UNIVERSITY OF WATERLOO
Software Engineering

Application of Machine Learning Models as a Solution to Hyperbola Identification in Ground Penetrating Radar Data

Sensors & Software Inc.
Mississauga, ON

Prepared by
Xiang Yi (Irene) Chen
Student ID: 20707344
User ID: xy29chen
1B Software Engineering

August 21, 2018

Xiang Yi (Irene) Chen
277 Lester Street
Waterloo, ON N2L 3W6

August 21, 2018

Dr. Derek Rayside, Director
Software Engineering
University of Waterloo
Waterloo, ON N2L 3G1

Dear Dr. Derek Rayside:

The following work term report, entitled "Application of Machine Learning Models as a Solution to Hyperbola Identification in Ground Penetrating Radar Data", has been prepared for Sensors & Software Inc. as my first work term report for my 1B term. The objective of this work term report is to explore the feasibility of different machine learning models to the problem of hyperbola recognition and classification in ground penetrating radar data.

Sensors & Software Inc. is a major provider of ground penetrating radar solutions, offering over both software and hardware of various customized uses—from ice mapping purposes to utility locating and forensics as well as providing for disaster rescue missions.

I would like to thank my supervisor, Adam Fazzari for providing support and guidance along the way, as well as my co-workers for furthering my understanding of geophysics and ground penetrating radars. Finally I would like to thank Sensors & Software Inc. for creating this opportunity and setting me up with the dataset and other resources needed to retrain the models.

I hereby confirm that I have received no further help, other than what is mentioned above, in writing this report. I also confirm that this report has not been previously submitted for academic credit at this or any other academic institution.

Sincerely,

Xiang Yi (Irene) Chen
Student ID: 20707344

Executive Summary

The following report investigates the suitability of a small sample of pre-trained machine learning models, in response to the challenge of identifying hyperbolas in ground penetrating radar (GPR). The problem stems from the fact that hyperbola artefacts in GPR data are highly contextual, and present themselves in many different forms. As such, streamlining the interpretation process with software aids for GPR users proves itself to be a challenge due to the visual variety in GPR data.

However, the identification of hyperbola may be solved with the use of machine learning training. Hence, this report explores the nature of convolutional neural networks (CNN), followed by a comparison of three notable CNN models—the Inception V3 model, as well as MobileNet V1 model, and VGG 19 model—to determine their suitability given restrictions to this identification problem. The choice of the CNN model was then tested with a proof-of-concept application to hyperbola identification.

Based on limitations in labelled datasets, restricted training time as well as non-ideal training environment with limited processing power, the decision to use MobileNet model was made with the trade-off between model distributivity versus accuracy and training load in mind, with a higher emphasis on MobileNet's lightweight architecture. The final proof-of-concept trained model was able to achieve general hyperbola detection, of up to 14 hyperbola detections per image. Its resulting classifier output was merely 22MB in size, trained by 171 labelled images for over 6000 steps.

Recommendations to optimize the model's setup and training based on these restrictions were made, in hopes to further improve the performance of the model. Finally, the report considers certain limitations which have affected the final result of the trained model.

Disclaimer

The pre-trained models and their retraining modules are publicly available on GitHub. They've been applied with proprietary GPR data provided by Sensors & Software Inc.

Table of Contents

List of Figures

List of Tables

1 Introduction

Ground Penetrating Radar (GPR) solutions serve a wide range of uses, from underground utility locating, to pavement or ice thickness mapping, to forensics as well as archaeology. However, because of this varied usage—combined with pre-processing and post-processing—GPR data presents itself in many various ways. Hence, interpretation of GPR data is one of the most error-prone aspects of using GPR solutions.

1.1 Problem Statement

One of the biggest challenges of GPR data interpretation is to avoid false positives, such as extreme ringing caused by metal, as well as interference caused by rebounding air waves off the data collection site.

Sensors & Software Inc.'s software is able to apply velocity fitting and other transformations to GPR data to improve and simplify data interpretation; such transformations can help identify false-positive air wave rebounds in GPR data through a calculation of data velocity. However, the first and most limiting step to streamlining this process is to allow the software to locate a hyperbola—evidence of a disturbance in the soil caused by an object.

1.2 Proposal of Solution

Since machine learning has been making advances in the field of computer vision, it seems to be a fit tool to solve this bottleneck in this interpretation process.

Machine learning, in the broadest of terms, is a subset of computer science which allows programs to improve and "learn" without explicitly being told what exactly to do—the construction of algorithms to make predictions based on recognition of patterns on a data set and to make changes to itself so that its future performance is improved.

Specifically, object detection and localization problems generally involve training feature extraction: associating certain characteristics of the image with different weights of importance based on their reoccurrence. In this problem, a hyperbola feature might consist of its curved nature, fitted against an apex of the underground object to the straight line of the surface. In this case, the presence—or lack thereof—of hyperbolas in GPR data will be explored with machine learning solutions, as well as their differentiation from other forms of features in GPR data, such as metal ringing and boundaries in the soil.

The follow section will be analysing the architecture and processes in a small sample of machine learning models, as well as the transferability and possibility of integration into existing GPR systems. The comparison process will be taking the following criteria into account:

- Correctness and accuracy of learning, based on the core transformations of the architecture of the model
- Ease of being transferred for real-time detection, based on minimizing the model's cost of memory and processing on ARM systems

For the sake of quickly creating a proof-of-concept model, transfer learning will be applied to an existing open-source pre-trained model. Transfer learning entails modifying layers of existing mature models to customize the model while retaining its generic base layers. Specifically, the following pre-trained models will be explored, chosen for their current popularity and abundant availability in terms of support and resources:

- Inception V3
- MobileNet V1
- VGG 19

2 Comparison & Analysis

For the comparison of potential machine learning models, the general idea is to apply supervised learning, inputting labelled datasets to distinguish hyperbolas in the image, then testing the model with sets of unlabelled data of various degrees of resemblance to the inputted training dataset.

