



Detection and classification of landmines using machine learning applied to metal detector data

L. Safatly, M. Baydoun, M. Alipour, A. Al-Takach, K. Atab, M. Al-Husseini, A. El-Hajj & H. Ghaziri

To cite this article: L. Safatly, M. Baydoun, M. Alipour, A. Al-Takach, K. Atab, M. Al-Husseini, A. El-Hajj & H. Ghaziri (2021) Detection and classification of landmines using machine learning applied to metal detector data, Journal of Experimental & Theoretical Artificial Intelligence, 33:2, 203-226, DOI: [10.1080/0952813X.2020.1735529](https://doi.org/10.1080/0952813X.2020.1735529)

To link to this article: <https://doi.org/10.1080/0952813X.2020.1735529>



Published online: 02 Mar 2020.



Submit your article to this journal [↗](#)



Article views: 525



View related articles [↗](#)



View Crossmark data [↗](#)



Citing articles: 7 View citing articles [↗](#)

ARTICLE



Detection and classification of landmines using machine learning applied to metal detector data

L. Safatly^a, M. Baydoun^b, M. Alipour^b, A. Al-Takach^b, K. Atab^b, M. Al-Husseini^b, A. El-Hajj^a and H. Ghaziri^b

^aElectrical and Computer Engineering Department, American University of Beirut, Beirut, Lebanon; ^bBeirut Research and Innovation Center, Lebanese Center for Studies and Research, Beirut, Lebanon

ABSTRACT

The current landmine clearance methods mostly rely on the manual use of metal detectors (MDs) and on the deminer's experience in differentiating between the sounds emitted due to the presence of a landmine or of harmless clutter. This process suffers from high false-alarm rates, which renders the demining effort slow and costly. In this paper, we report our attempts in using machine learning for decision making in the demining process. We have created our own database of the MD responses corresponding to landmines and/or clutter. A robotic rail is designed and assembled to accurately measure these responses and build the database. Several machine learning models are then developed using the database with the aim of detecting the presence of landmines and classifying them. It is shown that the classification algorithms lead to accurately discriminating the landmines and distinguishing between different buried objects including mines or other items based on the metal detector delivered data or signature.

ARTICLE HISTORY

Received 25 February 2019
Accepted 23 February 2020

KEYWORDS

Landmine localisation; metal detector; machine learning; classification

Introduction

Landmine pollution is a global problem that is created and worsened by wars and military conflicts. More than eighty countries in the world suffer from the landmine problem, with more than a hundred million buried landmines. In addition to thousands of persons being killed or maimed by landmines every year, the economic and social wellbeing of the population in affected areas is also hindered. The most deployed method for landmine clearance is metal detection (Waschl, 1994). Other detection technologies exist, such as ground penetrating radar (MacDonald, Lockwood, McFee, Altshuler, & Broach, 2003), chemical sensors (Krausa, Massong, Rabenecker, & Ziegler, 2002), biological sensors (Harper & Furton, 2007), and infrared imaging (Baertlein, 2003), to name a few. A concise review of several landmines detection methods is reported in (Mokalled, Al-Husseini, Kabalan, & El-Hajj, 2014).

Despite their widespread use, metal detectors (MDs) suffer from high false alarm (FA) rates since they cannot differentiate between the metal components in a landmine and in a harmless metal clutter. Deminers using MDs usually rely on their personal experience to differentiate between the sounds emitted by the MD when scanning a landmine or a clutter. Usually, they continue to excavate on a very large number of occasions and end up by finding a harmless piece of metal. For each found single landmine, it is estimated that a hundred to a thousand false positives are encountered (Schoolderman & Barrell, 2007). The very high FA rate renders the demining process very slow and

CONTACT L. Safatly ✉ ls58@aub.edu.lb 📧 Electrical and Computer Engineering Department, American University of Beirut, Beirut, Lebanon

expensive. This delays the recovery of the polluting lands and the return to normal life activities around the affected areas.

Several works have attempted to improve the accuracy of MD detection and bring down the FA rates. In (Kaneko, Fukushima, & Endo, 2014), the detection audio signal emitted by the MD is transformed into a visual representation, which is easier to interpret. Studies on classification techniques to extract the buried object features and characteristics have been reported in (Kruuger and Ewald, 2008). Signal detection theory coupled with pattern matching algorithms is discussed in (Collins et al., 2000), whereas matching pursuits dissimilarity measure coupled with the fuzzy clustering algorithm is reported in (Mazhar, Gader, & Wilson, 2009). In (Tantum & Collins, 2001), the target responses using the weighted sum of decaying exponentials are modelled, whereas in (Tran, Abeynayake, & Jain, 2012), the modelling was done using wavelets. Other works have tried to combine the signals obtained from the ground penetrating radar (GPR) and metal detectors or electromagnetic induction (EMI). As an example, a specialised cart was used in (Pinar et al., 2015) to collect GPR and EMI data and multi-kernel approaches were utilised for classification, and their approach relied on using histogram of oriented gradients and local binary features among others. The work in (Bruschini, Gros, Guerne, Pièce, & Carmona, 1998) targeted GPR and metal detector data but without classification.

In this paper, we present a novel method that uses machine learning techniques for the detection and classification of landmines. For this purpose, we designed and assembled a MD signals measurement system, which is a robotic rail installed over a sand container, where inert landmine samples are buried. Hundreds of measurements have been taken to form a database of the MD signatures corresponding to several landmines. A machine learning classifier was next developed using the database to distinguish between the buried objects. Several standard and state of the art machine learning methods were evaluated, including linear based methods, bagging, boosting and neural networks.

The paper is organised as follows. The next section describes the methods adopted in this work including the data acquisition setup, preprocessing, feature extraction and classification. The obtained results are next presented and the main contribution of this work in terms of classification accuracy is assessed. The results are thoroughly discussed in a separate section followed by a conclusive section.

Methods

In this section, we present the methodology including data acquisition, preprocessing, feature extraction, and classification. We also assess and validate the accuracy of the classifier and the machine learning model. Building a rich database of MD responses through measurements was an important factor for reaching a high accuracy.

Data Acquisition Setup

The most challenging part of this work is to build a setup to accurately measure MD responses. The setup consists of a robotic rail holding the metal detector to take in-lab measurements of the MD responses of some landmine inert models. A photo of the setup is shown in [Figure 1](#).

The scanning procedure consists of manually swinging the metal detector following a grid over a sand container containing buried objects. The rail system increments the metal detector position between one scan and another. At each grid point, the metal detector's voltage response is recorded. At the end of the scanning procedure, the MD signature of the corresponding buried objects is acquired. The details of the measurement setup are as follows.

The rail is 1.5 m by 1.5 m, with a height of 1 m. Four stepper motors are mounted on the rail, where two of them operate in parallel to move a transversal bar along the x-direction. An arm is mounted on the transversal bar, which is used to hold the MD. The third stepper motor moves the

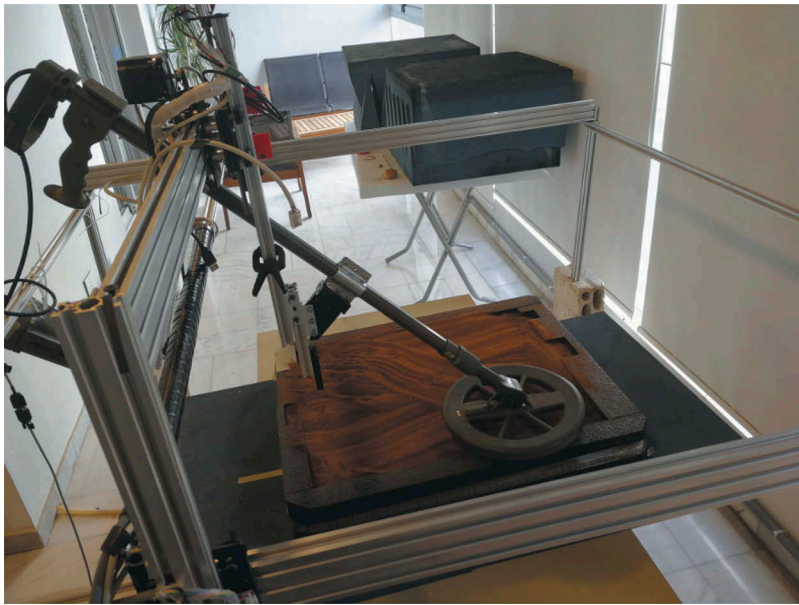


Figure 1. Photo of the robotic rail used to take the MD measurements.

arm, and as a result the MD, along the y-direction. The fourth stepper motor moves the arm in the vertical z-direction that is either up or down. This helps set the height of the MD above the soil. The arm is specially designed to hold the MD at an angle of 45 degrees with respect to the vertical, and that is to limit the interference to the MD from the transversal bar.

