Prediction of size of buried objects using Ground Penetrating RADAR and Machine Learning Techniques

Nairit Barkataki
Dept. of Instrumentation & USIC
Gauhati University
Guwahati, India

Sharmistha Mazumdar
Dept. of Instrumentation & USIC
Gauhati University
Guwahati, India

Rajdeep Talukdar
Dept. of Instrumentation & USIC
Gauhati University
Guwahati, India

Priyanka Chakraborty
Dept. of Instrumentation & USIC
Gauhati University
Guwahati, India

Banty Tiru
Dept. of Physics
Gauhati University
Guwahati, India

Utpal Sarma
Dept. of Instrumentation & USIC
Gauhati University
Guwahati, India

***Abstract*—Ground penetrating radar (GPR) uses electromagnetic (EM) wave to detect the subsurface objects. Interpretation and analysis of GPR signals are still challenging tasks as it requires skilled user (geologists in most cases). Particularly difficult is the prediction of the object sizes. This paper proposes a new method for predicting size of buried objects. First, standard scaling pre-processing techniques are used to optimise the B-Scan data. The features are then supplied to Random Forest (RF) and Support Vector Machine (SVM) classifiers to automatically predict the size of the buried object. The proposed feature based RF classifier shows similar performance in the accuracy of classification compared to SVM (Radial Basis Function kernel) system.**

***Index Terms*—machine learning, classification, object size prediction, ground penetrating radar**

I. Introduction

Ground Penetrating Radar (GPR) is used to detect subsurface objects with high resolution.GPR transmits High frequency electromagnetic wave through the ground at a velocity depending upon the dielectric property of the material [1]. Ground penetrating radar can be used to uncover hidden sources of disaster [2]. Consequently, it has been used in some applications such as estimation of bridge state of deterioration [3], soil surveys [4], pavement and sub-pavement structure diagnosis [5]. GPR can be used for investigation of tree root biomass which helps in soil amelioration, water infiltration, and prevention of erosion [6]. In short, GPR has applications in a varied fields like environmental, archaeological, civil engineering, military, geophysical and so on [7].

The interpretation of the collected GPR data requires highly skilled human operator along with experience. Also the subsurface signals contains noises which cannot be removed completely. These problems led to growing interest in the development of automated subsurface object detection and identification technique. One of the most significant way of meaningful physical interpretation of GPR data is the usage of machine learning algorithms. Applications of machine learning in GPR data interpretation includes automatic material classification of underground objects [8], detection of Landmines [9], [10], image processing for recognition of GPR data [11], predicting geometry of buried object [1] and many more.

Aim of present work

A lot of research work has been done in this field but prediction of size of the buried objects in different environmental conditions remains to be carried out extensively. [1] mainly concentrates on the size of basic objects (rectangle, circle and triangle) but with very few scenarios (<50). Moreover, the model uses a single classifier and does not compare its performance with other classifiers. This paper mainly focuses on -

1) Prediction of size of cylindrical object in 100 different scenarios.
2) Prediction of object size is done using both Random Forest and Support Vector Machine classifiers.
3) Test results of both the classifiers are compared to obtain the best fitted model.

II. GPR Data

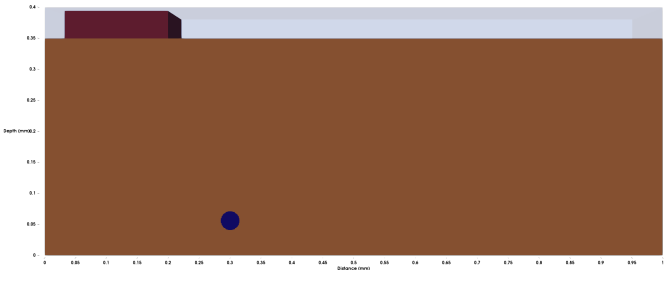

Fig. 1. Simulated model

A. Database Creation

The samples are created by using the electromagnetic simulator gprMax. gprMax is an open source software developed by researchers at the University of Edinburgh which simulates electromagnetic wave propagation for numerical modelling of Ground Penetrating Radar (GPR). It is based on Finite-Difference Time-Domain (FDTD) method [12]. The simulations are further accelerated using NVIDIA's CUDA framework [13].

