Northeastern University
Khoury College of
Computer Sciences

MS Thesis Approval

Thesis Title: Evaluation of Synthetic Training Data and Training-Data-Augmentation Techniques for Object Detection in Ground-Penetrating Radar Data Using Deep-Learning Models

Author: Jean Ruggiero

MS Thesis Approved as an elective towards the Master of Science in Computer Science.

Thesis Advisor	8/13/2021 *Date*
Ian Gorton *Thesis Reader*	8/13/21 *Date*
Thesis Reader	8/14/21 *Date*
Thesis Reader	*Date*
Thesis Reader	*Date*

KHOURY COLLEGE APPROVAL:

Ian Gorton Ian Gorton *Program Director*	8/13/21 *Date*
Program Director (if applicable)	*Date*
Associate Dean for Graduate Programs	8/18/2021 *Date*

COPY RECEIVED BY GRADUATE STUDENT SERVICES:

Ujwala Ramakrishna *Recipient's Signature*	8/18/21 *Date*

Distribution: *Once completed, this form should be scanned and attached to the front of the electronic dissertation document (page 1). An electronic version of the document can then be uploaded to the Northeastern University-UMI Website.*

Evaluation of Synthetic Training Data and Training-Data-Augmentation Techniques for Object Detection in Ground-Penetrating Radar Data Using Deep-Learning Models

Jean Ruggiero

A thesis presented for the degree of
Master of Science
in Computer Science

Khoury College of Computer Sciences
Northeastern University
Seattle, Washington

Advised by
Craig Martell
Khoury College of Computer Sciences

August 2021

Abstract

Deep learning models have been successfully applied to the task of object detection in photographic images. This success has led to experimentation with deep learning object detection in other types of image data, including subsurface radar images captured by ground-penetrating radar (GPR) devices. The high cost of GPR devices and of labeling GPR datasets has led to the use of synthetic GPR data for model training and evaluation in many prior studies. This work examines the current state of deep learning in the subsurface imaging field and expands upon it by evaluating the real-world performance of a synthetically-trained neural network model for object detection in GPR images. Further, three data augmentation techniques for GPR data are proposed and evaluated on a real-world test set: (1) random cropping, (2) negative augmentation, and (3) real noise application.

The results of these data augmentation experiments indicate that none of the data augmentation techniques evaluated leads to improved real-world performance of a synthetically-trained GPR object detection model. The control experiment in which no data augmentation techniques were applied yielded the best-performing model as evaluated on the real-world test set. Additional work is required to determine which, if any, data augmentation techniques will improve the real-world performance of synthetically-trained GPR deep learning models. Several techniques are proposed as topics for future research. The results also demonstrate that a model that performs well on a synthetic validation set does not necessarily perform well on a real-world dataset. Several experiments produced models with an f-1 score of 0.90 or greater when evaluated on the synthetic validation set. These same models produced f-1 scores of 0.50 or lower (worse than random guessing) when evaluated on the real-world test set. The results demonstrate by counterexample that it is not possible to generalize about the real-world performance of a synthetically-trained GPR deep learning model based on results obtained using a synthetic test or validation set.

Contents

List of Figures

List of Tables

1 Introduction

Ground-penetrating radar (GPR) is a subsurface imaging technology that enables detection and classification of objects of interest, or *targets*, buried under the ground or another medium. Despite its usefulness, there are challenges limiting the widespread use of GPR. One of these challenges is the level of expertise needed to operate GPR devices and correctly interpret the resulting data. The application of deep learning to data collected by GPR systems addresses this concern by introducing a model that processes output data and informs the user of the presence, location, size, and shape of any underground objects it detects. Deep learning has the potential to transform the subsurface sensing space in the same way it has transformed the medical imaging field [12].

This work provides an overview of prior applications of deep learning in the subsurface imaging field and expands upon it by evaluating the real-world performance of a synthetically-trained neural network model for object detection in GPR images. Further, techniques for augmenting synthetic GPR datasets are proposed and evaluated. The goal of these data augmentation techniques is to improve the real-world performance of a synthetically-trained model. The cost of GPR equipment and the expertise required to label GPR data have together led to a lack of publicly available labeled datasets of the scale required for machine learning. Previous studies in many problem domains have overcome this challenge by creating synthetic training datasets [20]. Several prior studies have shown that neural networks trained on synthetic GPR datasets generated by physics-based simulations generalize well to synthetic test data generated in the same manner, but none have included model evaluations on real-world GPR datasets.

This work presents the results of several experiments to evaluate the practicality of synthetic GPR data for use in training a deep learning model to perform a real-world object detection task. First, a synthetic GPR dataset was generated using available physics-based simulation software. The synthetic dataset was then partitioned into training and validation datasets. Next, a real-world GPR dataset was manually labeled and set aside to use as a test set. A binary classification convolutional neural network (CNN) for object detection was designed and refined using the training and validation sets. Finally, eight experiments were completed to evaluate the real-world performance of the synthetically-trained object detection model. Three data augmentation techniques were proposed and evaluated: (1) random cropping, (2) negative augmentation, and (3) real noise application. A different subset of these data augmentation techniques was applied to the synthetic training and validation datasets in each experiment. A control experiment was performed in which no data augmentation techniques were applied. In each experiment, the model was trained on the augmented training set and evaluated on the validation and test sets.

Section 2 provides background information about subsurface imaging and GPR. Section 3 provides an overview of the current literature related to the application of machine learning in the subsurface imaging domain and the use of synthetic training datasets for machine learning. Section 4 describes the methodology used to generate datasets, design an object detection model, and perform data augmentation experiments. Section 5 provides a summary of all experiments performed and a discussion of the results. Section 6 draws conclusions from the results presented and proposes topics for future research related to the application of machine learning to the sub-surface imaging field.

2 Subsurface Imaging Background

The goal of subsurface imaging is to generate human-readable "images" that allow human experts to detect and classify objects that are located beneath a surface, typically under the ground. The foremost sensing technology for subsurface imaging is Ground Penetrating Radar (GPR). GPR devices, such as the one depicted in Figure 1, allow users to generate images of the subsurface environment in a non-invasive manner.

Figure 1: Typical GPR device [15]

2.1 Applications

Subsurface imaging, particularly with GPR, has applications in a variety of fields, such as civil engineering, geology and geophysics, archaeology, and mine detection. Civil engineering applications include detection of cracks, voids, and other indications of compromised integrity in roadways [15] and concrete structures [4]. It is also used to collect data for utility location [11] and planning [4]. Applications in geophysics include location of geological structures [4] and glacier mass estimation [7]. Subsurface imaging is often used in archaeological investigations to plan excavations or glean information about buried structures or objects without disturbing them. The use of subsurface imaging for landmine detection has recently become a popular topic for both military and humanitarian purposes [4]. GPR apparatus can be fitted to a robot and controlled remotely in order to locate and mark landmines without putting humans at risk.

2.2 Sensing Mechanism

GPR provides subsurface imaging capability by emitting electromagnetic waves through the subsurface medium and measuring the reflected waveform. The transmitted waves propagate through the subsurface medium and reflect off of or are scattered by targets of interest. A schematic drawing of a simple GPR apparatus is shown in Figure 2.

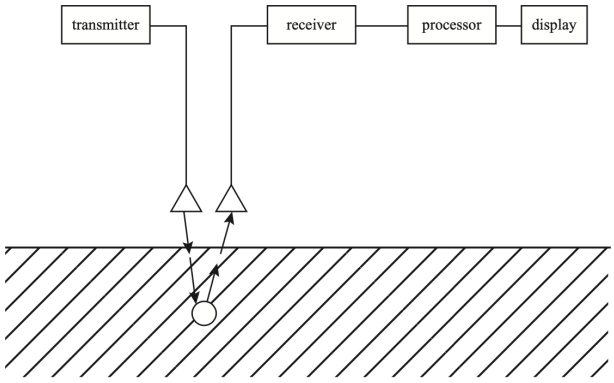

Figure 2: Simple GPR apparatus [4]

Electromagnetic waves are emitted by a transmitting antenna and later detected by a receiving antenna. The transmitted signal is typically a wavelet, such as a Ricker waveform [4], but other waveforms may be used. The received signal will differ from the transmitted signal as a result of the emitted electromagnetic wave propagating through subsurface media and reflecting off of one or more targets. The received waveforms can be used to deduce the presence of subsurface targets and, in some cases, identify features of the target(s). GPR has been used to detect targets such as humans, land mines, utilities, air pockets or voids, and interfaces between layers of different materials (e.g. ice and rock).

As depicted in Figure 2, typical GPR systems include an onboard processor that applies signal conditioning to the received waveforms and assembles them into human-readable "scans." These scans are then displayed to the user via an onboard or remote display. Examples of GPR scans are given in Figure 3 in the following section. A human expert is required to interpret these scans in order to identify targets of interest as well as their approximate size, shape, and location. Assumptions can be made about the material properties of the soil or subsurface medium and used to compute an approximate depth for targets identified by the user [4].

2.3 Data Modality

An example of a GPR image, called a *B-scan*, is shown in Figure 3. A B-scan is a 2-D matrix representing a vertical cross-section of the ground and the radar signatures of any targets located therein. The x-axis of the image represents distance along the surface of the ground. The y-axis represents the two-way travel time of radio waves transmitted into the surface. Often, the time scale can be converted into depth as shown in Figure 3 if the velocity of the transmitted waves through the subsurface medium is known.