In (Kaneko et al., 2014), a robot system is implemented to acquire data. The resolution of this system is 40,000 points per sqm. In (Kruger & Ewald, 2008), an ultrasonic position reference system is embedded on a handheld metal detector to ensure a resolution of 2 mm. In our work a grid (11x10) acquiring data from a 60 cm x 50 cm sand container, which is a resolution of 5 cm or 400 points per sqm. We believe that acquiring fewer data will simplify the complexity of the classification process and thus decrease the response time of the system. The measurements are taken over a grid made of 11 by 10 lines, which gives a total of 110 data points. At each point, the magnitude of the MD voltage response is recorded. This means that a single value is measured at each point and this is the only output of the MD.

Four types of landmine models are tested: M59, AP 72, PMA-2, and TS-50. These are the only types that we were given access to their certified equivalent magnetic models by CEIA (<https://www.ceia.net/>). Two of these certified models are shown in Figure 2.

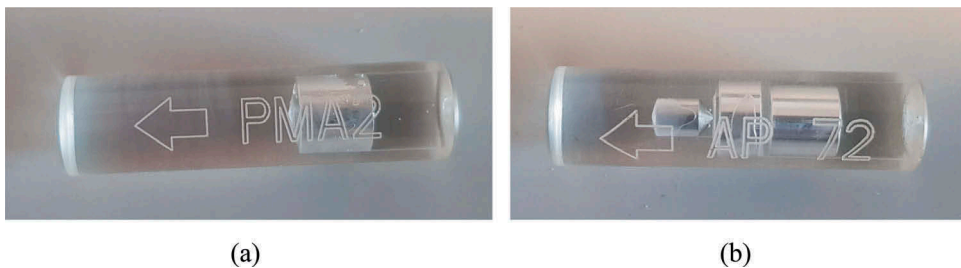


Figure 2. The certified equivalent magnetic models of (a) AP 72, (b) PMA 2.

For each one of the available models, we run the scanning procedure to acquire the corresponding MD signature. A set of MD signatures are taken for each model corresponding to different positions in the container, at different depths, and at different rotation angles. Other measurements are taken by including a clutter at a certain distance from the model. The MD signature of a standalone clutter is also acquired. In total, 201 MD signatures have been recorded with 110 data points each.

Metal Detector Data

Using the robotic rail setup, data are collected from 110 points equally distant and forming a grid of 11 × 10 points. Thus, the result of each measurement is a matrix of MD values. An example of the obtained matrix is shown in Table 1. To further utilise this matrix, it will be considered as an image representing a metal detector signature of the buried objects. Table 1 is shaded to provide a better visualisation of the data.

The MD values of the matrix depend strongly of the metallic component of the buried object. The certified equivalent magnetic model is shown in Figure 2 for two of the four landmines type studied in this paper. In addition, three non-mine objects were also tested to obtain various responses that could lead to more significant results. The non-mine objects were selected as common objects that could be buried in the ground after wars and battles. These objects are a pepsi can, a glass object, and a metallic object as shown in Figure 3.

The obtained database includes 201 measurements corresponding to the MD matrix resulted from various buried objects in different positions, angles, and depths. Table 2 shows the number of measurements performed for each buried object. Each sample measurement had at least 2 similar measurements since each measurement was repeated twice making a total of 3.

Table 1. Example of metal detector data.

6	4	5	6	0	0	6	8	5	8
0	8	0	4	2	4	7	4	0	17
2	4	1	10	7	2	7	0	4	3
2	15	4	7	4	5	1	6	0	8
1	0	6	0	6	1	4	4	2	1
0	7	1	1	12	13	8	6	1	7
2	2	8	0	2	2	4	10	0	2
0	1	8	1	23	22	12	4	8	1
6	0	1	24	1789	3167	1217	11	1	0
1	2	10	3234	11,114	13,719	10,153	1640	4	5
2	2	459	9171	14,703	15,671	14,981	6258	60	17

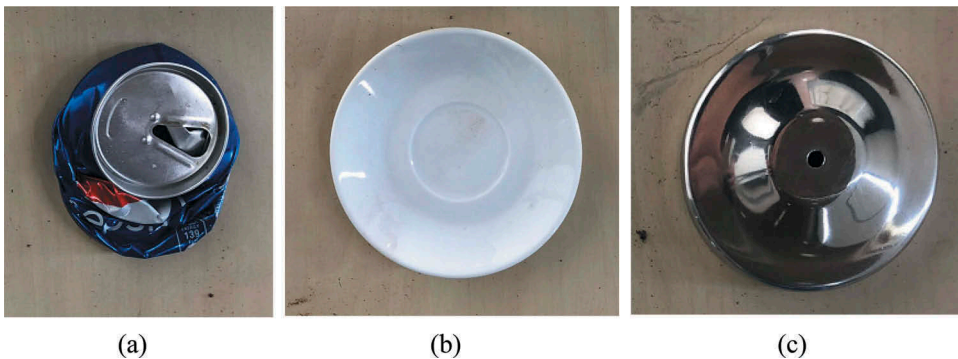


Figure 3. (a) Pepsi can, (b) Glass object, (c) Metallic object.

Table 2. Number of measurements for tested objects.

Object	Number of Measurements
Mine AP 72	60
Mine M59	15
Mine PMA 2	30
Mine TS 50	48
Soil Only	12
Pepsi Can	12
Glass Object	12
Metallic Object	12
Total Measurements	201

To further enrich the database, and since each measurement is an image-like two-dimensional array signature, it is possible to manipulate the data by flipping the matrices. The idea of flipping the two-dimensional arrays corresponds to placing the object in the opposite side of the container. In this work, three manipulations are considered: left and right flipping, up and down flipping, and flipping in both directions. Consequently, a total of 4×201 or 804 data measurements will be obtained without repeating each measurement four times by easily flipping each resulted measurement. This approach is commonly adopted in deep learning algorithms and we will extend it to all classification algorithms.

Preprocessing and Features Extraction

In this section, we describe the methods used for data preprocessing and feature extraction which is a critical step to prepare the database for the classifier leading to more accurate decisions.

Preprocessing

As previously mentioned, each measurement is a matrix of values corresponding to the metal detector responses. It was observed that these values range between 0 and 20,000. Also, the range of values in one matrix varies from one measurement to another. On the other hand, the convolutional neural networks require images as input, where each image pixel value range from 0 to 255 corresponding to 1 byte even though a floating value is used. Thus, to create an image of the metal detector matrix data, several methods could be applied. The first idea is to normalise every image by dividing over the maximum value of the image. This method preserves relative information and does not account for difference between varying images. Another way is to divide by the maximum value of all the images, approximately 20,000, which will result in losing some information due to low values such as 10 ($10/20000 * 255$ will be considered ~ 0).

While it is possible to utilise a single channel or grey-like image, there is a clear loss of information when using the values directly. The loss is caused by the wide initial interval as opposed to a byte value. To solve this issue, a three-channel image will be created for each array of data. The first channel is the normalised image where the image is divided by the local maximum. The second channel has the value at each data point divided by 20,000 or the global maximum. The 20,000 value is chosen because it is very close to the global maximum seen in all the images noting that it is possible to choose similar neighbouring or greater values but this seemed to provide the best performance. The third channel is computed by applying the logarithm function to all values of the image as in equation 1.

$$x_{\text{new}} = \log(1 + x) / \log(20000) \quad (1)$$

The reason behind using the log in the third channel is that large values become relatively comparable with smaller values with this approach. For example,

$$\log(2) = 0.69 \quad (2)$$

$$\log(20000) = 9.9 \tag{3}$$

This means that all the values are relatively close and yet still maintain a difference. In addition to this, using the log based third channel has proven to improve the accuracy. Definitely, all the values in all the channels are normalised between 0 and 1.

This preprocessing approach was only utilised for the convolutional neural network classifier where no feature extraction was performed. For the other classifiers, feature extraction is directly applied on the measured matrices without accounting for the range. Figure 4 shows an example of the obtained metal detector data images with and without preprocessing.

Feature Extraction

Feature Extraction is a crucial stage in building an accurate machine learning model. All classification algorithms, except deep learning-based methods, benefit from extracted features for better accuracy. In this work, horizontal and vertical derivative images are computed by taking the difference between each two adjacent rows and columns and these yield two images with a dimension of 10×10 and 11×9 respectively. Examples of the extracted images are shown in Figure 5.