Before simulation, several parameters have to be set to create a scenario. One of the models generated after simulation is shown in Figure 1.

The size of the model is $1000 \times 148 \times 400$ $(X \times Y \times Z)$ mm. The height of the model is 400 mm of which the top 50 mm layer is air and the remaining 350 mm is a soil layer. An object is buried at a depth of 294 mm from the surface of the soil layer. The object is a metallic cylinder made of aluminium (ϵ_r = 10.8, $\sigma = 3.5 \times 10^7$ Siemens/metre).

A total of 100 scenarios are created for 10 soil types and 10 object radii. The parameters used for creating the various soil types are shown in Table I

TABLE I
SOIL PROPERTIES

Sl. No.	Soil Type	Conductivity σ (S/m)	Relative permittivity ϵ_r
1	Dry, sandy, flat (coastal)	0.002	10
2	Pastoral Hills, rich soil	0.007	17
3	Pastoral medium hills and forestation	0.005	13
4	Fertile land	0.002	10
5	Rich agricultural land (low hills)	0.01	15
6	Rocky land, steep hills	0.002	12.5
7	Marshy land, densely wooded	0.0075	12
8	Marshy, forested, flat	0.008	12
9	Mountainous / hilly (to about 1000 m)	0.001	5
10	Highly moist ground	0.01	30

The object radius is changed from 10 mm to 55 mm. Parameters for the FDTD simulation are listed in Table II. Time window must be large enough to allow the EM waves

TABLE II
SIMULATION PARAMETERS

Parameter	Parameter value
Time Window	7-14 ns
Frequency	1500 MHz
Excitation Waveform Type	Ricker
Spatial resolution	2 mm
A-Scan internals	5 mm
No of A-Scans	100

to propagate from the Tx antenna through the soil, reflected back by the buried object and received by the Rx antenna. Its value primarily depends on the depth of the object and soil properties (ϵ_r & σ). Hence, 5 values of time windows are calculated for different values for ϵ_r of soil as given in Table III

TABLE III
TIME WINDOW VALUES

Time Window (ns)	Soil ϵ_r
7	5
9	10
10	12, 12.5, 13
11	15, 17
14	30

All simulations are accelerated using NVIDIA GPUs P100, T4, K80 and P4 through use of the NVIDIA CUDA programming environment. To save computation time, only 100 A-Scans are taken, which are enough to display the target object completely. As a reference, Figure 2 shows B-Scan (120 A-Scans) of an object buried at 270 mm below the soil surface. Each B-Scan, comprising of 100 A-Scans took 30-58 minutes depending on the GPU alloted.

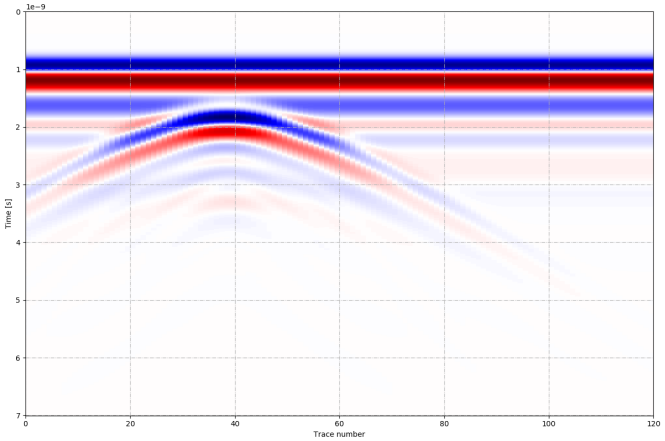

Fig. 2. A simulation result

III. DATA PREPROCESSING

The B-Scan data generated by gprMax uses HDF5 file format. All 100 B-Scan files are read and stored in a two dimensional (2D) array. The array consists of A-Scans, Soil σ, Soil ϵ_r, Object Radius, Object Depth, Object ϵ_r and Object σ. Since there are a large number of A-Scans and features with large variations, the data is normalised and redundant data removed to reduce computational requirements. The reduced dataset has little more than 8 million data-points which is then split into train-test set in 80:20 ratio.