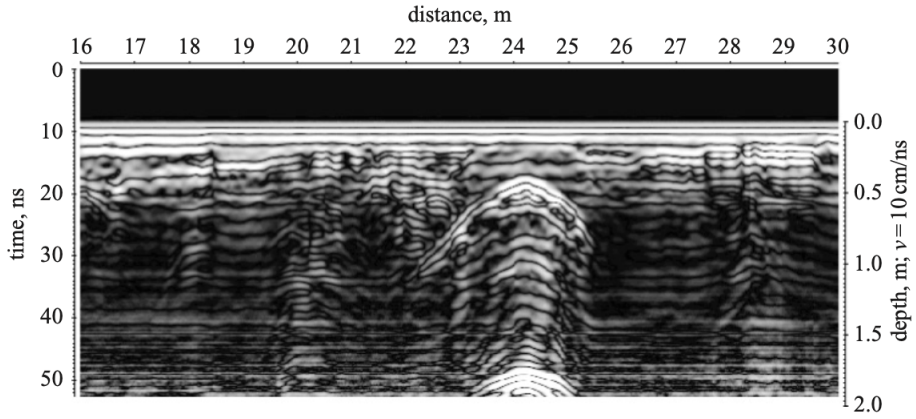

Figure 3: B-scan of underground crypt [4]

Each B-scan is a matrix of the form

$$B = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \dots & \dots & a_{mn} \end{bmatrix} \tag{1}$$

where each element represents the amplitude of the signal received by a GPR receiver antenna at a specific point in time. B-scans are generated by assembling several *A-scans*. Continuing the example above, the given B-scan is composed of n A-scans, where each column of B is a single A-scan. It follows that each A-scan is an m-element vector. B can be re-written as

$$B = \begin{bmatrix} \vec{a}_1 & \vec{a}_2 & \dots & \vec{a}_n \end{bmatrix} \tag{2}$$

where each $\vec{a}_i = \begin{bmatrix} a_{i1} & a_{i2} & \dots & a_{im} \end{bmatrix}^T$ is a single A-scan.

An A-scan is a measured time series of the amplitude of a reflected waveform as detected by the receiving antenna. The amplitude of the received waveform is sampled during a specified window

of time after the transmitted waveform pulse, typically on the order of 50-100 ns [5]. Each A-scan is measured from a fixed location on the surface. A series of A-scans generated at different points along a path are assembled in order to create a B-scan. Figure 4 shows an example of an A-scan and its relationship to the associated B-scan. The number of A-scans measured per linear meter ranges from approximately 30 to 200 [5].

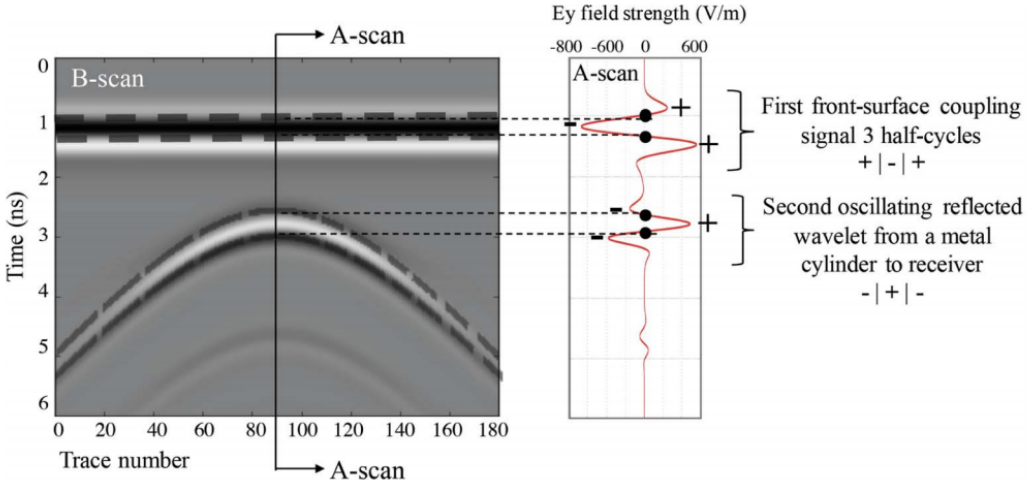

Figure 4: Sample A-scan and its relationship to the B-scan [28]

In the case where a device contains a single GPR transmitter/receiver antenna pair, the device can be moved across the surface, generating A-scans at fixed distance intervals. Once a sufficient number of these A-scans have been recorded, they can be assembled into a B-scan as shown in Figure 5.

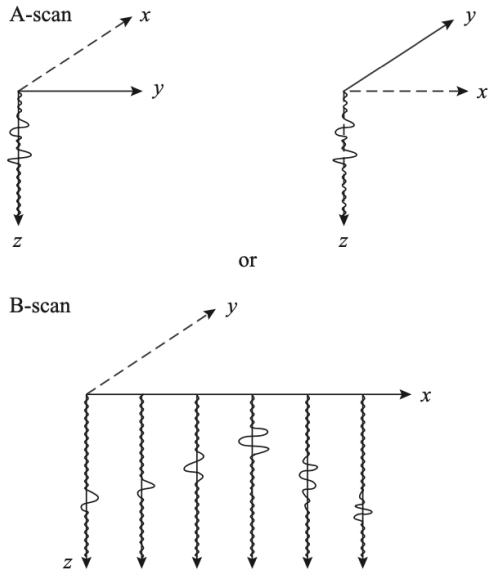

Figure 5: Illustration of generating B-scan from A-scans [4]

3 Prior Work

This section provides an overview of the current state of research related to the use of deep learning models for object detection and classification in GPR scans. It also outlines prior work on the use of synthetic data for deep learning model training.

3.1 Deep Learning Models for Object Detection and Classification in GPR Data

Previous applications of neural networks to detect and classify objects using GPR data have largely used convolutional neural networks (CNNs) to process 2-D B-scans. This section presents a summary of the literature related to the use of neural networks for object detection and classification in GPR scans. The studies performed are divided into two categories: Section 3.1.1 gives an overview of those studies that use synthetic GPR datasets for training and testing and Section 3.1.2 outlines those that use real GPR datasets for training and testing.

3.1.1 Synthetic GPR Data for Model Training and Testing

Several studies have been performed using synthetic GPR scans for both model training and testing. Alamimani et al. achieved a classification accuracy of 90% and above using a model that classifies synthetic GPR scans by size, depth, material, and dielectric constant [1]. Zhang et al. achieve similar classification accuracy in determining the shape and material of targets in synthetic GPR scans[29]. They achieved errors of less than 10% in the prediction of an object's depth and dielectric constant. Similar error rates were observed in the prediction of the size of circular and square objects; the size of trangular objects was more difficult to predict and resulted in an error of approximately 30%. Ponti et al. discuss the use of a DenseNet CNN architecture to predict the relative permittivity, radius, and depth of cylinders in synthetic B-scans [18].

Ozkaya and Seyfi applied a dictionary learning approach to classification of objects by shape in synthetic B-scans [14]. Alvarez and Kodagoda designed a U-net encoder-decoder model for image-to-image transformation with the goal of training a model to output a cross-sectional image of the underground geometry based on synthetic B-scans [2]. Sonoda and Kimoto applied a CNN to estimate the relative permittivity of targets in synthetic B-scans [21].

Unfortunately, large open datasets of GPR scans are not publicly available to researchers. The high cost of GPR devices makes it fiscally challenging to purchase or rent a device for data collection. There are also logistical complications related to excavating and constructing a test site. Thus, the use of synthetic data for model training is attractive. However, none of the studies outlined above include results showing the performance of the model on real-world GPR data.

3.1.2 Real GPR Data for Model Training and Testing

Due to the lack of availability of large, labeled GPR datasets, many studies that leverage real GPR data are performed at a small scale. Tong et al. collected real GPR scans of roadway defects and built a CNN classifier to classify scans as containing no defect, sinkhole, uneven settlement,

or subgrade crack [22]. Their dataset consisted of 267 training scans and 45 test scans and their model achieved 95% classification accuracy. The same group applied CNNs to detect concealed cracks (and their lengths), subgrade settlement, and cavities in pavement using a larger (500 B-scans) dataset [23]. Most recently, Tong et al. collected 27,820 m of GPR data across three different transmission frequencies and used a CNN to detect and compute attributes of three types of pavement stress [24]. However, the paper mentions that data labels were verified by taking core samples. Since it would not be practical to perform core samples of the entire 27,820 m section scanned, it is likely that not all labels were verified against ground truth.

Pham and Lefevre took a hybrid approach by combining 100 real GPR B-scans with 50 simulated B-scans [16]. They describe the application of a Faster-RCNN object detection model pre-trained on the CIFAR-10 dataset in three different scenarios: (1) train and test on simulate data, (2) train and test on real data, and (3) train on a combination of simulated and real data and test on real data. The authors present two examples of B-scans labeled by their model, but include no quantitative results of classification performance.

Picetti et al. applied an autoencoder to detect objects in GPR B-scans [17]. The authors trained the model only on B-scans not containing a buried object and then applied it as an anomaly detector to detect the presence (anomaly) or absence (normal) of an object in a GPR B-scan. They collected a dataset of 178 B-scans containing landmines, rocks, or no object in order to train and test their model.

A number of studies that applied CNNs to identify and/or classify buried objects in B-scans used traditional GPR preprocessing techniques to transform B-scans upstream of the model's input layer. Besaw and Stimac applied traditional GPR preprocessing techniques alongside a CNN-based model to differentiate between anti-tank and anti-personnel explosives in GPR B-scans [3]. They collected a dataset consisting of 367 B-scans of anti-tank explosives and 419 B-scans of anti-personnel explosives (786 B-scans in total). Dinh, Gucunski, and Duong used a similar combination of conventional image processing and a CNN-based model to detect rebar in GPR scans of bridge decks [6]. Veal et al. combined traditional image processing with a CNN to detect underground explosives in B-scans performed with a handheld GPR device [25]. They also explored the feasibility of using a Generative Adversarial Network (GAN) to transform random noise inputs into B-scans indistinguishable from real B-scans with the goal of augmenting their training set. The results of their experiments with GANs are discussed further in Section 3.2.1.

He et al. applied a CNN to the time-frequency map of GPR A-scans collected from a freeway tunnel in order to detect abnormalities in the surrounding rock [9]. The authors collected 4,800 A-scans for testing and 1,200 for training. They achieved an accuracy of 92% with a CNN, compared with 67% achieved using an SVM for the same task. Kim et al. used B- and C-scan data to construct 3D representations of subsurface regions and applied a CNN to the 3D regions in order to detect the presence of cavities, pipes, and manholes in roadways [10].

