and deploying the system have been meticulously documented. Each component of the system was subjected to various testing methodologies, with the neural network's performance represented through corresponding metrics and visualisations.

The system deployment was conducted locally utilising Docker software, incorporating both the database and back-end components. For future development endeavours, it is advisable to consider procuring hardware from cloud service providers.

7. Conclusions

This research has presented a sophisticated system for the passive detection and classification of mines utilising deep neural networks. Two models were developed, featuring one and two hidden layers, which achieved accuracies of 97.9% and 99.2%, respectively. Although the neural network with a single hidden layer exhibited slightly lower accuracy than its counterpart with two hidden layers, it was less computationally intensive during the training phase. Remarkably, the classification accuracy attained in this study was 99.2%, surpassing the performance of a similar heuristic algorithm, k-NN, which employed a fuzzy metric by a margin of 1%.

In comparison to the findings presented in reference [6], several notable advancements were achieved, including superior accuracy metrics:

1. Heatmap Analysis results illustrated in the heatmap demonstrated a significantly reduced number of misclassified instances, with correct classifications exceeding 180 for each category. At the same time, the total misclassifications were limited to fewer than 10.

2. The learning process of a neural network with two hidden layers is faster, as shown in Figures 16-18. Thus, the training and accuracy of a model with two hidden layers is faster than that of a neural network with one hidden layer due to the optimised morphology of the neural network with two hidden layers.
3. ROC Curve Performance values were notably higher than those reported in the analogous study involving a single hidden layer. The graphs in Figure 15 exhibited larger areas under the curves across all mine classes, indicating enhanced discriminative capability.
4. An AUC of more than 0.99 was achieved, showing high efficiency in classifying each mine class. The neural network model clearly distinguishes one type of mine from another, taking into account the soil structure.
5. The F1 metric takes a value greater than 0.8, which means a low number of classified false negatives and false positives.
6. The loss rates for the neural network reach a value of less than 0.1, compared to the rates from another study where they peaked at 0.1, which means practical neural network training.

The described approach is not perfect and could be improved. To improve the performance of passive mine detection based on data from magnetic field sensors and their classification using neural networks, the following steps must be taken:

- Increase the dataset based on actual data, considering different types of soil structure.
- Develop optimised models of other neural network architectures.

Acknowledgement

The research was carried out with the grant support of the National Research Fund of Ukraine "Methods and means of active and passive recognition of mines based on deep neural networks", project registration number 273/0024 from 1/08/2024 (2023.04/0024). Also, we would like to thank the reviewers for their precise and concise recommendations that improved the presentation of the results obtained.

References

[1] About Us. International Campaign to Ban Landmines. URL: http://www.icbl.org/en-gb/resources/partner-links.aspx.
[2] McGee-Abe J. One year on: 10 technologies used in the war in Ukraine. Tech Informed. URL: https://techinformed.com/one-year-on-10-technologies-used-in-the-war-in-ukraine/.
[3] In Ukraine, 175,000 square kilometers of territories are mine-hazardous. URL: https://www.ukrinform.ua/rubric-ato/3619278-v-ukraini-175-tisac-kvadratnih-kilometriv-teritorij-e-minno-nebezpecnimi-dsns.html.
[4] L. Collier, Clearing landmines from Ukraine may take decades; Work to find, map, and remove them has already begun. Geneva : GICHD, 2022.
[5] J. Baur, G. Steinberg, A. Nikulin, K. Chiu, Timothy Smet, Applying Deep Learning to Automate UAV-Based Detection of Scatterable Landmines, Remote Sensing 12 (2020) 859. 10.3390/rs12050859.
[6] C. Yilmaz, H. T. Kahraman, S. Söyler, Passive Mine Detection and Classification Method Based on Hybrid Model, IEEE Access 6 (2018) 47870-47888, doi: 10.1109/ACCESS.2018.2866538.
[7] Landmine Detector. URL: https://cepdnaclk.github.io/e17-3yp-Landmine-Detector/#intro.
[8] B. Zhang, S. Tang, M. Jin, C. Xu, M. Tong, Research on Mine Robot Positioning Based on Weighted Centroid Method," in International Conference on Robots & Intelligent System (ICRIS), Changsha, China, 2018, pp. 17-20, doi: 10.1109/ICRIS.2018.00013.
[9] V. Motyka, M. Nasalska, Y. Stepaniak, V. Vysotska, M. Bublyk, Radar Target Recognition Based on Machine Learning, CEUR Workshop Proceedings, Vol-3373 (2023) 117-128.