IV. CLASSIFICATION & PREDICTION

A. *Hyperparameter Tuning:*

Hyperparameters are the variables in a machine learning algorithm that govern the whole training process. In Random

Forest classifier, two hyperparameters are generally considered, viz. the number of features considered by each tree while splitting of a node and the number of trees in the forest. But these parameters must be set manually before training and it's impossible to determine the hyperparameters ahead of time. Hyperparameter tuning makes this process of determining the best hyperparameters for a given model much easier and less tedious. It works by running multiple trials in a single training job and optimises the performance of the model.

B. Cross Validation:

Optimisation of model for the training data leads to very high performance of the model only on data for training but may show poor performance on test set. Cross validation (CV) can be used to solve this problem.

One of the commonly used methods for Cross Validation is K-Fold CV. In this method, the dataset is split into k different subsets. The model is trained for k-1 subsets and the last subset is used a test set. This process is done k times with each time having different combinations of subsets for train set and test set, as shown in Figure 3. The average performance for each fold is obtained and the model with the best performance score is finalised.

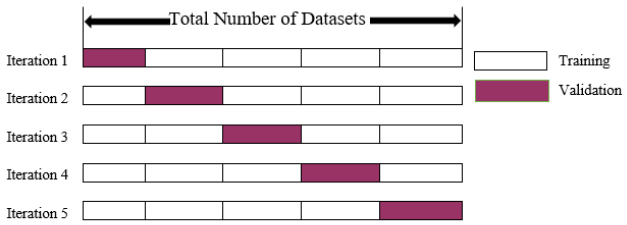

Fig. 3. Visualisation of 5-fold cross validation

For 5-Fold CV , the dataset is split into 5 subsets. In the first iteration, the model is trained on last 4 folds and evaluated on the first fold. In the second iteration, training is done on the first, third, fourth and fifth folds and evaluated for the second fold. This process is repeated 3 more times, validating a different fold each time. When the training ends, the average of performance of each fold is obtained to come up with the final validation. Each training and testing iteration is given a score based on accuracy. In a multi-class problem, the fraction of correct predictions (acc) over m_{samples} is defined as [14],

$$\texttt{acc}(y, \hat{y}) = \frac{1}{m_{\text{samples}}} \sum_{j=0}^{m_{\text{samples}}-1} 1(\hat{y}_j = y_j) \qquad (1)$$

Here,
y is the set of true pairs,
$\hat{y}$ is the set of predicted pairs,
y_j is its true value,
$\hat{y}_j$ is the predicted value, and
$1(x)$ is the indicator function

A 5-fold cross validation is performed on both Random Forest and SVC (RBF) classifiers to tune their respective hyperparameters. For Random Forest, the number of trees (estimators) and their quality of split are the hyperparameters that are tuned. In case of SVC (RBF), C and gamma values are tuned. The gamma parameter describes the extent of impact of a training sample. High and low values of this parameter imply 'close' and 'far' respectively. The C parameter tries to balance between decision function's margin and correct classification of training samples. Smaller margin will be considered for larger values of C, if the decision function is better in classifying all the training samples correctly. A lower value of C will implies a larger margin, and hence a simpler decision function, compromising the training accuracy [14].

The performance scores of different values of hyperparameters for Random Forest and SVC (RBF) classifiers are shown in Figures 4 & 5 respectively. It is seen that an estimator size of 100 with entropy criteria for split quality give the best performance score of 0.9993. On the hand, C=100 and gamma=0.1 give the best performance score of 0.9995 for SVC (RBF).

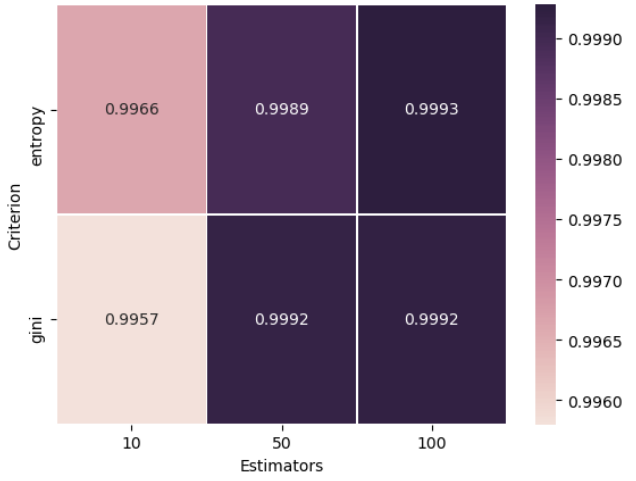

Fig. 4. CV scores with varying hyperparameters in RF classifiers

TABLE IV
PERFORMANCE OF DIFFERENT CLASSIFIERS

Classifier	Accuracy	F Measure	Mean Squared Error	R Squared
Random Forest (estimators = 100)	99.97%	1.00	0.02106	0.9998
SVC-Linear	11.5%	0.09	455.7091	-1.2375
SVC-RBF (C=100, gamma=0.1)	99.95%	1.00	0.0016	0.9999