3.2 Synthetic Training Datasets for Machine Learning

A major challenge with the application of deep learning to GPR data is the lack of publicly available labeled datasets of the scale required for deep learning. In order to build a model that detects and/or classifies objects, several thousands of expert-labeled GPR scans would be needed. Each scan would need to be annotated with the presence of an object or lack thereof, the object's lateral position, depth, and other attributes related to the classification target. No such publicly available dataset of GPR scans could be located, so the prospect of using synthetic data for model training was investigated.

As outlined in Section 3.1.1, many previous applications of neural networks to GPR data have used synthetic data for training. These studies hint at the feasibility of object detection and classification in GPR scans using neural networks. However, because all of these studies used synthetic data for both training and testing, it is unclear whether a model trained in this manner will perform well on a real-world dataset.

3.2.1 Image Data Augmentation for Deep Learning

Data augmentation is the process of transforming a synthetic, small, or otherwise insufficient training dataset into an augmented dataset with the goal of improving the performance of a model trained on the dataset. A data augmentation technique is considered successful if a machine learning model trained on the augmented dataset outperforms a model of the same architecture trained on the original dataset. Shorten and Khoshgoftaar conducted a survey of image data augmentation techniques [19]. A summary of those techniques relevant to GPR B-scan data is included below.

Random Cropping Random cropping can be used to increase the size and variance of a training dataset [19]. Several randomly-cropped images can be generated from a single image. Further, in the case of an object detection task, random cropping results in the object of interest being randomly located within the image. This reduces the risk of a model "learning" that the object is always in the center of the image, for example.

Gaussian Noise Injection Gaussian noise injection is accomplished by generating noise matrices of the same size as the original images by sampling from a Gaussian distribution. Each sample from the original dataset is refined by adding to it one of the generated noise matrices [19].

Generative Adversarial Networks Veal et al. explored the feasibility of using a generative adversarial network (GAN) to transform random noise inputs into "generated" B-scans [25]. They performed a number of experiments with mixed results. Inclusion of the entire set of GAN-produced B-scans in the training set led to a decrease in model performance. The authors hypothesized that the GAN-produced dataset contained improperly labeled scans (i.e. scans with a positive label but that appeared to a human to belong to the negative class or vice-versa). They confirmed this by employing a human expert to remove the improperly labeled GAN-produced scans from the training set, which led to an increase in model performance. The authors noted "that there [was]

too much non-target imagery coming out of the GAN unfortunately that is dirtying the data – labeling samples as targets that actually look more like non-target data."

Shrivastava et al. introduce a GAN-based approach to augmenting a synthetic image dataset while retaining its labels [20]. The authors set out to create a synthetic dataset for the task of identifying gaze direction from images of human eyes. The authors created a model called *SimGAN*, which takes as input labeled, synthetic images of human eyes and transforms them into images that look indistinguishable from real human eyes but that retains their labels.

SimGAN uses a GAN consisting of a *Refiner* and a *Discriminator*. The Refiner is a model that takes synthetic images as input and attempts to transform them into realistic images (similar to *deepfakes*) while maintaining their labels. The Discriminator is a binary classifier trained to classify images as either *real* or *synthetic*. The Discriminator is trained on unlabeled real and synthetic images (images are labeled *real* or *synthetic* but are not labeled with the ultimate target of the classification task, e.g. gaze direction). A functional diagram of SimGAN is given in Figure 6.

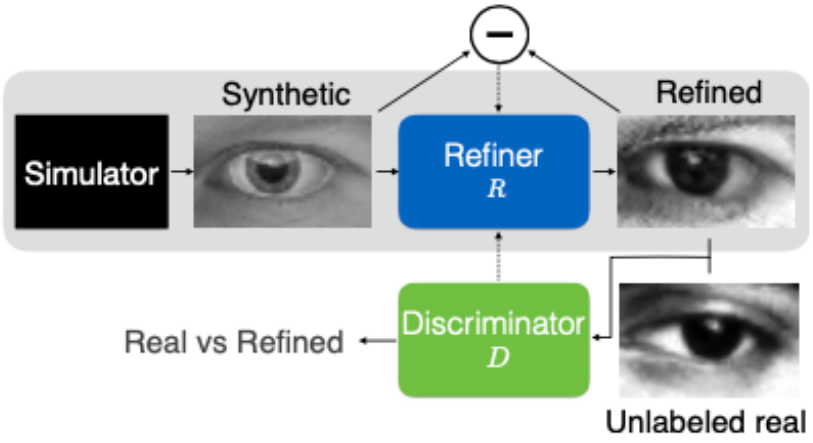

Figure 6: Functional diagram of SimGAN [20].

The Discriminator and Refiner are trained in tandem. During each round of training, a specified number of gradient descent steps are taken to train the Refiner. Next, a specified number of gradient descent steps are taken to train the Discriminator. The process repeats until training is complete. At each step in the training of the Refiner, the Refiner applies a transformation to each synthetic image in the current mini-batch. The Discriminator then classifies the Refiner's output and returns a probability indicating the likelihood that the image is real. This score is used as a component of the Refiner's loss function, driving the Refiner to generate more realistic output images at each step of training. A term is added to the Refiner's loss function to ensure that the label originally applied to the synthetic images is retained during transformation. In the case of SimGAN, a pixel-wise L1 norm was used to ensure that, on average, the pixels in the refined image don't deviate wildly from those in the synthetic image.

Once the refiner is trained, it can be used to pre-process labeled synthetic images in order to create a labeled, augmented dataset for machine learning model training. Shrivastava et al. conducted a "Visual Turing Test" in which they showed human subjects an equal number of real and refined human eye images and asked the subjects to label the images as real or fake. The humans classified

the eye images with an accuracy of 51.7%, indicating that the refiner had successfully transformed the synthetic images into images indistinguishable from real human eyes.

In the original application of SimGAN, the synthetic images of human eyes were generated using a video game engine. The authors showed that the model could also be applied to hand pose images from the NYU hand pose dataset. Simulated images are cheap to create and they can be generated based off of a provided label (e.g. this eye is looking 15° to the left). In this way, a large amount of simulated *labeled* data can be generated. The learned Refiner can then be used to pre-process the synthetic images to create a labeled dataset for a machine learning task, such as human gaze direction detection.

4 Methodology

In order to evaluate the performance of a synthetically-trained deep learning model on a real-world GPR object detection task, a deep neural network was trained on a synthetic dataset to detect the presence or absence of a cylinder within GPR B-scans and was then evaluated on a real-world test set. First, a synthetic GPR dataset was generated using physics-based simulation software. Next, a binary classification object detection model was designed and iteratively optimized. The resulting model was evaluated on a test dataset consisting of real-world GPR scans. Next, several data augmentation techniques were evaluated on the real-world test set. Finally, several additional data augmentation techniques were investigated and are proposed herein as topics for future research.

4.1 Synthetic Dataset Generation

Previous GPR modeling experiments using synthetic datasets [1][29][18][14] generated those datasets using open source software called gprMax [27]. gprMax provides a physics-based simulation to compute a transmitted electromagnetic waveform's propagation through subsurface media, reflection off of subsurface objects, and detection by a receiving antenna [8]. The software uses Maxwell's equations and a Finite-Difference Time-Domain solver to generate simulated received waveforms based on a specified transmitted waveform, scene geometry, and material properties. gprMax was used to generate a synthetic training dataset for use in the experiments described in Section 5.

A software application was developed to encapsulate the generation of each synthetic A-scan as a discrete computational task, turning synthetic dataset generation into an embarrassingly parallel workload. A framework was built to spread the tasks over multiple GPU-equipped Amazon Web Services (AWS) EC2 instances. Each simulation instance leveraged threading to parallelize the work of generating gprMax input files from a geometry specification database, running simulations, and writing the synthetic data to a data lake in an AWS S3 bucket. The gprMax GPU solver was used for even greater parallelization of the simulation itself.

A total of 669 synthetic B-scans were generated using two-dimensional scene geometries. Of these, 548 were positive (cylinder present in the scan) and 121 were negative (no cylinder present in the scan). The gprMax input specification file schema was used to define the parameters of each simulation including the geometry and material properties of the scene [26]. All synthetic B-scans were generated using a domain 3 m wide (x-direction), 3.25 m deep (y-direction), and 0.002 m thick (z-direction) with a spatial resolution of 0.02 m in the x- and z-directions and 0.04in the y-direction. The ground was modeled as a half-space located 0.25 m below the upper limit of the domain. The ground material was defined as having a relative permittivity of 6, a conductivity of 0 S/m, a relative permeability of 1, and a magnetic loss of 0 Ω/m.

The receiver and transmitter were 0.04 m apart from one another in the x-direction and were moved in tandem across the scene in increments of 0.02 m in order to generate a series of A-scans that could be combined into a B-scan depicting the scene. The height of the receiver and transmitter above the ground was varied across the simulations. All simulations used a Ricker waveform as the transmitted waveform. The center frequency of the waveform was varied across simulations. A time window of 6×10^{-8} s was used in all cases.

4.1.1 Positive Synthetic B-Scans

The 548 positive synthetic B-scans were generated with varying cylinder properties and receiver/-transmitter heights as described in Table 1 below. The cylinder depth refers to the depth in meters of the cylinder below the upper ground surface. The receiver/transmitter height refers to the height in meters of the receiver and transmitter above the upper ground surface. In all cases, the cylinder is located in the center of the scene with respect to the x-direction. In some cases, random cropping was introduced in the model input processing pipeline in order to produce scans with varying cylinder x-coordinates. Additional details related to the random cropping procedure are provided in Section 4.5.1.