[10] S. Vladov, R. Yakovliev, V. Vysotska, M. Nazarkevych, V. Lytvyn, The Method of Restoring Lost Information from Sensors Based on Auto-Associative Neural Networks, Applied System Innovation 2024, 7(3), 53. doi:10.3390/asi7030053.
[11] V. Teslyuk, V. Chornenkyi, V. Tsapiv, I. Kazymyra, Investigating Hand Gesture Recognition Models: 2D CNNs vs. Visual Transformers, CEUR Workshop Proceedings 3688 (2024) 29-38.
[12] S. Vladov, V. Vysotska, V. Sokurenko, O. Muzychuk, M. Nazarkevych, V. Lytvyn, Neural network system for predicting anomalous data in applied sensor systems, Applied System Innovation 7(5) (2024). doi: 10.3390/asi7050088.
[13] Y. Kryvenchuk, M. Dmytryshyn, Utilization of Machine Learning in Recognition of Rocks and Mock-mines by Sonar Chirp Signals, CEUR Workshop Proceedings 3688 (2024) 209-218.
[14] A. Vasyliuk, T. Basyuk, V. Lytvyn, Specialised interactive methods for using data on radar application models, CEUR Workshop Proceedings 2631(2020) 1–11.
[15] I. N. Garkusha, V. V. Hnatushenko, V. V. Vasyliev, Research of influence of atmosphere and humidity on the data of radar imaging by Sentinel-1, in IEEE 37th International Conference on Electronics and Nanotechnology (ELNANO), Kiev, 2017, pp. 405-408. doi: 10.1109/ELNANO.2017.7939787.
[16] V. Hnatushenko, I. Garkusha, V. Vasyliev, Creating soil moisture maps based on radar satellite imagery, in Proc. SPIE 10426, Active and Passive Microwave Remote Sensing for Environmental Monitoring, 104260J (3 October 2017); doi: 10.1117/12.2278040.
[17] O. Kavats, V. Hnatushenko, Y. Kibukevych, Y. Kavats, Flood Monitoring Using Multi-temporal Synthetic Aperture Radar Images. Advances in Intelligent Systems and Computing 1080. (2020) Springer, Cham. doi: 10.1007/978-3-030-33695-0_5.
[18] M. Makaruk, A. Nazarov, I. Shubin, N. Shanidze, Knowledge Representation Method for Object Recognition in Nonlinear Radar Systems, CEUR Workshop Proceedings 2870 (2021) 948-958.
[19] L. Mochurad, R. Bliakhar, N. Reverenda, Identification and Tracking of Unmanned Aerial Vehicles Based on Radar Data, CEUR Workshop Proceedings 3426 (2023) 171-181.
[20] A. Mihalyov, V. Hnatushenko, V. Hnatushenko, N. Vladimirska, Optimisation model lifetime wireless sensor network, in IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015, pp. 867-871, doi: 10.1109/IDAACS.2015.7341427.
[21] I. N. Garkusha, V. V. Hnatushenko, Modeling the TOA Brightness Temperature on the SWIR-Sensors, in IEEE International Geoscience and Remote Sensing Symposium, 2018. doi:10.1109/igarss.2018.8519287.
[22] V. Hnatushenko, V. Hnatushenko, Recognition of High Dimensional Multi-Sensor Remote Sensing Data of Various Spatial Resolution, in IEEE Third International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 2020, pp. 262-265, doi: 10.1109/DSMP47368.2020.9204186.
[23] V. Lytvynenko, et al., Optical combustion sensor data interpretation using hybrid negative selection algorithm with artificial immune networks, Mathematical Modeling and Computing, 2(1) (2015) 58–70.
[24] S. Voloshyn, R. Peleshchak, I. Peleshchak, V. Vysotska, Big Data Analysis for Multispectral Images Recognition Based on Deep Learning, in: Proceedings of the IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), 22-25 Sept., Lviv, Ukraine. 2021, Vol. 1, pp. 160–170.
[25] Y. Kondratenko, O. Gerasin, A. Topalov, A simulation model for robot's slip displacement sensors, International Journal of Computing 15(4) (2016) 224-236.
[26] R. Peleshchak, V. Lytvyn, I. Peleshchak, V. Vysotska, Stochastic Pseudo-Spin Neural Network with Tridiagonal Synaptic Connections, in: IEEE International Conference on Smart Information Systems and Technologies (SIST), Nur-Sultan, Kazakhstan, 2021. https://ieeexplore.ieee.org/document/9465998.
[27] A. Sartiukova, R. Peleshchak, I. Peleshchak, V. Vysotska, The Multiclass Classification of Objects Based on Multispectral Images Recognition, in: Proceedings of the IEEE 16th International