After extracting the derivative image, the features are computed from each MD measurement based on the signature matrix and the derivative maps. The extracted features that are shown in Table 3 are commonly used features in machine learning since they usually provide a significant

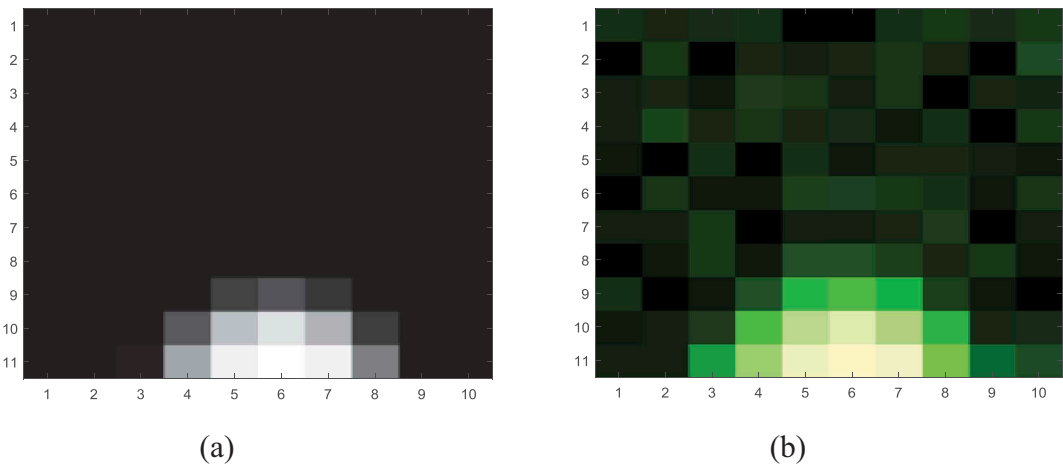


Figure 4. (a) Example of metal detector data (Original Data), (b) Corresponding preprocessed image.

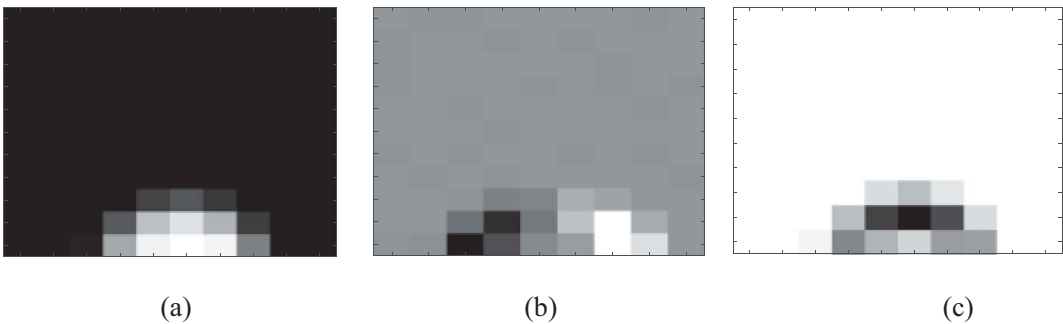


Figure 5. Example of (a) Metal detector data, (b) Metal detector vertical derivative (11x9), (c) Metal detector horizontal derivative (10x10).

Table 3. Extracted features from the metal detector data.

Feature Name	Number of Extracted Features
Mean	1
Standard Deviation	1
Third order Moment	1
Entropy	1
Maximum	1
Mean of each column	10
Mean of each row	11
Standard Deviation of each column	10
Standard Deviation of each Row	11
Minimum Mean along the rows	1
Maximum mean along the rows	1
Minimum Mean along the columns	1
Maximum mean along the columns	1
Minimum Standard Deviation along the rows	1
Maximum Standard Deviation along the rows	1
Minimum Standard Deviation along the columns	1
Maximum Standard Deviation along the columns	1
Image Histogram (8 bins)	8
Total Number of Features	63

improvement in the classifier’s accuracy (Strand & Taxt, 1994). These features do not require a preprocessing stage except for the image histogram features which require normalising the data by dividing over the maximum value. Features are extracted from the original and the derivative matrices resulting in 63 features from the original matrix and 61 from each derivative matrix because of their lower dimension. Thus, the total number of features is 185 features for each metal detector measurement.

Classification

Discriminating mines from clutters and identifying the type of landmines can be considered as a multi-class classification problem. For the identification of the type of landmines, the learning goal is to consider each mine type or object as a separate category and aim to identify the specific mine or object. For discrimination, another learning goal is to only consider three categories: mine, non-metallic, or metallic. Thus, two classification problems are considered: the identification problem with 8 classes, and the discrimination problem with 3 classes. The class labelling of the available samples can be considered according to the manner shown in Table 4 which shows the identity of each class either between 1–8 or 1–3.

In this work, several classification algorithms were tested to search for the best machine learning model. Linear Discriminant Analysis (LDA), Quadratic Support Vector Machines (Q-SVM), Bagging, Boosting, and Convolutional Neural Networks are commonly used for image data and were investigated in this work. We present hereafter a very brief explanation of these techniques.

Table 4. Class labelling of the tested objects.

Object	Class ID (8-class)	Class ID (3-class)
Mine AP 72	1	1
Mine M59	2	1
Mine PMA 2	3	1
Mine TS 50	4	1
Soil Only	5	2
Pepsi Can	6	3
Glass Object	7	2
Metallic Object	8	3

Linear Discriminant Analysis

LDA is one of the simpler and common classification techniques that performs classification through linearly combining the features based on the means and the covariances of the data. Since logistic regression does not work with multiclass problems, LDA is commonly used in such cases.

Quadratic Support Vector Machines

Support Vector Machines (SVM) is one of the most common classification algorithms. It achieves classification by creating hyperplanes that separate between the data. SVM aims to maximise the margins around the hyperplanes by utilising support vectors. SVM is initially linear since the hyperplanes are linear, but can be made nonlinear through kernelization or the kernel trick. In the case of quadratic SVM, the kernel is the polynomial kernel with a second degree and can be defined according to equation 4:

$$k(x, y) = (x^T y + c)^2 \quad (4)$$

Where x and y are the vectors of features of a certain sample and $c \geq 0$ is a free parameter trading off the influence of higher-order versus lower-order terms in the polynomial.

Bagging

Bagging, which is originally called Bootstrap aggregating as introduced by Breiman in 1996, is a common ensemble method that combines the outputs from various classification algorithms to yield a better prediction in comparison with a single algorithm. Bagging can be applied to any classifier including decision trees as adopted in this work. Bagging aims at avoiding over-fitting and reducing the variance of a certain classifier by generating a number of training sets (T) of similar size through sampling from the original patterns with replacement. Each one of the (T) sets then yields a different classifier, also called learner in ensemble methods. Afterwards, the output of the bagging model is obtained by voting among the different learners. Bagging can be used for binary, multi-class and regression problems.

It is important to mention here that bagging can be considered as a special case of Random Forests which is another popular tree based algorithm for classification where the subset of features for each new tree is the same and consists of all the original features. In this work, the bagging technique was used for multi- class classification.

Boosting

Boosting algorithms exist in many versions including AdaboostM1 (Freund & Schapire, 1997), AdaBoostM2 (Friedman, Hastie, & Tibshirani, 2001), Least Squares Boost (Hastie, Tibshirani, Friedman, & Franklin, 2005), XgBoost (Chen & Guestrin, 2016), and others (Hristakeva, 2008). Several versions of boosting were tested on the metal detector data, and then AdaboostM2 and XgBoost were selected because they provided an example of utilising both standard and state of the art boosting techniques.

AdaboostM2 is one of the earliest methods and the most commonly used multi-class boosting algorithm. It is a generalisation of AdaboostM1 for binary classification. This algorithm is an ensemble method that requires a number of learners (T), where the used learner is commonly a single leaf decision tree or stump. The algorithm starts with the learner having equal weights for all samples. After training, the error rate or pseudo loss $e(i)$ of the current (i)th learner where (i) belongs to the interval $(0..(T))$ is obtained. After each step, the weight of each training sample is modified in a way that the false classified samples maintain their weights while the true found samples weights have a reduced weighting. All the weights are then normalised, and this weight modification scheme leads to a new learner through training. This process is repeated for a preset number of weak learners, which leads to the creation of a combined strong learner that determines the class of an input pattern by weighted voting for all the weak learners.

In addition, this work utilises the extreme gradient boosting (XgBoost) method proposed by Chen et al. in 2016 that achieved high accuracies (see next section). The main idea in gradient boosting is that new added models, such as decision trees, are created based on predicting the residual errors of prior models while using the gradient descent algorithm to minimise the required loss as opposed to modifying the sample weights as in Adaboost.