Table 1: Summary statistics of synthetic positive B-scans

	Radius (m)	Wall Thickness (m)	Cylinder Depth (m)	Receiver/Transmitter Height (m)
mean	0.125512	0.018827	1.505860	0.103473
std	0.069494	0.010424	0.288928	0.055205
min	0.005133	0.000770	1.004589	0.010832
25%	0.063262	0.009489	1.250548	0.053235
50%	0.129312	0.019397	1.508990	0.103245
75%	0.184411	0.027662	1.761548	0.150415
max	0.249353	0.037403	1.999050	0.199407

Two different cylinder wall and fill materials were used. The cylinder wall was composed of either metal (modeled using gprMax's built-in perfect electrical conductor material) or PVC (modeled using a custom material with relative permittivity 4, conductivity 10^{-6} S/m, relative permeability 1, and magnetic loss 0 Ω/m). The cylinder was filled with either air (relative permittivity 0, conductivity 0 S/m, relative permeability 0, and magnetic loss 0 Ω/m) or water (relative permittivity 80, conductivity 4.194×10^{-6} S/m, relative permeability 1, and magnetic loss 100 Ω/m). Table 2 below outlines the number of synthetic positive scans generated in each cylinder material configuration.

Table 2: Cylinder material summary of synthetic positive B-scans

Cylinder Material	Cylinder Fill Material	Sample Count
PVC	Water	148
	Air	133
Metal	Water	129
	Air	138

Finally, nine different waveform center frequencies were used as outlined in Table 3. The waveform frequencies were selected to reflect the frequencies present in the real-world test dataset.

Table 3: Center frequency count of synthetic positive B-scans

Center Frequency (MHz)	Sample Count
800	67
500	63
400	59
600	55
270	54
350	51
200	32
900	28
250	5

4.1.2 Negative Synthetic B-Scans

The 121 negative synthetic B-scans were generated with varying receiver/transmitter height as described in Table 4 below. Further, surface water depth was introduced as an additional parameter to increase the variability among the negative simulated B-scans. Of the 121 negative simulated B-scans, 62 did not include surface water and 59 included surface water with a depth of 1 mm. The count of negative B-scans associated with each waveform center frequency is given in Table 5.

Table 4: Summary statistics of synthetic negative B-scans

	Receiver/Transmitter Height (m)
mean	0.106579
std	0.055388
min	0.010522
25%	0.051936
50%	0.107087
75%	0.153914
max	0.197956

Table 5: Center frequency count of synthetic negative B-scans

Center Frequency (MHz)	Sample Count
500	24
270	20
800	16
350	15
600	15
400	13
200	11
900	7

4.2 Model Selection

The task selected for model evaluation was to detect the presence or absence of a cylinder located within the field of a GPR B-scan. Thus, the object detection task was framed as a binary classification problem. Each sample in the input is a 2-D matrix representing a B-scan. Each sample was assigned a label of 0 (indicating no cylinder present in B-scan) or 1 (indicating cylinder present in B-scan). Figure 7 shows examples of negative and positive samples.

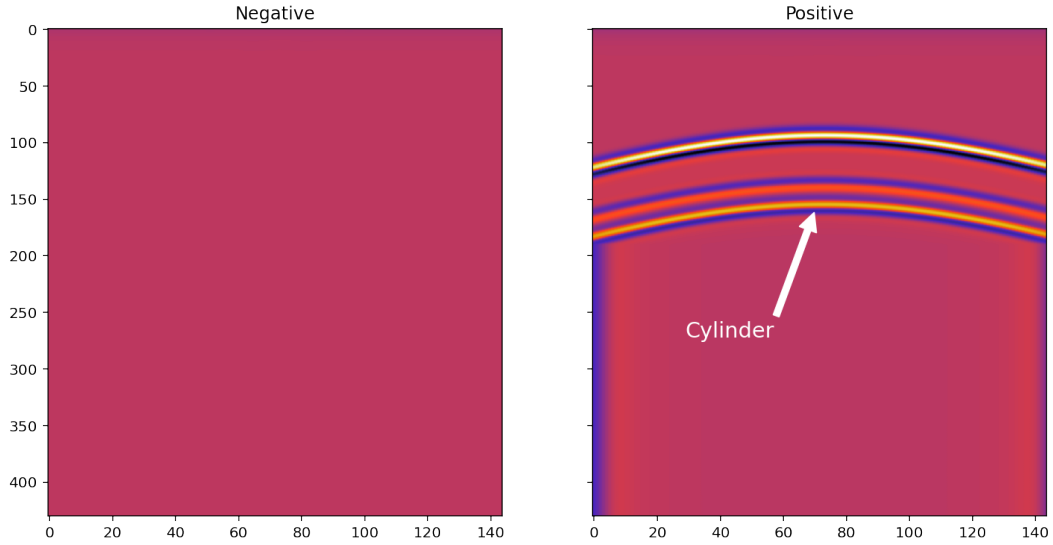

Figure 7: Sample synthetic B-scans (air-ground interface reflection cropped for clarity)

Several classes of neural network were assessed and ultimately a convolutional neural network (CNN) was chosen as the base model architecture. The model architecture was tuned by testing several combinations of hyperparameters such as number of layers, number of filters in each layer, number of nodes per dense layer, kernel size, dropout rate, and l2 regularization α. Each combination of hyperparameters was tested on the task outlined as Experiment 2 in Section 5. Once an f1-score of 1 was attained, further optimization was deemed unnecessary. If a larger number of real, labeled GPR B-scans were available, the set could be partitioned into a test set and a second validation set. The model architecture could then be tuned using the real validation set and evaluated on the real test set. This is left as a topic for future work.

Attempts were made to test more complex model architectures than the one eventually selected. This included deeper models with additional blocks of layers, increased numbers of kernels in convolutional layers, and increased numbers of nodes in dense layers. Unfortunately, computational resource constraints limited the complexity of model that could be trained. Design and evaluation of more complex models is left as an additional topic for future work.

Figure 8 provides an overview of the final model architecture. The tensorflow model definition code is provided in Appendix A. Several regularization techniques were employed including batch normalization, dropout, and L2 regularization.

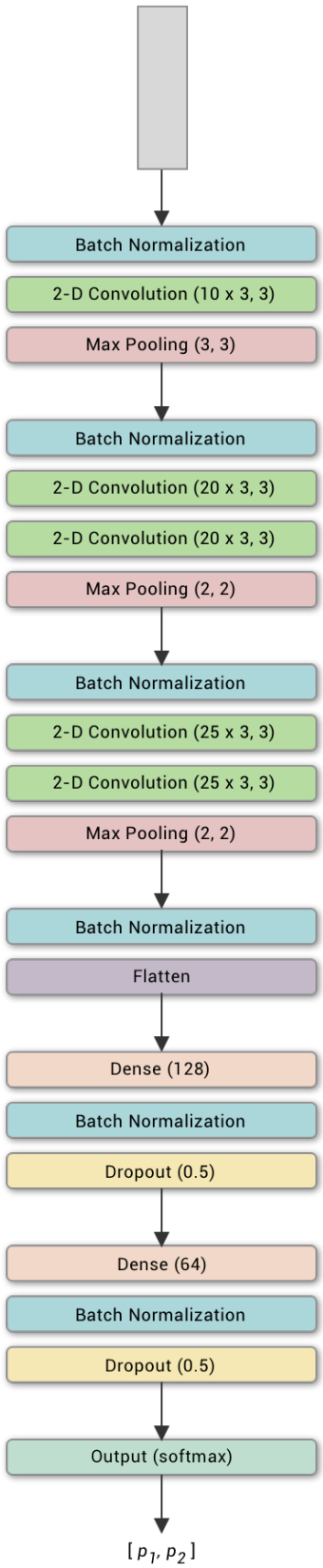

Figure 8: Binary classification CNN model architecture

4.3 Training, Validation, and Test Datasets

Training & Validation Sets The training and validation datasets were generated in the same manner during each experiment. They were initially generated as one large dataset. 90% of the samples were assigned at random to the training set and the remaining 10% were assigned to the validation set. The training and validation sets were based on a synthetic dataset for all experiments but in some cases data augmentation techniques were applied to both. Section 5 provides additional details regarding the size, makeup, and augmentation of the training and validation sets used in each experiment.

Test Set The test dataset was composed of real GPR B-scans collected from the IFSTTAR Geophysical Test Site in Nantes, France and compiled into an open database by Derobert and Pajewski [5]. The data provided in the open database were manually cropped, segmented, and labeled as positive (contains cylinder) or negative (does not contain cylinder). The detailed layout diagrams of the test site provided by the authors were referenced to ensure accurate labeling of the data. Each sample in the test dataset is a single B-scan represented as a 480 $\times$ 100 matrix (120 ns $\times$ 2 m). The dataset originally contained 880 total samples: 337 positive and 543 negative. The test set was then balanced by removing negative samples at random to obtain a final test dataset of 674 samples: 337 positive and 337 negative. Figure 9 below shows examples of positive and negative samples taken from the test set.