Performance characteristics of the best optimised model of each classifier are shown in Table IV. When using all 10 classes, the confusion matrix obtained from Random Forest classifier with 100 estimators is shown in Figure 6, where the true class labels are listed along the y-axis and the Random

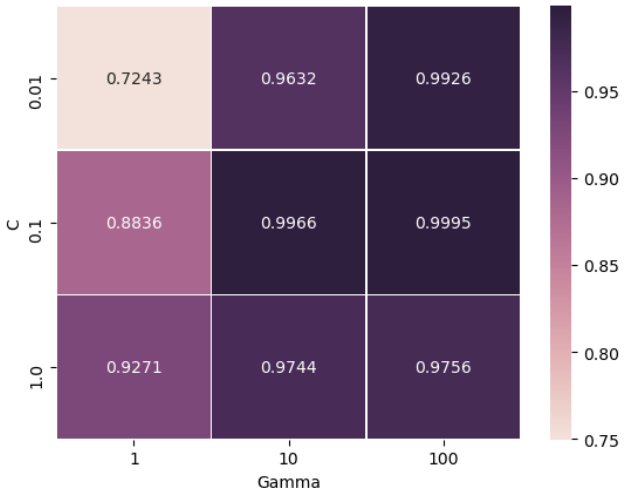

Fig. 5. CV scores with varying values of hyperparameters in SVC (RBF)

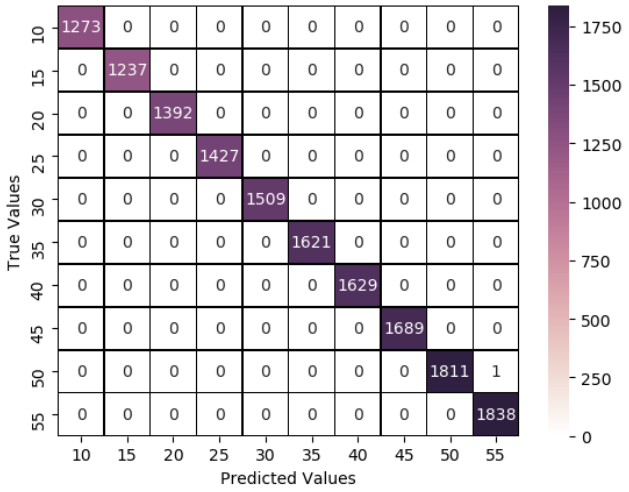

Fig. 7. Confusion matrix for SVC (RBF) classifier

Forest class predictions along the x-axis. Like wise, confusion matrix of SVC (RBF) classifier is shown in Figure 7.

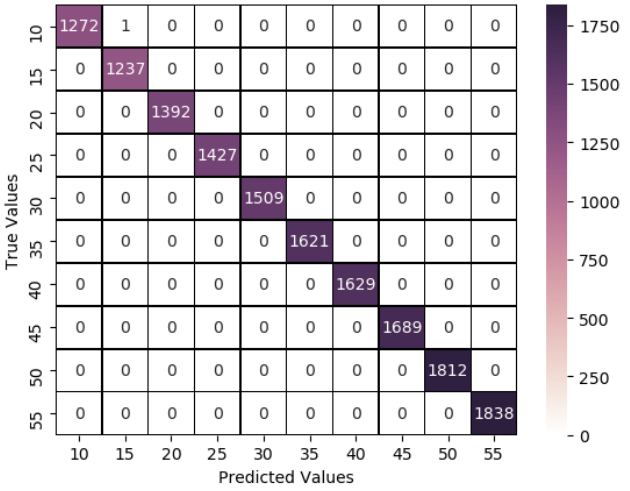

Fig. 6. Confusion matrix for Random Forest classifier

V. Conclusion

In the confusion matrix of RF classifier, it can be seen that 1 sample of radius 10 is wrongly predicted as radius 15. On the other hand, in SVC (RBF) classifier, 1 sample of radius 50 is wrongly predicted as radius 55. It is seen that both Random Forest and SVC (RBF) classifiers give similar performance results. However, SVC (RBF) classifier has a little better error margin with a mean squared error of 0.0016 and R^2 = 0.9999. All the 10 sizes of buried objects are correctly predicted by both Random Forest and SVC (RBF) classifiers with >99.90% accuracy and >99.90% precision.