Deep Convolutional Neural Networks

Deep learning (deep structured learning, hierarchical learning or deep machine learning) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers composed of several non-linear transformations as opposed to few layers. Deep Learning Neural Networks are considered as the most common and powerful algorithm for machine learning (Shin et al., 2016). In this work, we utilise convolutional neural networks as a form of deep learning since we are dealing with two-dimensional arrays which can be considered as relatively small images.

Convolutional neural networks (CNN) are the most common form of image based deep learning and has been very successful in this field especially that a CNN is automatically capable of detecting underlying features which eliminates the need for a feature extraction stage in contrast with other classification algorithms. A CNN mainly uses a varying set of hidden layers where each layer is composed of a set of neurons. The layers either perform the convolution or the pooling operations. One main advantage of CNN is being invariant to translation which makes it more suitable to this work in particular and images in general.

The main reason behind using a CNN in this work is that these networks provide state of the art results amongst classification algorithms when handling image data, so it is very important to test them since each data sample can be considered an image. The utilised CNN consists of two main parts. The first part is mainly composed of 6 convolutional layers where each two are followed by a max pooling layer and a dropout. Each convolutional layer utilises a rectified linear unit activation function (Relu). The number of convolutional kernels for the first 2 layers was set to 16 and was doubled after each max pooling layer. All the convolutional kernels have a size of 3×3 . At the end of the first part, a global max pooling layer is used to obtain the features. The second part of the CNN has only two convolutional layers followed by dropout and global max pooling to get the features. However, the size of the convolutional kernels was set to 7×7 as opposed to 3×3 in the first part and a total of 128 kernel were used. The output of each part is concatenated and two dense layers with a Relu activation of size 128 and 32 respectively are utilised before the output which can be 8 or 3 depending on the classification problem. [Figure 6](#) shows the architecture of the utilised network. Furthermore, the Adam optimiser is used to train the network with a learning rate of 0.01 noting that this utilised value of this hyperparameter is the best tested one between 0.0001 to 1.

In the next section, we assess the performance of the various machine learning models that were tested. We also highlight the importance of using feature extraction with ensemble-based techniques.

Results

In this section, we present the results obtained when applying different classification algorithms including Adaboost, Bagging, Xgboost, and CNN. First, it is important to note that 3-fold cross validation was used to validate the accuracy of the different algorithms. The 3-fold choice was dictated by the data setup and experiments since each available sample had 2 similar measurements as previously mentioned in the Metal Detector Data section. Using a higher number of folds such as 5-fold or 10-fold which is more common in classification problems would yield a possibility of having a missing kind of a certain sample which would render the results inaccurate whether that achieves lower or higher accuracy.

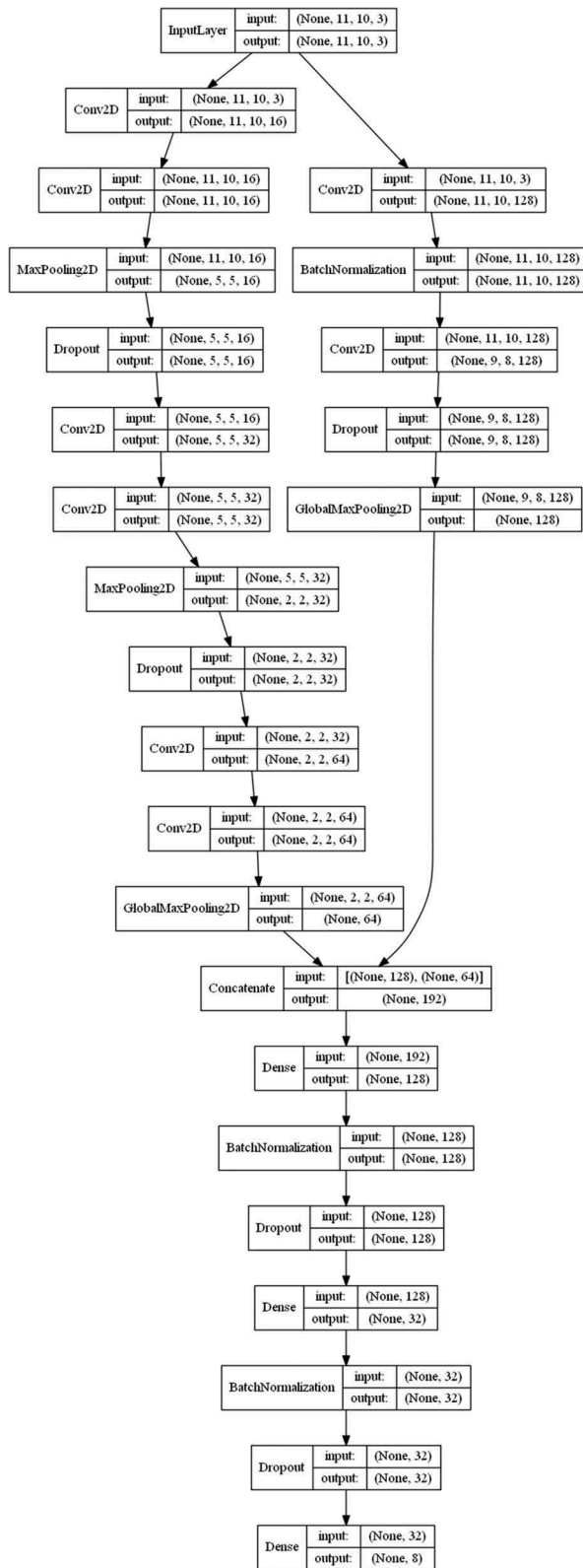


Figure 6. Utilised CNN architecture.

Each fold should have equal instances or samples of the same object to ensure that the results are not biased or that a training instance is similar to a testing one. Thus, $201/3 = 67$ data matrices are available in each fold. Then, each fold is augmented using the matrix flipping procedure described in the Metal Detector Data subsection. Consequently, each fold consists of $67 * 4 = 268$ samples. For each iteration, 2 folds are used for the training process and the remaining group is used for testing. This process is repeated 3 times and the reported accuracy of prediction is the average accuracy over all iterations. If the average accuracy is high enough with a low standard deviation, then the model is accepted with high confidence.

The measurement error metric can vary according to the required learning objective. Since we have a multi-class classification problem with emphasis on mine detection a suitable error metric should be adopted. Confusion matrix is very common in multi-class problems and it is adopted in this work. The confusion matrix provides a comparison between the output classes and the correct ones. It can be used in any classification problem whether binary or multi-class. The matrix is composed of columns and rows where the columns correspond to the target or correct class and the rows correspond to the prediction, noting that it can be shown in a transposed manner. A best-case scenario would be when all the elements of the matrix except for the diagonal ones are 0 since other elements correspond to errors where the predicted class is different from the target one. In addition to the confusion matrix, the receiver operating characteristic curve, or (ROC) curve are shown along with the area under the curve (AUC). Here, the ROC curves and AUC are found for each class on its own by considering it as the positive class and all the other classes combined as the negative class.

The results are presented for the two classification problems as explained in the methods section. The first problem is to identify a specific landmine type resulting in 8 categories. The confusion matrix obtained for 8 classes with LDA, Quadratic SVM, Bagging, AdaboostM2, Xgboost, and CNN are shown in [Figures 7, 9, 11, 13, 15 and 17](#) respectively. In addition, [Figures 8, 12, 14, 16, and 18](#) show the ROC curves and the AUC for the 8 classes. All the results shown here are for a performance of a typical one fold out of the utilised three folds. Besides, it is worth mentioning that Random Forests were tested with results similar to the bagging algorithm.