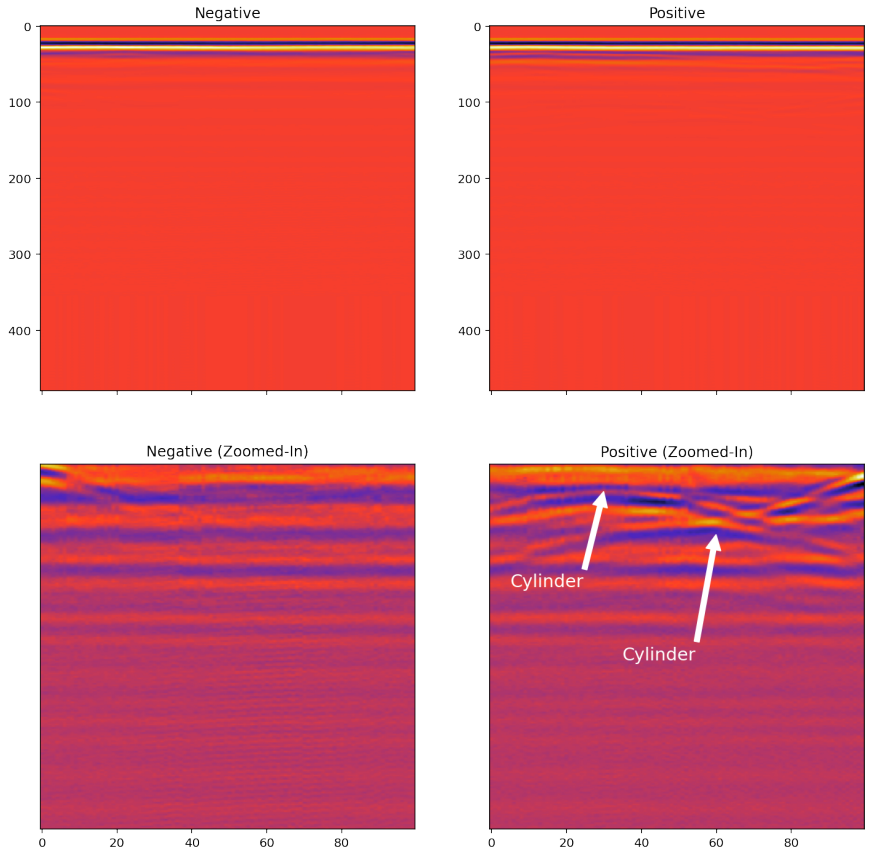

Figure 9: Sample B-scans from test set

4.4 Data Pre-Processing Pipeline

A pre-processing pipeline was constructed in order to ensure that all samples in the training, validation, and test datasets are of a consistent size, width (in meters), and depth (in ns). Due to the large size of the training dataset, a batch processing scheme was designed in order to load the dataset into memory and perform pre-processing incrementally. The data was loaded and pre-processed in ten batches. The first batch was used as the validation set and the remaining nine batches were used as the training set. The model was trained on all nine training batches in sequence in all experiments performed. The samples were randomly assigned to the batches. The pre-processing pipeline consists of the following stages:

Merging In the case of the synthetic training and validation sets, the data were stored in AWS S3 as many HDF5 files, each containing a single A-scan. The first stage of the pipeline merges all synthetic A-scans from each simulated scene into a single synthetic B-scan. The merging process is computationally expensive due to the large amount of network I/O required. Thus, merged scans were written back to the S3 data lake to be reused across multiple modeling experiments.

Distance Domain Resampling The input dataset was resampled in the distance domain so that all samples in the resulting dataset contain 50 scans per meter. This ensures that all samples in the input represent objects using the same physical scale. The synthetic B-scans were generated with 50 scans per meter so distance domain resampling was not required for the synthetic training and validation sets. Two different resampling strategies were assessed for distance domain resampling: last value and linear interpolation. Linear interpolation yielded the best results based on a visual inspection, but the last value method is far more computationally efficient. The last value method was chosen in order to reduce computational costs.

Time Domain Resampling and Padding The input dataset was resampled in the time domain so that each A-scan (i.e. each column of each B-scan) represented 120 ns of time with 4 samples per ns. Each column of the input matrix was first resampled and then either truncated or padded as required to obtain a vector of length 480 (120 ns $\cdot$ 4 samples/ns). Three resampling strategies were assessed for time domain resampling: Fourier resampling, linear interpolation, and last value. The Fourier resampling method yielded the best results based on a visual inspection, but the last value method was ultimately selected due to its significantly lower time complexity.

Data Augmentation Data augmentation was performed as the final stage of the pipeline. The various techniques used are introduced in Section 4.5 and their application is described in Section 5.

4.5 Data Augmentation Techniques

4.5.1 Random Cropping

Random cropping, as described in Section 3.2.1, was performed by selecting a number, n, of randomly generated windows of width 100 columns each from each sample in the training and validation sets. In the general case, the number of samples in both the training and validation sets increases by a factor of n when random cropping is applied in this way. In the case where $n = 1$, the size of the training and validation sets is unchanged. For $n > 1$, this method acts as a form of bootstrapping by increasing the size of the training and validation sets in a random manner. The value of n is a hyperparameter that can be tuned to optimize performance on a real-world dataset.

4.5.2 Negative Augmentation

Negative augmentation is a data augmentation technique devised to increase the size of the training dataset by introducing real negative samples from an auxiliary dataset. A large set of real negative samples is sampled at random without replacement until the desired number, n_{real}, of real negatives has been acquired. These real negative samples are then concatenated with the synthetic dataset. The value of n_{real} is a hyperparameter that can be tuned to optimize model performance. n_{real} may be chosen in such a way that allows negative augmentation to be used as a means of balancing the training and validation datasets.

In this particular application, negative augmentation is an attractive choice since real negative samples are widely available, but real positive samples are not. The US Geological Survey (USGS) has publicly released several large, unlabeled datasets containing overwhelmingly negative samples. One such dataset was collected from glaciers in Alaska as part of a project that measures the change in glacial mass over time [7]. The dataset collected from the Gulkana glacier was selected at random from among those available and downloaded for use in real negative injection [13]. The Gulkana glacier dataset was partitioned by the original contributors into two sets: DATA01 and DATA02. DATA01 was used for negative augmentation and DATA02 was reserved for real noise generation.

4.5.3 Real Noise

Several data augmentation techniques introduce noise with the goal of reducing over-fitting on the training dataset [19]. In the case of a synthetic GPR dataset, the samples are devoid of noise because the simulation that generated them makes certain simplifying assumptions. Further, the synthetic scans contain only a single target, in this case a cylinder, whereas real GPR B-scans typically include many small point scatterers such as small rock. Real B-scans also likely include radar reflections from interfaces between different types of soil or rocks. Finally, real B-scans include some amount of Gaussian noise. Collectively, these sources of noise are referred to as *clutter*. Figure 10 shows examples of real and synthetic positive and negative B-scans. Note the presence of clutter in both of the real scans. Note also that the cylinder is difficult to see in the real positive scan at this zoom level (Figure 9 shows the cylinder more clearly).

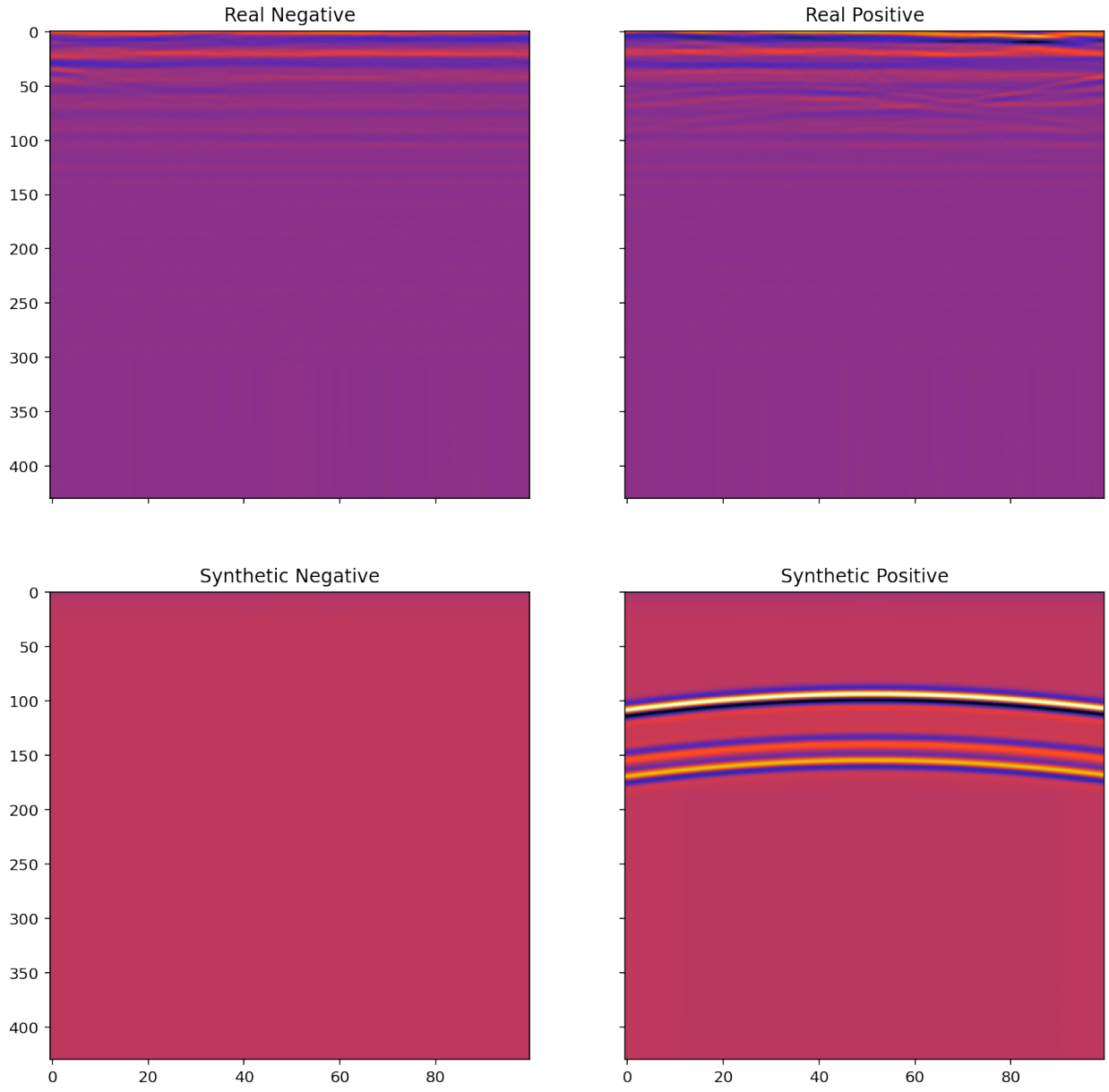

Figure 10: Noise comparison of real and synthetic B-scans (air-ground interface reflection cropped for clarity)

In many applications, sampling from a Gaussian distribution to obtain a noise matrix which can be applied to samples in the test set yields good results [19]. However, Gaussian noise has no spatial dependency. One can see by examining Figure 10 that the noise present in the real B-scans is spatially dependent. This observation led to the idea of a data augmentation technique whereby noise is extracted from real B-scans and applied to synthetic ones.