Conference on Computer Sciences and Information Technologies (CSIT), 22-25 Sept., Lviv, Ukraine. 2021, Vol. 1, pp. 52–60.
[28] S. Tchynetskyi, R. Peleshchak, I. Peleshchak, V. Vysotska, A Neural Network Development for Multispectral Images Recognition, in: Proceedings of the IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), 22-25 Sept., Lviv, Ukraine. 2021, Vol. 2, pp. 278–284.
[29] R. Peleshchak, V. Lytvyn, N. Kholodna, I. Peleshchak, V. Vysotska, Two-Stage AES Encryption Method Based on Stochastic Error of a Neural Network, in: Proceedings of IEEE 16th International conference on Advanced trends in radioelectronics, telecommunications and computer engineering, TCSET-2022, Lviv-Slavske, Ukraine, Feb. 22 - 26, 2022.
[30] V. Lytvyn, I. Peleshchak, R. Peleshchak, I. Shakleina, N. Mozol, D. Svyshch, Object image recognition using multilayer perceptron combined with Singular value decomposition, CEUR Workshop Proceedings 3711 (2024) 225-234.
[31] V. Lytvyn, R. Peleshchak, I. Rishnyak, B. Kopach, Y. Gal, Detection of Similarity Between Images Based on Contrastive Language-Image Pre-Training Neural Network, CEUR Workshop Proceedings 3664 (2024) 94-104.
[32] I. Peleshchak, D. Koshtura, M. Luchkevych, V. Tymchuk, Classification of Dynamic Objects Using a Multilayer Perceptron, CEUR Workshop Proceedings 3664 (2024) 192-203.
[33] R. Peleshchak, V. Lytvyn, I. Peleshchak, S. Voloshyn, Neural Network Technology of Mine Recognition Based on Data from Magnetic Field Sensors, in Proceedings of the 2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Dortmund, Germany, 7–9 September 2023; pp. 546–549.
[34] R.M. Peleshchak, V.V. Lytvyn, M.A. Nazarkevych, I.R. Peleshchak, H.Y. Nazarkevych, Influence of the Symmetry Neural Network Morphology on the Mine Detection Metric, Symmetry 2024, 16, 485. https://doi.org/10.3390/sym16040485
[35] R. Peleshchak, V. Lytvyn, O. Mediakov, I. Peleshchak, Morphology of Convolutional Neural Network with Diagonalized Pooling, in The International Conference on Modelling and Development of Intelligent Systems (MDIS 2022), Sibiu, Romania, 2022, doi: 10.1007/978-3-031-27034-5_11.
[36] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg, 2006.
[37] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[38] P. Glasserman, The Normal Distribution. Managerial Statistics, 2001.
[39] NumPy community. NumPy User Guide, URL: https://numpy.org/community/.
[40] P. Muhammad Ali, R. Faraj, Data Normalisation and Standardisation: A Technical Report, 2014. 10.13140/RG.2.2.28948.04489.
[41] R. Peleshchak, V. Lytvyn, I. Peleshchak, A. Khudyy, Z. Rybchak, S. Mushasta, Text Tonality Classification Using a Hybrid Convolutional Neural Network with Parallel and Sequential Connections Between Layers, CEUR Workshop Proceedings 3171 (2022) 904-915.
[42] A. Alan Pol, Machine Learning Anomaly Detection Applications to Compact Muon Solenoid Data Quality Monitoring. Artificial Intelligence [cs.AI]. Université Paris-Saclay, 2020.
[43] J. Luna, Python vs R for Data Science: Which Should You Learn?. Datacamp. URL: https://www.datacamp.com/blog/python-vs-r-for-data-science-whats-the-difference.
[44] C. Lortie, A review of R for Data Science: key elements and a critical analysis. 10.7287/peerj.preprints.2873v1.
[45] J. Rogel-Salazar, MATLAB for Data Science and Machine Learning. Domino. URL: https://www.dominodatalab.com/blog/why-choose-matlab-for-data-science-and-machine-learning.
[46] A. Ravikiran, C++ Vs Python: Overview, Uses & Key Differences. Simplilearn. URL: https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-vs-python.

[47] J. Markoulidakis, I. Rallis, I. Georgoulas, G. Kopsiaftis, & A. Doulamis, & N. Doulamis, Multiclass Confusion Matrix Reduction Method and Its Application on Net Promoter Score Classification Problem, Technologies 9 (2021) 81. doi:10.3390/technologies9040081.
[48] M. Majnik, & Z. Bosnic, ROC analysis of classifiers in machine learning: A survey, Intelligent Data Analysis 17 (2013) 531-558. doi:10.3233/IDA-130592.
[49] Y. Sasaki, The truth of the F-measure. URL: https://nicolasshu.com/assets/pdf/Sasaki_2007_The%20Truth%20of%20the%20F-measure.pdf.
[50] F. Pedregosa, et al., Scikit-learn: Machine Learning in Python, JMLR 12 (2011) 2825-2830.
[51] Y. Lu, Food Image Recognition by Using Convolutional Neural Networks (CNNs). URL: https://arxiv.org/abs/1612.00983.
[52] M. Thoma, Analysis and Optimisation of Convolutional Neural Network Architectures. ArXiv, abs/1707.09725.
[53] K. He, X. Zhang, S. Ren, J. Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, in IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1026-1034, doi: 10.1109/ICCV.2015.123.
[54] D. Merkel, Docker: lightweight linux containers for consistent development and deployment, Linux Journal (239) (2014) 2.