It is important to explain the obtained results and provide the related information about each used algorithm. [Figures 7 and 8](#) for the LDA algorithm show that the algorithm confuses landmine AP-72 with landmine PMA-2 and the other way around. Also, landmine TS-50 is confused with landmine AP-72. The main error lies in confusing soil with glass and the other way around which is logical due to the nature of the glass material. LDA was used with Matlab with Gamma set to 0 and a "pseudoLinear" discrimination function. The main command used in Matlab is according to the following:

```
templateSVM('KernelFunction', 'polynomial','PolynomialOrder', 2,'KernelScale', 'auto','BoxConstraint', 1,'Standardise', true);
```

[Figures 9 and 10](#) for the Q-SVM algorithm show that the algorithm sometimes confuses landmine AP-72 both PMA-2 and TS-50. Also, M-59 is sometimes confused with TS-50. Moreover, PMA-2 is sometimes confused with AP-72. The highest error lies in confusing soil with glass and the other way around. In this work, Q-SVM was used with Matlab with default parameters after standardising the data. The main command used in Matlab after noting that X and Y are the inputs and outputs respectively is according to the following:

```
fitcdiscr(X,Y,'DiscrimType', 'pseudoLinear','Gamma', 0,'FillCoeffs', 'off');
```

[Figures 11 and 12](#) for the Bagging method show that the approach correctly classifies all the landmines. The only error lies in sometimes confusing soil with glass and the other way around. This indicates that bagging provides very accurate results. In this work, bagging was tested in Matlab with standard parameters such as a classification tree as the main learner and with 100 classification trees. Bagging is preformed according to the following command:

Confusion Matrix for LDA

Output Class	1	76 28.4%	0 0.0%	4 1.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	95.0% 5.0%
	2	0 0.0%	20 7.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
	3	8 3.0%	0 0.0%	20 7.5%	12 4.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	50.0% 50.0%
	4	0 0.0%	0 0.0%	0 0.0%	64 23.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
	5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	12 4.5%	0 0.0%	4 1.5%	0 0.0%	75.0% 25.0%
	6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	0 0.0%	0 0.0%	100% 0.0%
	7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 1.5%	0 0.0%	12 4.5%	0 0.0%	75.0% 25.0%
	8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 1.5%	0 0.0%	0 0.0%	12 4.5%	75.0% 25.0%
			90.5% 9.5%	100% 0.0%	83.3% 16.7%	84.2% 15.8%	60.0% 40.0%	100% 0.0%	75.0% 25.0%	100% 0.0%
		AP 72	M59	PMA 2	TS-50	soil	pepsi	glass	metal	
		Target Class								

Figure 7. Confusion matrix for LDA.

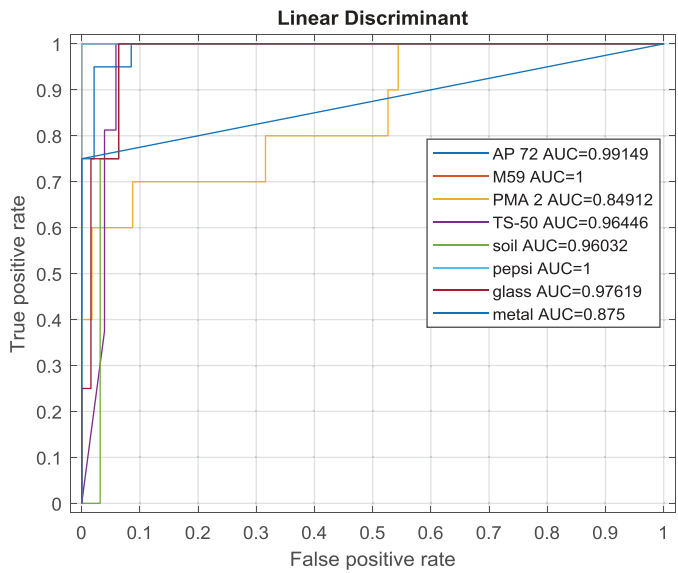


Figure 8. ROC and AUC results for LDA.

Confusion Matrix for Quadratic SVM

Output Class	1	76 28.4%	0 0.0%	4 1.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	95.0% 5.0%
	2	0 0.0%	20 7.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
	3	4 1.5%	0 0.0%	36 13.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	90.0% 10.0%
	4	4 1.5%	2 0.7%	0 0.0%	58 21.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	90.6% 9.4%
	5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	14 5.2%	0 0.0%	2 0.7%	0 0.0%	87.5% 12.5%
	6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	0 0.0%	0 0.0%	100% 0.0%
	7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	6 2.2%	0 0.0%	10 3.7%	0 0.0%	62.5% 37.5%
	8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	100% 0.0%
			90.5% 9.5%	90.9% 9.1%	90.0% 10.0%	100% 0.0%	70.0% 30.0%	100% 0.0%	83.3% 16.7%	100% 0.0%
		AP 72	M59	PMA 2	TS-50	soil	pepsi	glass	metal	
		Target Class								

Figure 9. Confusion matrix for Quadratic SVM.

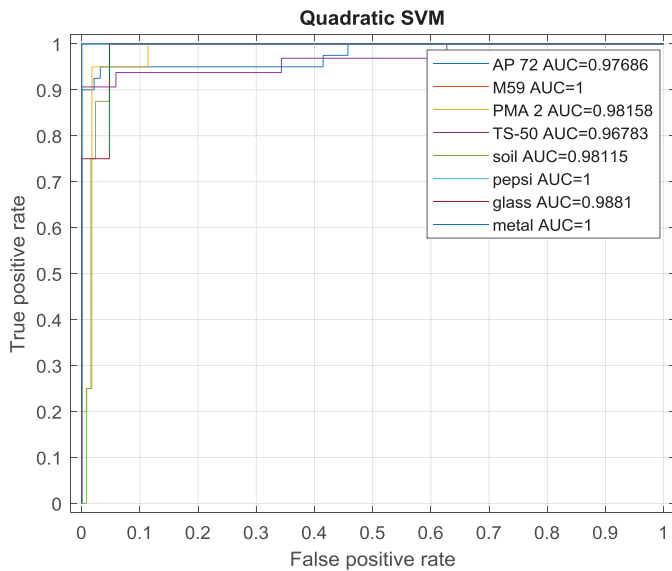


Figure 10. ROC and AUC results for Quadratic SVM.

Confusion Matrix for Bagging

Output Class	AP 72	M59	PMA 2	TS-50	soil	pepsi	glass	metal	
1	80 29.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	20 7.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	40 14.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	64 23.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	13 4.9%	0 0.0%	3 1.1%	0 0.0%	81.3% 18.8%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	6 2.2%	0 0.0%	10 3.7%	0 0.0%	62.5% 37.5%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	68.4% 31.6%	100% 0.0%	76.9% 23.1%	100% 0.0%	96.6% 3.4%

Figure 11. Confusion matrix for bagging.

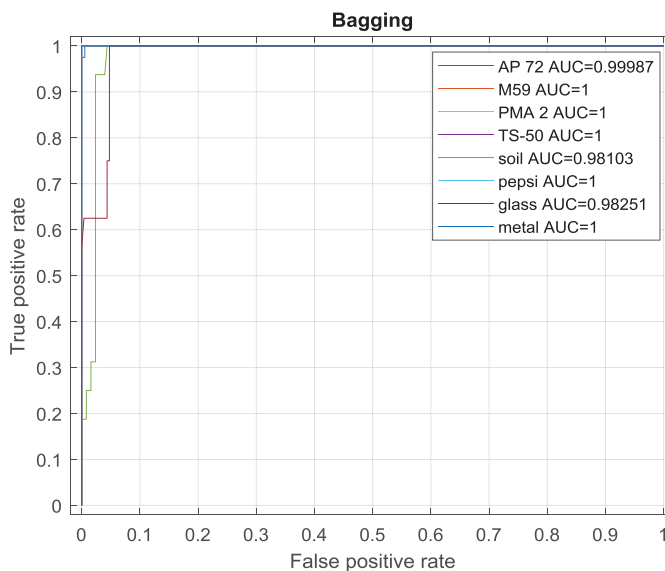


Figure 12. ROC and AUC results for bagging.

Confusion Matrix for Adaboost

Output Class	AP 72	M59	PMA 2	TS-50	soil	pepsi	glass	metal	Accuracy
1	80 29.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	20 7.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3	0 0.0%	4 1.5%	32 11.9%	4 1.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	80.0% 20.0%
4	0 0.0%	0 0.0%	0 0.0%	64 23.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	10 3.7%	0 0.0%	6 2.2%	0 0.0%	37.5% 62.5%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	100% 0.0%
	100% 0.0%	83.3% 16.7%	100% 0.0%	94.1% 5.9%	61.5% 38.5%	100% 0.0%	100% 0.0%	100% 0.0%	93.3% 6.7%
	AP 72	M59	PMA 2	TS-50	soil	pepsi	glass	metal	
	Target Class								

Figure 13. Confusion matrix for adaboost.

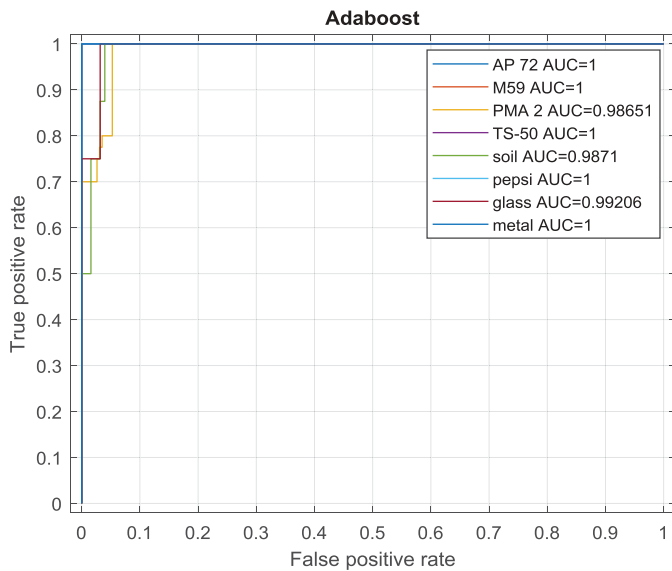


Figure 14. ROC and AUC results for adaboost.