Further examination of real A-scans such as that shown in Figure 11 indicates that the noise is typically of a higher frequency than the true signal. Thus, a Butterworth highpass filter with cutoff frequency f was applied to real A-scans in order to extract noise vectors. Before applying the

filter, the real A-scans were first scaled so that each had a minimum value of 0 and maximum value of 1. The second chart in Figure 11 shows the results of applying a highpass filter with $f = 5$ to the real A-scan. Note that the amplitude of the noise signal is increased where the real signal amplitude is increased. This occurs at the air-ground interface, which is not in the same location in all B-scans. In order to compensate for this fact, the magnitude of the noise signal was clipped at some percentile p. In order to perform the clipping, the absolute value of the filtered scan was taken and the pth percentile, a_p, was then computed. The signal was then clipped at $\pm a_p$. Values that were out of this range on the high end were imputed with a_p and values out of range on the low end were imputed with $-a_p$. The bottom chart of Figure 11 shows the filtered scan clipped with $p = 99$. The filter parameters f and p are hyperparameters that can be tuned to improve the performance of the model on a real-world dataset.

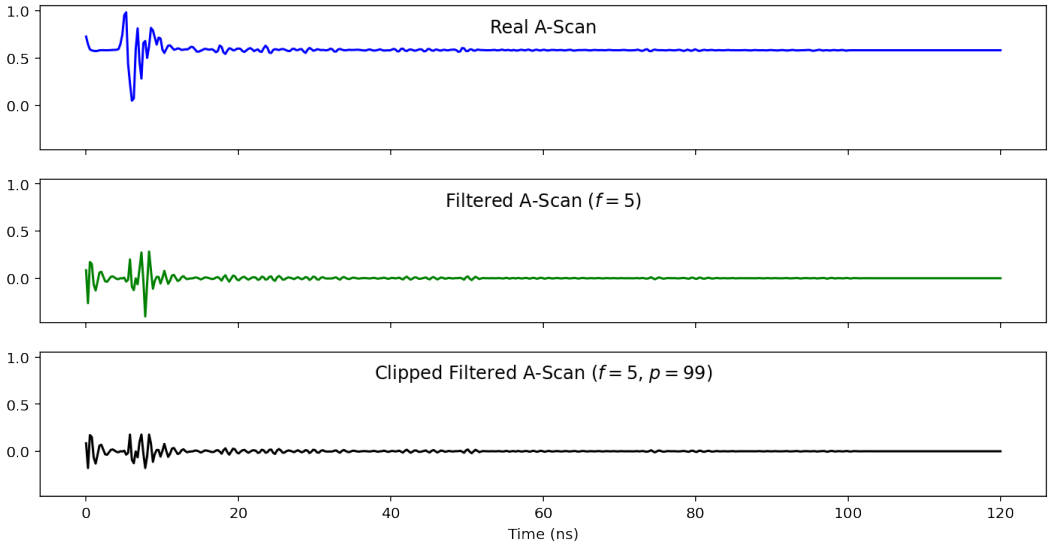

Figure 11: A-scan noise generation process

The noise vector was then applied to a scaled synthetic A-scan by adding them together to get a "noised" A-scan. Figure 12 shows an example of a real A-scan from which noise was extracted, a noised A-scan after applying that noise, and the synthetic A-scan to which the noise was applied.

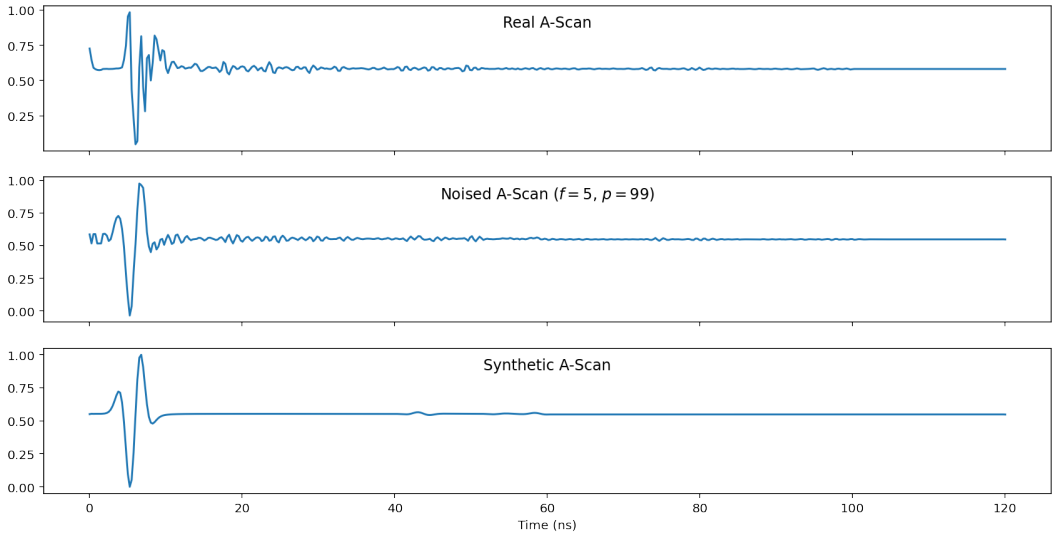

Figure 12: A-scan noise application process

This procedure was extended to B-scans by first randomly selecting a real B-scan of the same shape as the synthetic B-scan and repeating the noise extraction procedures for each of its A-scans (columns) to obtain a noise matrix. This noise matrix was then added to a synthetic B-scan in order to obtain a noised B-scan. Figure 13 provides a B-scan analogy to Figure 12. Note that the label has been retained; the two hyperbolic reflections in the synthetic B-scan, which represent the radar signature of the buried cylinder, are still visible in the noised B-scan.

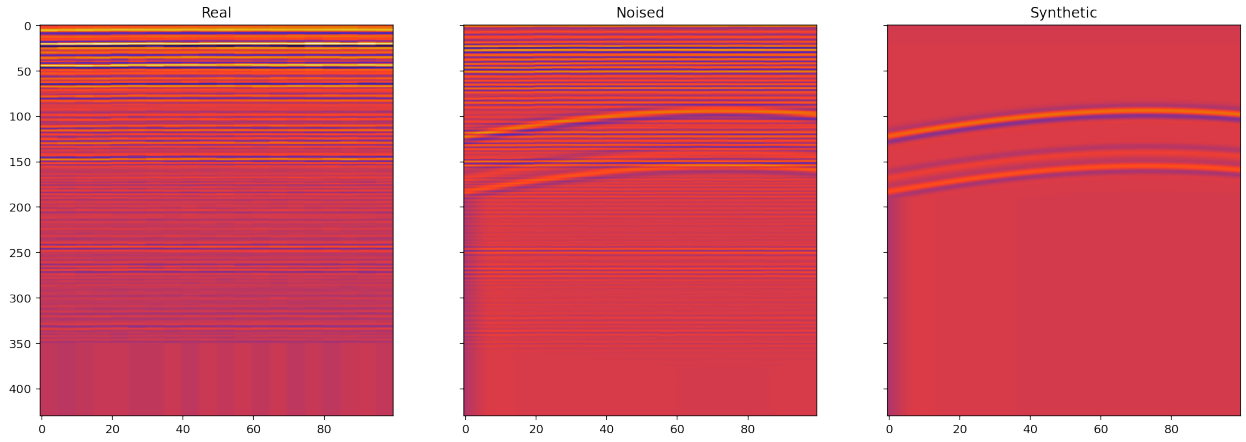

Figure 13: B-scan noise application process (air-ground interface reflection cropped for clarity)

As noted above, the DATA02 subset of the Gulkana glacier dataset [13] was used as the source of real noise for all real noise experiments performed.

4.6 Model Training

A consistent model training procedure was applied during all experiments detailed in Section 5. The Adam (adaptive moment estimation) optimizer was used with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. As described in Section 4.4, the model was trained on nine batches of data in succession during each experiment. The model was trained on each batch for up to 30 epochs with early stopping implemented to prevent over-fitting. Early stopping was configured to stop training on a particular batch and roll back to the best weights obtained thus far on that batch if validation accuracy had not improved over the past ten epochs.

4.7 Model Evaluation

In each experiment, the model was evaluated using three metrics: *precision*, *recall*, and *f1-score*. Recall measures the percentage of actual positive samples that were correctly detected by the model. Recall is given by

$$r = \frac{TP}{TP + FN} \tag{3}$$

where TP is the number of true positives (the actual class and the labeled class are both positive) and FN is the number of false negatives (the actual class is positive and the labeled class is negative).

Precision measures the percentage of actual positives among those samples identified by the model as positive. Precision is given by

$$p = \frac{TP}{TP + FP} \tag{4}$$

where TP is as defined above and FP is the number of false positives (the actual class is negative but the labeled class is positive).

The f1-score is the harmonic mean of precision and recall. Traditionally, the harmonic mean has been used because it punishes a system that increases one of precision or recall at the expense of the other. The f1-score is given by

$$\text{f1-score} = \frac{1}{\frac{1}{r} + \frac{1}{p}} = \frac{2pr}{r + p} \tag{5}$$

5 Results and Discussion

Table 6 below summarizes the results of the experiments performed. The same test set was used for all experiments. Details regarding the source, labeling, and descriptive statistics of the test set are provided in Section 4.3. Additional details about each experiment are included in the following subsections.