A. Limitations

The present work demonstrates machine learning applications in predicting size of buried objects. However, the model can predict the size of the object from among a limited number of values (object radius). Real life scenarios will have have objects with custom sizes and shapes. Additionally, the present model has to be verified experimentally by implementing it on a hardware system.

B. Future Work

The authors plan to improve the proposed model by:

- Generating more datasets by varying other features such as object material, object depth, object shape and so on.
- Implementing and validating the proposed model on a hardware system, being developed by the authors.

Acknowledgement

The authors would like to thank Google for providing a computation platform like Google Colaboratory, without which the dataset generation would have been a tedious task. The authors would also like to thank Dr Manoj Kumar Phukan (Sr. Scientist, Geo Sciences and Technology Division, CSIR-NEIST, Jorhat) who gave detailed insights on interpretation of GPR data and Mr. Arnob Doloi for his inputs regarding database creation from time to time.

References

[1] N. R. Syambas, "An approach for predicting the shape and size of a buried basic object on surface ground penetrating radar system," *International Journal of Antennas and Propagation*, vol. 2012, 2012.

[2] X. Xu, S. Peng, and F. Yang, "Development of a ground penetrating radar system for large-depth disaster detection in coal mine," *Journal of Applied Geophysics*, vol. 158, pp. 41 – 47, 2018.

[3] M. Alsharqawi, T. Zayed, and S. A. Dabous, "Integrated condition rating and forecasting method for bridge decks using visual inspection and ground penetrating radar," *Automation in Construction*, vol. 89, pp. 135 – 145, 2018.

[4] K. Zajícová and T. Chuman, "Application of ground penetrating radar methods in soil studies: A review," *Geoderma*, vol. 343, pp. 116 – 129, 2019.

[5] A. Benedetto, F. Benedetto, and F. Tosti, "Gpr applications for geotechnical stability of transportation infrastructures," *Nondestructive Testing and Evaluation*, vol. 27, no. 3, pp. 253–262, 2012.

[6] K. A. Borden, M. E. Isaac, N. V. Thevathasan, A. M. Gordon, and S. C. Thomas, "Estimating coarse root biomass with ground penetrating radar in a tree-based intercropping system," *Agroforestry systems*, vol. 88, no. 4, pp. 657–669, 2014.

[7] H. M. Jol, *Ground penetrating radar theory and applications*. elsevier, 2008.

[8] Q. Lu, J. Pu, and Z. Liu, "Feature extraction and automatic material classification of underground objects from ground penetrating radar data," *Journal of Electrical and Computer Engineering*, vol. 2014, 2014.

[9] N. Smitha and V. Singh, "Target detection using supervised machine learning algorithms for gpr data," *Sensing and Imaging*, vol. 21, no. 1, pp. 1–15, 2020.

[10] H. Frigui and P. Gader, "Detection and discrimination of land mines in ground-penetrating radar based on edge histogram descriptors and a possibilistic k-nearest neighbor classifier," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 1, pp. 185–199, 2008.

[11] S. Shihab, W. Al-Nuaimy, and A. Eriksen, "Image processing and neural network techniques for automatic detection and interpretation of ground penetrating radar data," in *Proceedings of the 6th International Multi-Conference on Circuits, Systems, Communications and Computers, Cancun, Mexico*, 2002, pp. 12–16.

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

[13] C. Warren, A. Giannopoulos, A. Gray, I. Giannakis, A. Patterson, L. Wetter, and A. Hamrah, "A cuda-based gpu engine for gprmax: Open source fdtd electromagnetic simulation software," *Computer Physics Communications*, vol. 237, pp. 208–218, 2019.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.