Confusion Matrix for XgBoost

Output Class	AP 72	M59	PMA 2	TS-50	soil	pepsi	glass	metal	Accuracy
1	80 29.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	20 7.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3	2 0.7%	0 0.0%	38 14.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	95.0% 5.0%
4	0 0.0%	0 0.0%	0 0.0%	64 23.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	8 3.0%	0 0.0%	8 3.0%	0 0.0%	50.0% 50.0%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	5 1.9%	0 0.0%	11 4.1%	0 0.0%	68.8% 31.3%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	100% 0.0%
	97.6% 2.4%	100% 0.0%	100% 0.0%	100% 0.0%	61.5% 38.5%	100% 0.0%	57.9% 42.1%	100% 0.0%	94.4% 5.6%

Figure 15. Confusion matrix for XgBoost.

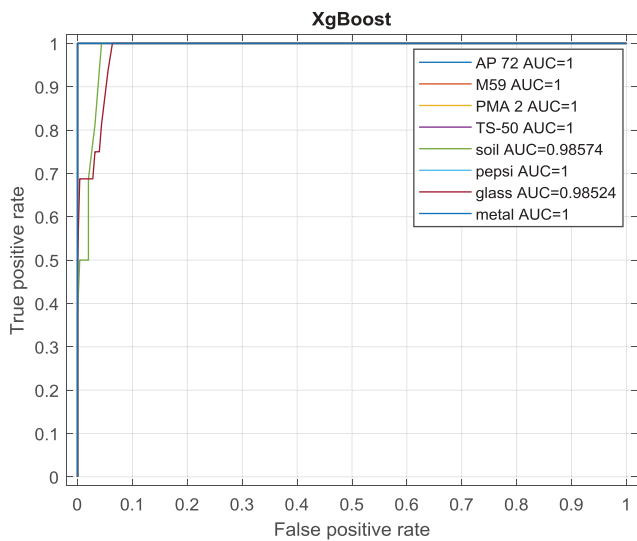


Figure 16. ROC and AUC results for XgBoost.

Confusion Matrix for CNN

Output Class	AP 72	M59	PMA 2	TS-50	soil	pepsi	glass	metal	Accuracy	Missed
1	76 28.4%	0 0.0%	4 1.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	95.0%	5.0%
2	0 0.0%	20 7.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100%	0.0%
3	3 1.1%	0 0.0%	37 13.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	92.5%	7.5%
4	0 0.0%	0 0.0%	0 0.0%	64 23.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100%	0.0%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 5.6%	0 0.0%	1 0.4%	0 0.0%	93.8%	6.3%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	0 0.0%	0 0.0%	100%	0.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	11 4.1%	0 0.0%	5 1.9%	0 0.0%	31.3%	68.8%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 6.0%	100%	0.0%
	96.2% 3.8%	100% 0.0%	90.2% 9.8%	100% 0.0%	57.7% 42.3%	100% 0.0%	83.3% 16.7%	100% 0.0%	92.9%	7.1%
	AP 72	M59	PMA 2	TS-50	soil	pepsi	glass	metal		

Target Class

Figure 17. Confusion matrix for CNN.

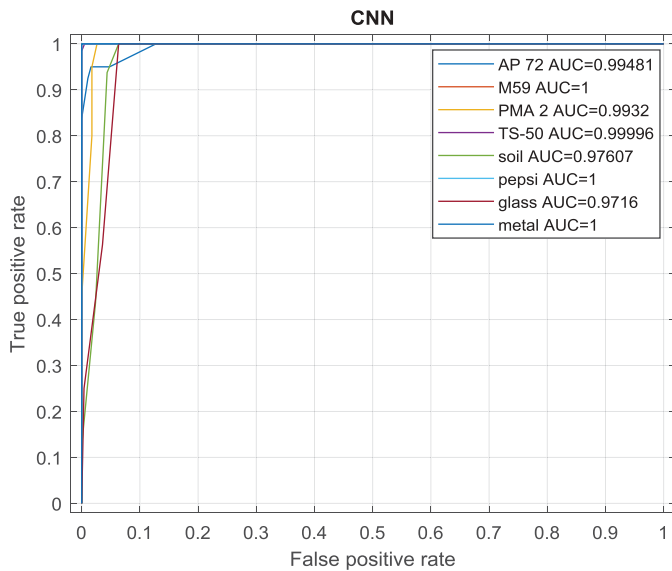


Figure 18. ROC and AUC results for CNN.

```
fitcensemble(X,Y,'Method','Bag','Learners','Tree','NumLearningCycles',100);
```

Figures 13 and 14 for the Adaboost technique show that the algorithm sometimes confuses landmine M-59 with TS-50 and the other way around. The highest error lies in confusing soil as glass but glass is always correctly classified unlike other methods which shows that Adaboost is biased towards glass. In this work, Adaboost was used with Matlab with standard parameters such as a classification tree with as the learner and with 100 classification trees which is similar to bagging. Adaboost is preformed according to the following command:

```
fitcensemble(X,Y,'Method','AdaBoostM2','Learners','Tree','NumLearningCycles',100);
```

Figures 15 and 16 for the XgBoost algorithm show that the algorithm has few errors in confusing landmine AP-72 as PMA-2. The highest error lies in confusing glass with glass and the other way around. We utilised the XgBoost package provided for python with a learning rate of 0.1 and 100 decision trees for the main learner. The main command utilised for XgBoost is according to the following:

```
XGBClassifier(max_depth = 7,min_child_weight = 1,learning_rate = 0.1,n_estimators = 100,silent = True,
gamma = 0,max_delta_step = 0,subsample = 1,colsample_bytree = 1,colsample_bylevel = 1,reg_alpha = 0,
reg_lambda = 0,scale_pos_weight = 1,seed = 1,missing = None,objective = 'multi:softprob')
```

Figures 17 and 18 for CNN show that there are few errors in confusing landmine AP-72 with PMA-2 and the other way around. The highest error lies in confusing soil with glass followed by the other way around. The ADAM optimiser was used to train the CNN with a learning rate of 0.001 with the loss function being the categorical cross-entropy. Python was used with Keras based on Tensorflow. The main commands utilised are according to the following:

```
model.compile(optimizer= Adam(lr = 1e-3), loss = 'categorical_crossentropy',metrics = ['mae', 'accuracy'])
early_stopping = EarlyStopping(monitor = 'val_loss', mode = 'min',patience = 20, verbose = 1)
model_checkpoint = ModelCheckpoint(save_model_name, monitor = 'val_loss', mode = 'min',
save_best_only = True, verbose = 1)
reduce_lr = ReduceLRonPlateau(monitor = 'val_loss', mode = 'min', factor = 0.5, patience = 5,min_lr = 0.00001,
verbose = 1)
```

Besides showing the results for each algorithm, a comparison is shown in Table 5 which provides the mean accuracy and standard deviation of each technique for the 3-folds in addition to the average rank of each algorithm to provide a better overview of the different performances. In addition, Table 5 provide the margin of error (MOE) for each approach assuming a 95% confidence interval along with the lower confidence and upper confidence levels respectively (LCL and UCL).

Table 5 indicates that XgBoost provides a stable accuracy but all results are rather comparable in terms of stability and Bagging is clearly the best performing algorithm.

However, Table 5 does not show the per class stability performance, so, it is worthy showing the results for each algorithm with each class and these are displayed in Table 6.

Table 5. Results of the different utilised algorithms.

Method	LDA	Q-SVM	Bagging	Adaboost	XgBoost	CNN
Accuracy \pm (std)	85.6 \pm 3.1	90.3 \pm 2.6	95.4 \pm 2.1	92.1 \pm 2.8	94.0 \pm 0.9	92.2 \pm 1.3
Average Rank	6	5	1.33	3.33	1.67	3.67
MOE	\pm 0.43	\pm 0.36	\pm 0.29	\pm 0.39	\pm 0.13	\pm 0.18
LCL	85.1	89.9	95.1	91.7	93.8	92.0
UCL	86.0	90.6	95.7	92.5	94.1	92.4

Table 6 shows that bagging is indeed the best performing algorithm with the highest number of 100% per class accuracy, which is for 5 classes, and with the highest low accuracy of 66.7% for the soil classification which is the most common error in all algorithms especially when it is mistaken as glass and vice versa. Bagging is also the best performing algorithm if we consider the per class average accuracy as the evaluation metric.