Table 6: Data augmentation experiment results

Experiment	1	2	3	4	5	6	7	8
Augmentation	-	C	N	R	C+N	C+R	N+R	C+N+R
Training Set Size	968	9680	968	968	9680	9680	968	9680
	+ 484	+ 4840	+ 484	+ 484	+ 4840	+ 4840	+ 484	+ 4840
	- 484	- 4840	- 484	- 484	- 4840	- 4840	- 484	- 4840
Validation Set Size	116	1160	116	116	1160	1160	116	1160
	+ 58	+ 580	+ 58	+ 58	+ 580	+ 580	+ 58	+ 580
	- 58	- 580	- 58	- 58	- 580	- 580	- 58	- 580
Test Set Size	674	674	674	674	674	674	674	674
	+ 337	+ 337	+ 337	+ 337	+ 337	+ 337	+ 337	+ 337
	- 337	- 337	- 337	- 337	- 337	- 337	- 337	- 337
Val f-1	0.83	1.00	0.97	0.94	0.93	1.00	0.88	0.93
Val precision	0.71	1.00	0.97	0.90	0.87	1.00	0.87	0.87
Val recall	0.98	1.00	0.98	0.98	1.00	1.00	0.90	1.00
Test f-1	0.64	0.50	0.37	0.37	0.29	0.22	0.43	0.15
Test precision	0.49	0.72	0.55	0.54	0.60	0.79	0.52	0.45
Test recall	0.93	0.38	0.28	0.28	0.19	0.12	0.37	0.09

Key: **C**: Random Cropping, **N**: Negative Augmentation, **R**: Real Noise

The test results shown in the bottom section of Table 6 indicate that none of the evaluated data augmentation techniques increased the performance of the model on the real-world test set. In fact, all evaluated data augmentation techniques resulted in lower model performance than the control experiment (1) in which no data augmentation techniques were applied. Thus, none of these techniques, parameterized as described in the following sections, increases the real-world performance of a synthetically-trained GPR object detection model. Optimization of these techniques' hyperparameters is left as a topic for future research.

Comparison of the bottom two sections of Table 6 shows that model performance measured on a synthetic dataset is not an indication of how well the model will perform on real-world data. Most of the experiments detailed above yielded models that perform very well on the synthetic validation set. However, these models performed worse than random guessing when evaluated on the real-world test set.

5.1 No Data Augmentation

The training and validation sets were balanced so that they included an equal number of positive and negative scans. Since the synthetic dataset contained more positive than negative samples, bootstrapping was used to increase the number of negative samples in the dataset. The existing negative samples were sampled randomly with replacement until a sufficient number of additional negative samples were obtained. These negative samples were then concatenated with the existing dataset. Table 7 shows the results of this experiment. In this case, the model overwhelmingly classifies samples in the test set as positive, obtaining a recall of 0.93 but a precision of only 0.49.

Table 7: Experiment 1 results

Validation

		Predicted	
		T	**F**
Actual	**T**	57	1
	F	23	35

Test

		Predicted	
		T	**F**
Actual	**T**	312	25
	F	329	8

5.2 Random Cropping

Random cropping, as described in Section 4.5.1 was applied to the training and validation sets with $n = 10$. The training and validation sets were balanced after the random cropping was applied using the same bootstrapping method as in Experiment 1. The results are given in Table 8.

Table 8: Experiment 2 results

Validation

Actual \ Predicted	T	F
T	580	0
F	0	580

Test

Actual \ Predicted	T	F
T	128	209
F	51	286

5.3 Negative Augmentation

The training and validation sets were balanced via negative augmentation. That is, real negative samples were randomly selected from a large dataset of real negatives until the number of positive and negative samples in the training and validation sets were equal. These real negative samples were then concatenated with the synthetic datasets. The results of this experiment are given in Table 9.

Table 9: Experiment 3 results

Validation

Actual \ Predicted	T	F
T	57	1
F	2	56

Test

Actual \ Predicted	T	F
T	94	243
F	77	260

5.4 Real Noise

Real noise was applied to all samples in the training and validation sets as described in Section 4.5.3. The real noise generation filter was constructed with a cutoff frequency $f = 5$ and clipping percentile $p = 99$. The results of this experiment are given in Table 10.

Table 10: Experiment 4 results

Validation

Actual \ Predicted	T	F
T	57	1
F	6	52

Test

Actual \ Predicted	T	F
T	94	243
F	79	258

5.5 Random Cropping + Negative Augmentation

Random cropping and real negative injection were applied together to the training and validation sets. Random cropping with $n = 10$ was first applied to the training and validation sets and both were then balanced via negative augmentation. The results are given in Table 11.

Table 11: Experiment 5 results

Validation

Actual \ Predicted	T	F
T	580	0
F	90	490

Test

Actual \ Predicted	T	F
T	65	272
F	43	294

5.6 Random Cropping + Real Noise

Random cropping and real noise were applied together to the training and validation sets. Random cropping with $n = 10$ was first applied to the synthetic dataset. The dataset was then balanced via bootstrapping. Next, real noise was applied to all samples. The real noise generation filter was constructed with a cutoff frequency $f = 5$ and clipping percentile $p = 99$. The results of this experiment are given in Table 12.

Table 12: Experiment 6 results

Validation

		Predicted T	Predicted F
Actual	T	580	0
	F	0	580

Test

		Predicted T	Predicted F
Actual	T	42	295
	F	11	326

5.7 Negative Augmentation + Real Noise

Negative augmentation and real noise were applied together to the training and validation sets. Real noise was first applied to the synthetic training and validation sets. The real noise generation filter was constructed with a cutoff frequency $f = 5$ and clipping percentile $p = 99$. The training and validation sets were then balanced via negative augmentation. The results of this experiment are given in Table 13.

Table 13: Experiment 7 results

Validation

		Predicted T	Predicted F
Actual	T	52	6
	F	8	50

Test

		Predicted T	Predicted F
Actual	T	124	213
	F	115	222

5.8 Random Cropping + Negative Augmentation + Real Noise

All three data augmentation techniques were applied to the training and validation sets. Random cropping with $n = 10$ was first applied to the synthetic dataset. Real noise was then applied to all synthetic samples. The real noise generation filter was constructed with a cutoff frequency $f = 5$ and clipping percentile $p = 99$. Finally, the training and validation datasets were balanced via negative augmentation. The results of this experiment are given in Table 14.

Table 14: Experiment 8 results

Validation

Actual \ Predicted	T	F
T	580	0
F	90	490

Test

Actual \ Predicted	T	F
T	31	306
F	38	299

6 Conclusion and Future Work

The application of deep learning for object detection in GPR data is motivated by the cost and scarcity of trained experts able to interpret GPR scans. Many prior applications of deep learning to GPR data have used synthetic datasets for both training and evaluation due to the lack of available real-world labeled datasets of the size necessary for machine learning. These prior studies have produced models that perform well when evaluated on a synthetic test set but none provide model evaluation results generated using a real-world test set.

This work expand upon these prior studies by evaluating the real-world performance of a synthetically-trained neural network model for object detection in GPR images. Synthetic GPR datasets for training and validation were generated using gprMax and the real-world GPR test dataset was obtained from an open database and labeled manually. A binary classification convolutional neural network (CNN) for object detection was then trained on the synthetic training set and evaluated on both the synthetic validation set and the real-world test set. The results provided in Section 5 demonstrate that the model metrics provided in studies using synthetic test sets are not necessarily indicative of how the models will perform on real-world GPR data. Future work should revisit these prior studies with the goal of evaluating the proposed models on a real-world test dataset.

Data augmentation techniques have been successfully applied to improve the performance of image recognition models applied to photographic images. This work proposes and evaluates three data augmentation techniques for GPR B-scans: (1) random cropping, (2) negative augmentation, and (3) real noise application. The goal of these data augmentation techniques is to improve the real-world performance of a synthetically-trained GPR object detection model. All possible subsets of the three techniques were evaluated, including a control experiment in which no data augmentation techniques were applied. None of the proposed techniques or combinations thereof increased the performance of the synthetically-trained model on a real-world dataset. Each of the data augmentation techniques proposed in Section 4.5 exposes hyperparameters that may be tuned to optimize model performance on real-world data. It remains to be shown whether hyperparameter tuning will improve the performance of these techniques; this is left as a topic for future research.

A final suggestion for future work is extension and evaluation of the SimGAN framework introduced in Section 3.2.1 to GPR data. Veal et al. performed data augmentation experiments using a GAN to transform random noise matrices into "generated" B-scans [25]. They noted that the content of the generated B-scans often did not reflect their assigned labels. Application of the SimGAN framework to produce "refined" GPR scans from labeled synthetic scans rather than from random noise has the potential to alleviate the problem noted by the authors. An experiment similar to those performed as part of this study could be conducted to determine if SimGAN-based data augmentation has the potential to improve the real-world performance of a synthetically-trained GPR object detection model.

References

[1] Maha Almaimani et al. "Classifying GPR Images Using Convolutional Neural Networks". en. In: *Proceedings of the 11th EAI International Conference on Mobile Multimedia Communications*. Qingdao, People's Republic of China: EAI, 2018. ISBN: 978-1-63190-164-5. DOI: 10.4108/eai.21-6-2018.2276629. URL: http://eudl.eu/doi/10.4108/eai.21-6-2018.2276629 (visited on 01/28/2021).

[2] J. K. Alvarez and S. Kodagoda. "Application of deep learning image-to-image transformation networks to GPR radargrams for sub-surface imaging in infrastructure monitoring". In: *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. ISSN: 2158-2297. May 2018, pp. 611–616. DOI: 10.1109/ICIEA.2018.8397788.

[3] Lance E. Besaw and Philip J. Stimac. "Deep convolutional neural networks for classifying GPR B-scans". In: *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XX*. Vol. 9454. International Society for Optics and Photonics, May 2015, p. 945413. DOI: 10.1117/12.2176250. URL: https://www-spiedigitallibrary-org.ezproxy.neu.edu/conference-proceedings-of-spie/9454/945413/Deep-convolutional-neural-networks-for-classifying-GPR-B-scans/10.1117/12.2176250.short (visited on 03/09/2021).

[4] David Daniels. *Ground Penetrating Radar*. English. 2nd ed. Vol. 15. IET Radar, Sonar, Navigation and Avionics. London, United Kingdom: The Institution of Engineering and Technology, 2007. ISBN: 978-0-86341-360-5.