Moreover, it is insightful to provide a statistical analysis of the obtained results. The analysis of variance or (ANOVA) can be considered as a statistical analysis technique to examine possible differences between similar variables. It is utilised in this work to assess the performance of the utilised algorithms. Figure 19 displays the boxplot that compares the resulting accuracies of the different techniques which validates that bagging is clearly the best performing method followed by XgBoost and so on according to the ranking displayed in Table 5.

The second learning problem is three-class to distinguish between mines, non-metallic, and metallic objects. It is important to emphasise this because the main purpose is classifying landmines. Interestingly, all the results for all the algorithms achieved 100% accuracy as shown in Figure 20. The AUC for the three classes is 1 so there is no need to show the ROC curves for these cases.

Discussion

Based on results presented in the above section, several remarks and contributions should be addressed. First, the results of the different algorithms are relatively close, and more importantly,

Table 6. Results of the different utilised algorithms for each class.

Accuracy±(std) for Method per class	LDA	Q-SVM	Bagging	Adaboost	XgBoost	CNN
AP-72	95.0 ± 5	88.3 ± 5	96.2 ± 1.2	98.3 ± 2.8	97.8 ± 2.7	97.1 ± 2.7
M-59	100 ± 0	100 ± 0	100 ± 0	100 ± 0	93.3 ± 11.5	100 ± 0
PMA-2	76.6 ± 11.5	90.8 ± 9.5	100 ± 0	86.7 ± 11.5	91.7 ± 2.9	89.1 ± 10
TS-50	91.6 ± 14.4	94.3 ± 0.9	100 ± 0	100 ± 0	100 ± 0	100 ± 0
Soil	66.7 ± 25	64.1 ± 26.4	66.7 ± 13.0	51.1 ± 24	55.4 ± 15.2	52.5 ± 28.1
Pepsi	100 ± 0	100 ± 0	100 ± 0	100 ± 0	91.7 ± 14.4	100 ± 0
Glass	51.7 ± 18.8	72.9 ± 9.5	70.8 ± 7.3	83.3 ± 14.4	79.7 ± 14.4	83.1 ± 2.3
Metal	91.7 ± 14.4	100 ± 0	100 ± 0	91.7 ± 14.4	91.7 ± 14.4	100 ± 0
Average (Per Class)	84.2	88.8	91.7	88.9	87.8	90.2

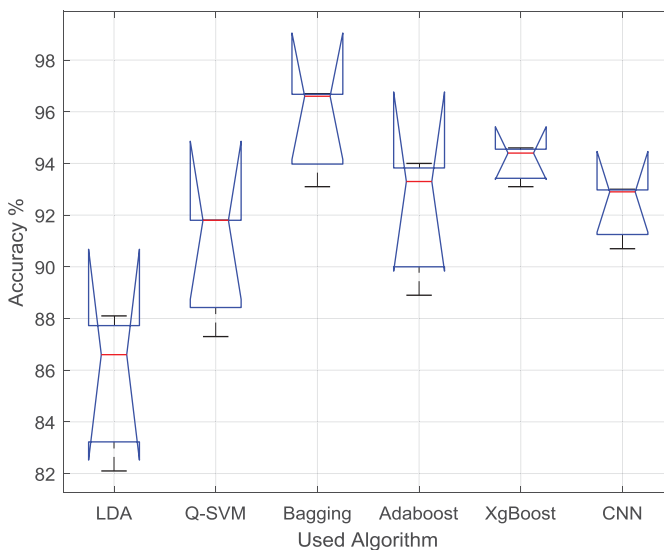


Figure 19. Boxplot comparing the utilised algorithms.

Confusion Matrix for 3 Categories

Output Class	Mine	204 76.1%	0 0.0%	0 0.0%	100% 0.0%
	Non Metallic	0 0.0%	32 11.9%	0 0.0%	100% 0.0%
	Metallic	0 0.0%	0 0.0%	32 11.9%	100% 0.0%
		100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%
		Mine	Non Metallic	Metallic	
		Target Class			

Figure 20. Classification results for 3 categories using all algorithms.

each classification algorithm never outputs a false negative or fails to detect a mine. This was further demonstrated while using only 3 classes and achieving a 100% accuracy with all classifiers as shown in Figure 20. This implies that the presented methodology could be fully approved according to the demining protocols since it is compliant the safety margins of the deminers. Next, we discuss some of the important ideas related to this work.

Data Collection

The data collection is a critical part in this work. We relied on the data acquisition setup discussed in the methods section. The addressed data in this work was obtained from a metal detector manufactured by Ceia. Since Ceia is not the only manufacturer of metal detectors, it is expected that other detectors will output different data. This was observed previously in literature as in (Pinar et al., 2015) and (Bruschini et al., 1998) where the metal detector data is clearly different. Thus, it is expected that a different detector would lead to another database, which would require to repeat the training phase of the model. In this case, the classification algorithms should be also revisited to select the most accurate model.

In addition, the utilised setup can be used for collection of other types of data such as GPR based data which are extensively used in literature. This represents an important outcome since data from multiple sources can be combined to yield a complete and more accurate learning model.

Convolutional Neural Networks

We discuss hereafter the utilised architecture of the CNN. Several architectures were tested and the most accurate model, described in the methods section, was selected. The Keras library (Chollet, 2018) with a Tensorflow (Abadi et al., 2016) backend was used with the Python programming language to test the different architectures where the size of the input images was varying, in addition to the different combinations of layers and the activation functions. The most adequate

architecture is the 11x10x3 images discussed in the preprocessing section, which uses 3 channels without any resizing or additional processing.

In addition, no residual layers were used in the architecture to avoid any negative effect on the performance. The training time of the used network can be completed in less than 10 minutes on an Intel CPU noted in the time analysis section, which allowed testing for multiple architectures.

Feature Analysis

The features presented in the methods section can be considered as direct features in terms of their computation technique or for being standard features in image processing. The features were tested in accordance with the classification approaches excluding the CNN. Other features were also tested including the histogram of oriented gradients, local binary patterns, and other filtering kernels but no added value was observed in terms of the accuracy of the results. It is therefore important to reassess these features with new and considerably bigger potential databases.

In any case, analysing the utilised features and testing whether they are optimal or not by selecting the best set of features is definitely important for classification algorithms. For this purpose, it is possible to rank the features according to their importance based on the utilised algorithm. We concentrate on the ranking obtained using XgBoost and Bagging, so we provide the first important 20 features in Table 7. These 20 features are considered by averaging the importance for each of the 3-folds as opposed to using one only or the whole data directly.

Table 7. First 20 important features with XgBoost and Bagging.

Method	XgBoost	Bagging
Rank 1	Entropy of input	Max mean along rows of input
Rank 2	Maximum of input	Mean of input
Rank 3	Mean of input	Standard Deviation of Row 6 in Horizontal Derivative Image
Rank 4	Standard Deviation of Row 6 in Horizontal Derivative Image	Maximum of input
Rank 5	Entropy of Vertical Derivative Image	Mean column 5 of input
Rank 6	Minimum standard deviation along rows of Vertical Derivative Image	Max mean along columns of input
Rank 7	Standard Deviation of Row 5 in Horizontal Derivative Image	Max standard deviation along columns of input
Rank 8	Standard Deviation of Row 7 in Horizontal Derivative Image	Entropy of Vertical Derivative Image
Rank 9	Min standard deviation along columns of Horizontal Derivative Image	Min standard deviation along columns of Horizontal Derivative Image
Rank 10	Entropy of Horizontal Derivative Image	Max standard deviation along rows of Horizontal Derivative Image
Rank 11	Min mean along columns	Standard Deviation of Horizontal Derivative Image
Rank 12	Minimum Mean along rows of Horizontal Derivative Image	Standard Deviation of column 5 in input
Rank 13	Max mean along rows of Horizontal Derivative Image	Standard Deviation of column 6 in input
Rank 14	Max standard deviation along rows of Horizontal Derivative Image	Standard Deviation of Row 4 in Horizontal Derivative Image
Rank 15	Histogram bin-2 of input	Standard Deviation of column 10 in Vertical Derivative Image
Rank 16	Mean row 6 of input	Standard Deviation of column 5 in Vertical Derivative Image
Rank 17	Min mean along columns of Vertical Derivative Image	Entropy of input
Rank 18	Max mean along columns of Vertical Derivative Image	Max standard deviation along rows of Vertical Derivative Image
Rank 19	Histogram bin-5 of input	Max standard deviation of columns in Horizontal Derivative Image
Rank 20	Min standard deviation along rows of input	Standard Deviation of input

It is possible here to perform the classification while utilising only a subset of features. If we only use the first 20 features, a similar accuracy is obtained especially in XgBoost case and a slightly lower accuracy in the case of Bagging. However, XgBoost considers some features with no importance and these can be removed but this is not the case for bagging.

Regarding whether the utilised set of features is optimal or not, one should test all the combinations of features which is extremely time consuming. Still, the utilised features achieved a 100% accuracy in the 3 categories case which indicates that the set of features is near optimal for this task.

Time Analysis

This section discusses the time and performance of the different tasks in this work. Regarding the data collection, for each sample, 110 measurements are collected. The head of the MD is moved to each position corresponding to one of the 110 points. The time required to obtain a single data value is below 1 microsecond. The time needed to move from one point to another is approximately 3.5 seconds. As a result, each set of measurements takes less than 7 minutes with most of this time going to the mechanical movement of the MD.

Regarding the feature based algorithms, the process of performing the feature computation for a single sample requires less than 1 millisecond (ms) on an Intel i7 CPU i5500 running at 2.4 GHz and having 8 GB of memory using Matlab, which indicates that this process can be considered trivial. For predicting the class of the sample, this varies according to the algorithm but does not exceed 1.2 (ms) for a sample using the Bagging algorithm running on Matlab which is the highest required execution time amongst the chosen algorithms. This means that all the process requires less than 2.2 (ms) including XgBoost running on Python which is faster than the bagging and Boosting algorithms.

Regarding CNN, the process of obtaining an output directly through the signal including the preprocessing time requires approximately 400 (ms) on the same CPU without any GPU operations using Python with Keras on Tensorflow.

All the noted time intervals are measured for the execution or testing part without any training, but in any case the training time does not exceed 10 seconds for any of the feature based algorithms and 10 minutes for the CNN.

Future work

The results of this work can be considered of high accuracy especially that no false negatives or cases where a mine was missed occurred. This will preserve the margin safety rate of deminers. The results indicate that the metal detector accurately distinguishes the quantities or the sizes of the metallic objects since the mines had much smaller metallic parts in comparison with the pepsi can and the other metallic item. Therefore, it is important to note that future work should consider various objects where there is a great variety of shapes. Also, future work should account for much more cases and the data prior to any augmentation should have more than a 1000 measurement. Moreover, a main objective of this work is to account for additional cases of landmines with varying metal parts to enhance the database and ensure a safer process. Once more data is available, the dataset should be updated along with all the classification models and as indicated in the previous time analysis section such a process does not require much computational time. Besides these, planning field tests that mimic real scenarios would result a more accurate and significant database.

Conclusion

A special automated robotic rail setup is implemented to fully utilise metal detectors and their output data to build a database of signatures for buried objects with emphasis on landmines. Several machine learning algorithms such as boosting, bagging and neural networks are tested to accurately classify the buried objects and distinguish the mine from non-mine objects. Utilising this approach

helps the deminer in differentiating between objects without solely relying on the sound signals emitted from the metal detector. This work should be further updated by extending the database with more samples and performing field or real-like experiments with trained individuals to validate the proposed ideas. It should lead to a fully automated system that reduces the risk of human injuries and provides a safer and faster demining process.

Acknowledgments

This work was made possible by the help of the Lebanon Mine Action Center (LMAC) and the contributions of Mr. Stuart Henley and metal detector manufacturer CEIA. This work was partially funded by a University Research Board fund from the American University of Beirut (AUB), and by an Associated Research Unit fund from the Lebanese National Council for Scientific Research (CNRS).

Disclosure Statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Lebanese National Council for Scientific Research; University Research Board of the American University of Beirut (AUB).

ORCID

L. Safatly  <http://orcid.org/0000-0003-4573-4721>

References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., & Kudlur, M. (2016). *Tensorflow: A system for large-scale machine learning*. In *12th USENIX symposium on operating systems design and implementation OSDI, Savannah, GA, USA, 16* (pp. 265–283).
- Baertlein, B. (2003). Infrared/hyperspectral methods (paper I). In *Alternatives for landmine detection, 24*. Santa Monica, USA: RAND.
- Bruschini, C., Gros, B., Guerne, F., Pièce, P. Y., & Carmona, O. (1998). Ground penetrating radar and imaging metal detector for antipersonnel mine detection. *Journal of Applied Geophysics, 40*(1–3), 59–71.
- Chen, T., & Guestrin, C. (2016). *Xgboost: a scalable tree boosting system*. In: *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, USA*, (pp. 785–794). ACM.
- Chollet, F. (2018). *Keras: The python deep learning library* (Astrophysics Source Code Library). <https://keras.io/>
- Collins, L., Gao, P., Makowsky, L., Moulton, J., Reidy, D., & Weaver, D. (2000). *Improving detection of low-metallic content landmines using EMI data*. In *IGARSS 2000. IEEE 2000 international geoscience and remote sensing symposium. taking the pulse of the planet: The role of remote sensing in managing the environment. Proceedings (Cat. No. 00CH37120)* (Vol. 4, pp. 1631–1633). Honolulu, HI, USA: IEEE.
- Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119–139.
- Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning (Vol. 1, No. 10)*. New York: Springer series in statistics.
- Harper, R. J., & Furton, K. G. (2007). Biological detection of explosives. In Jehuda Yinon (Ed.), *Counterterrorist detection techniques of explosives* (pp. 395–431). Elsevier Science BV.
- Hastie, T., Tibshirani, R., Friedman, J., & Franklin, J. (2005). Reviews-the elements of statistical learning: data mining, inference and prediction. *Mathematical Intelligencer, 27*(2), 83–84.
- Hristakeva, M. (2008). *Different boosting algorithms and underlying optimization problems*. Santa Cruz, USA: University of California.
- Kaneko, A. M., Fukushima, E., & Endo, G. (2014). A discrimination method for landmines and metal fragments using metal detectors. *Journal of Conventional Weapons Destruction, 18*(1), 16.

- Krausa, M., Massong, H., Rabenecker, P., & Ziegler, H. (2002). Chemical methods for the detection of mines and explosives. In Hiltmar Schubert and Andrey Kuznetsov (Eds.), *Detection of explosives and landmines* (pp. 1–19). Dordrecht: Springer.
- Kruger, H., & Ewald, H. (2008, October). *Handheld metal detector with online visualisation and classification for the humanitarian mine clearance*. In *SENSORS, 2008 IEEE* (pp. 415–418). Lecce, Italy: IEEE.
- MacDonald, J., Lockwood, J. R., McFee, J., Altshuler, T., & Broach, T. (2003). *Alternatives for landmine detection (No. RAND/MR-1608-OSTP)*. Santa Monica, CA, USA: Rand Corp Santa Monica CA.
- Mazhar, R., Gader, P. D., & Wilson, J. N. (2009). Matching-pursuits dissimilarity measure for shape-based comparison and classification of high-dimensional data. *IEEE Transactions on Fuzzy Systems*, 17(5), 1175–1188.
- Mokalled, L., Al-Husseini, M., Kabalan, K. Y., & El-Hajj, A. (2014). Sensor review for trace detection of explosives. *International Journal of Scientific and Engineering Research*, 5(6), 337–350.
- Pinar, A., Masarik, M., Havens, T. C., Burns, J., Thelen, B., & Becker, J. (2015, May). Approach to explosive hazard detection using sensor fusion and multiple kernel learning with downward-looking GPR and EMI sensor data. In *Detection and sensing of mines, explosive objects, and obscured targets XX* (Vol. 9454, pp. 94540B). Bellingham, Washington, USA: International Society for Optics and Photonics.
- Schoolderman, A., & Barrell, Y. (2007). Improving productivity in manual demining by magnetic clutter reduction? The Forum 2007, Orlando, Florida. Retrieved from: https://www.gichd.org/fileadmin/pdf/LIMA/SCHOOLDERMAN_UXOForum2007.pdf
- Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., & Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35(5), 1285–1298.
- Strand, J., & Taxt, T. (1994). Local frequency features for texture classification. *Pattern Recognition*, 27(10), 1397–1406.
- Tantum, S. L., & Collins, L. M. (2001). A comparison of algorithms for subsurface target detection and identification using time-domain electromagnetic induction data. *IEEE Transactions on Geoscience and Remote Sensing*, 39(6), 1299–1306.
- Tran, M. D. J., Abeynayake, C., & Jain, L. C. (2012). A target discrimination methodology utilizing wavelet-based and morphological feature extraction with metal detector array data. *IEEE Transactions on Geoscience and Remote Sensing*, 50(1), 119–129.
- Waschl, J. (1994). *A review of landmine detection*. Defense Sci. Technol. Org.: Aeronaut. Edinburgh, Australia: Maritime Res. Lab., Explosives Ordnance Div.