[5] Xavier Dérobert and Lara Pajewski. "TU1208 Open Database of Radargrams: The Dataset of the IFSTTAR Geophysical Test Site". en. In: *Remote Sensing* 10.4 (Apr. 2018). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 530. DOI: 10.3390/rs10040530. URL: https://www.mdpi.com/2072-4292/10/4/530 (visited on 02/18/2021).

[6] Kien Dinh, Nenad Gucunski, and Trung H. Duong. "An algorithm for automatic localization and detection of rebars from GPR data of concrete bridge decks". en. In: *Automation in Construction* 89 (May 2018), pp. 292–298. ISSN: 0926-5805. DOI: 10.1016/j.autcon.2018.02.017. URL: https://www.sciencedirect.com/science/article/pii/S0926580517307136 (visited on 03/09/2021).

[7] *Glaciers and Climate: GPR Data*. URL: https://www2.usgs.gov/landresources/lcs/glacierstudies/gpr.asp (visited on 03/09/2021).

[8] *gprMax: Electromagnetic simulation software*. URL: https://www.gprmax.com/about.shtml (visited on 02/12/2021).

[9] Yu-yao He et al. "An interpretation model of GPR point data in tunnel geological prediction". In: *Eighth International Conference on Graphic and Image Processing (ICGIP 2016)*. Vol. 10225. International Society for Optics and Photonics, Feb. 2017, 102252J. DOI: 10.1117/12.2266226. URL: https://www-spiedigitallibrary-org.ezproxy.neu.edu/conference-proceedings-of-spie/10225/102252J/An-interpretation-model-of-GPR-point-data-in-tunnel-geological/10.1117/12.2266226.short (visited on 03/09/2021).

[10] Namgyu Kim et al. "A novel 3D GPR image arrangement for deep learning-based underground object classification". In: *International Journal of Pavement Engineering* 0.0 (Aug. 2019). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10298436.2019.1645846, pp. 1–12. ISSN: 1029-8436. DOI: 10.1080/10298436.2019.1645846. URL: https://doi.org/10.1080/10298436.2019.1645846 (visited on 03/09/2021).

[11] Shuai Li et al. "Estimating Features of Underground Utilities: Hybrid GPR/GPS Approach". en. In: *Journal of Computing in Civil Engineering* 30.1 (Jan. 2016), p. 04014108. ISSN: 0887-3801, 1943-5487. DOI: 10.1061/(ASCE)CP.1943-5487.0000443. URL: http://ascelibrary.org/doi/10.1061/%28ASCE%29CP.1943-5487.0000443 (visited on 02/11/2021).

[12] Geert Litjens et al. "A Survey on Deep Learning in Medical Image Analysis". In: *Medical Image Analysis* 42 (Dec. 2017). arXiv: 1702.05747, pp. 60–88. ISSN: 13618415. DOI: 10.1016/j.media.2017.07.005. URL: http://arxiv.org/abs/1702.05747 (visited on 03/08/2021).

[13] Daniel McGrath and et. a; *Raw Ground Penetrating Radar Data, Gulkana Glacier, Alaska, ver. 2.1, September 2018: U.S. Geological Survey data release*. Sept. 2018. URL: https://doi.org/10.5066/F7M043G7.

[14] Umut Ozkaya and Levent Seyfi. "Deep dictionary learning application in GPR B-scan images". en. In: *Signal, Image and Video Processing* 12.8 (Nov. 2018), pp. 1567–1575. ISSN: 1863-1711. DOI: 10.1007/s11760-018-1313-x. URL: https://doi.org/10.1007/s11760-018-1313-x (visited on 03/09/2021).

[15] *PaveScan RDM System — Ground Penetrating Radar Equipment — GSSI*. en. URL: https://www.geophysical.com/products/pavescan-rdm (visited on 01/29/2021).

[16] M. Pham and S. Lefèvre. "Buried Object Detection from B-Scan Ground Penetrating Radar Data Using Faster-RCNN". In: *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. ISSN: 2153-7003. July 2018, pp. 6804–6807. DOI: 10.1109/IGARSS.2018.8517683.

[17] F. Picetti et al. "Convolutional Autoencoder for Landmine Detection on GPR Scans". In: *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*. July 2018, pp. 1–4. DOI: 10.1109/TSP.2018.8441206.

[18] F. Ponti et al. "Deep Learning for Applications to Ground Penetrating Radar and Electromagnetic Diagnostic". In: *2019 PhotonIcs Electromagnetics Research Symposium - Spring (PIERS-Spring)*. ISSN: 1559-9450. June 2019, pp. 547–551. DOI: 10.1109/PIERS-Spring46901.2019.9017753.

[19] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (July 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0 (visited on 07/07/2021).

[20] Ashish Shrivastava et al. "Learning from Simulated and Unsupervised Images through Adversarial Training". In: *arXiv:1612.07828 [cs]* (July 2017). arXiv: 1612.07828. URL: http://arxiv.org/abs/1612.07828 (visited on 02/08/2021).

[21] J. Sonoda and T. Kimoto. "Object Identification form GPR Images by Deep Learning". In: *2018 Asia-Pacific Microwave Conference (APMC)*. Nov. 2018, pp. 1298–1300. DOI: 10.23919/APMC.2018.8617556.

[22] Zheng Tong, Jie Gao, and Haitao Zhang. "Innovative method for recognizing subgrade defects based on a convolutional neural network". en. In: *Construction and Building Materials* 169 (Apr. 2018), pp. 69–82. ISSN: 0950-0618. DOI: 10.1016/j.conbuildmat.2018.02.081. URL: https://www.sciencedirect.com/science/article/pii/S0950061818303246 (visited on 03/09/2021).

[23] Zheng Tong, Jie Gao, and Haitao Zhang. "Recognition, location, measurement, and 3D reconstruction of concealed cracks using convolutional neural networks". en. In: *Construction and Building Materials* 146 (Aug. 2017), pp. 775–787. ISSN: 0950-0618. DOI: 10.1016/j.conbuildmat.2017.04.097. URL: https://www.sciencedirect.com/science/article/pii/S095006181730747X (visited on 03/09/2021).

[24] Zheng Tong et al. "Pavement-distress detection using ground-penetrating radar and network in networks". en. In: *Construction and Building Materials* 233 (Feb. 2020), p. 117352. ISSN: 0950-0618. DOI: 10.1016/j.conbuildmat.2019.117352. URL: https://www.sciencedirect.com/science/article/pii/S0950061819328041 (visited on 03/09/2021).

[25] Charlie Veal et al. "Generative adversarial networks for ground penetrating radar in hand held explosive hazard detection". In: *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXIII*. Vol. 10628. International Society for Optics and Photonics, Apr. 2018, 106280T. DOI: 10.1117/12.2307261. URL: https://www-spiedigitallibrary-org.ezproxy.neu.edu/conference-proceedings-of-spie/10628/106280T/Generative-adversarial-networks-for-ground-penetrating-radar-in-hand-held/10.1117/12.2307261.short (visited on 03/09/2021).

[26] Craig Warren and Antonios Giannopoulos. *Input file commands*. URL: http://docs.gprmax.com/en/latest/input.html (visited on 07/06/2021).

[27] Craig Warren, Antonios Giannopoulos, and Iraklis Giannakis. "gprMax: Open source software to simulate electromagnetic wave propagation for Ground Penetrating Radar". en. In: *Computer Physics Communications* 209 (Dec. 2016), pp. 163–170. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2016.08.020. URL: https://www.sciencedirect.com/science/article/pii/S0010465516302533 (visited on 07/06/2021).

[28] Chenxi Yuan et al. "GPR Signature Detection and Decomposition for Mapping Buried Utilities with Complex Spatial Configuration". en. In: *Journal of Computing in Civil Engineering* 32.4 (July 2018), p. 04018026. ISSN: 0887-3801, 1943-5487. DOI: 10.1061/(ASCE)CP.1943-5487.0000764. URL: http://ascelibrary.org/doi/10.1061/%28ASCE%29CP.1943-5487.0000764 (visited on 01/28/2021).

[29] Yu Zhang, Dryver Huston, and Tian Xia. "Underground object characterization based on neural networks for ground penetrating radar data". en. In: ed. by Tzuyang Yu et al. Las Vegas, Nevada, United States, Apr. 2016, p. 980403. DOI: 10.1117/12.2219345. URL: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2219345 (visited on 01/29/2021).

Appendix A: Model Definition

```
from tensorflow import keras

alpha = 0.08

model = keras.models.Sequential([
  keras.layers.BatchNormalization(),
  keras.layers.Conv2D(filters=10, kernel_size=(3, 3), strides=(1, 1),
    kernel_regularizer=l2(alpha),
  activation='relu', padding='same'),
  keras.layers.MaxPool2D(pool_size=(3, 3), strides=(1, 1)),
  keras.layers.BatchNormalization(),
  keras.layers.Conv2D(filters=20, kernel_size=(3, 3), strides=(1, 1),
    kernel_regularizer=l2(alpha),
  activation='relu', padding='same'),
  keras.layers.Conv2D(filters=20, kernel_size=(3, 3), strides=(1, 1),
    kernel_regularizer=l2(alpha),
  activation='relu', padding='same'),
  keras.layers.MaxPool2D(pool_size=(2, 2), strides=(1, 1)),
  keras.layers.BatchNormalization(),
  keras.layers.Conv2D(filters=25, kernel_size=(3, 3), strides=(1, 1),
    kernel_regularizer=l2(alpha),
  activation='relu', padding='same'),
  keras.layers.Conv2D(filters=25, kernel_size=(3, 3), strides=(1, 1),
    kernel_regularizer=l2(alpha),
  activation='relu', padding='same'),
  keras.layers.MaxPool2D(pool_size=(2, 2), strides=(1, 1)),
  keras.layers.BatchNormalization(),
  keras.layers.Flatten(),
  keras.layers.Dense(128, activation="relu"),
  keras.layers.BatchNormalization(),
  keras.layers.Dropout(0.5),
  keras.layers.Dense(64, activation="relu"),
  keras.layers.BatchNormalization(),
  keras.layers.Dropout(0.5),
  keras.layers.Dense(2, activation='softmax')
])

```