2.1 General Procedure

2.1.1 Retraining Neural Networks

In the most intuitive sense, neural networks is an imitation of how the human brain works. Input features are weighted positively or negatively, forming and reinforcing connections between common features similar to neurons forming connections between each other.

Each layer performs a transformation on a single matrix input, and the learning process is achieved by chaining these layers together, either cyclically or linearly. Deep learning refers to models with a great number of layered transformations. The penultimate layer—the bottleneck—summarizes the extracted features, to allow the last layer to perform the classification task.

Finally, weights of a model is a measure of the strength of the reinforcements between connections—equivalent to the output parameters to be learned throughout training and the final product used in order to deploy the model for real-life usage.

2.1.2 Convolutional Neural Networks (CNN)

For this hyperbola identification problem, GPR data can be easily converted to images. Thus, convolutional neural networks(CNN) will be used for their suitability with image inputs and manipulations. A CNN is a subset of multi-layer neural networks, characterized by their convolution layer process—the union of integrals, or how much two functions overlap as one passes over the other—usually used for image classification. This overlap is the main idea behind how common features between images are reinforced and extracted. Processing is achieved by obtaining the union of the product of input images as matrices. [2]

The common components of CNN model architecture are as follow: the convolutional layer, the pooling layer, the ReLU activation layer and the fully-connected layer.

The convolutional layer determines the receptive field of the model—a feature specific to CNN and object detection— which is limited by the size of the convolution as well as its center. The receptive field is essentially the region in the input space that a CNN model can examine a feature, where the importance of the object is exponentially examined the closer it is located towards the center of the receptive field.

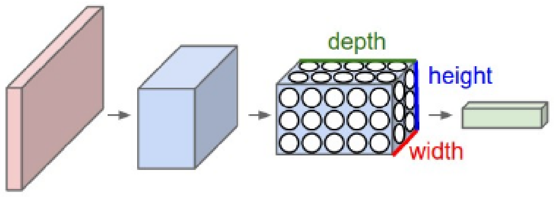


Figure 2-1: A neuron in a convolutional network condensing image inputs into a 3D entity. [**?**]

Pooling is a transformation specific to convolutional neural networks: it combines clusters into a single entity to be used in the next layer, to condense and downsample previous operations with the intent to reduce parameters. This step assumes an image input and condenses the input into a 3-dimensional array, a convolutional neuron as seen in Figure 2-1. Max pooling, one of the most common types of pooling, is a downsampling function which takes the largest value from the prior cluster, effectively reducing the volume of data to process. Downsampling is not only used to reduce processing load, but by discarding select information, it prevents the model to be overfitted too closely to the training dataset, and hence, less accurately solving the real-life problem, as demonstrated in Figure 2-2.

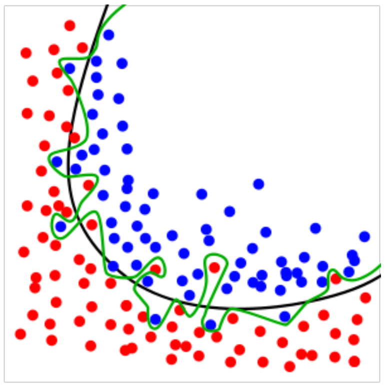


Figure 2-2: Example of an overfitted model classifiying two classes in green, a case to be avoided. [**?**]

The convolution process is usually paired with a concatenation step as well as an activation ReLU layer. ReLU stands for rectified linear unit, a process to introduce non-linearity in the CNN—effectively performing $f(x) = max(0, x)$. This is a transformation on convolution outputs that doesn't affect the size of the matrix.

Finally, the fully connected (FC) layer flattens the transformed image matrix. During this FC layer, the softmax function normalizes the output to land between 0 and 1, as a sigmoid function similar to a categorical probability distribution. In this case, it should be executed in the end to bring the model closer to indicating the probability and likelihood of hyperbolas.

2.2 Inception V3

Inception V3, also previously known as GoogleNet, is a CNN of 11 convolutional layers, along with the typical max pooling and softmax functions towards the end of the model. However, it differs from the standard design of CNN models by its successive stacking of different sized convolution functions scattered throughout the model between the convolutional layers, and is hence recognized as a deep training network.

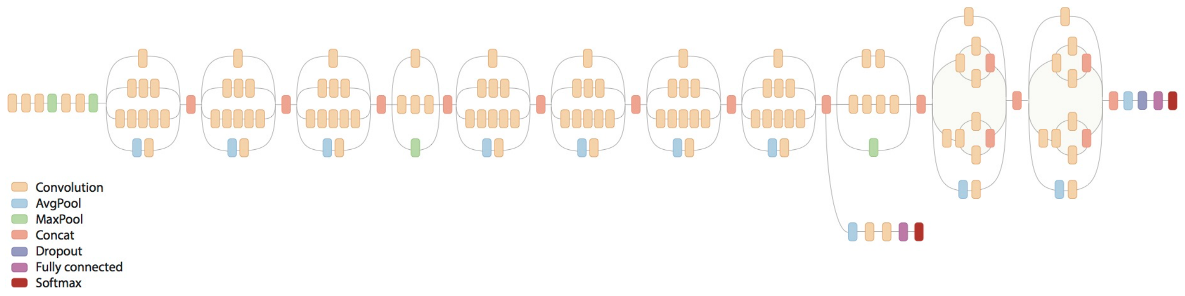


Figure 2-3: Inception V3's CNN model architecture as described. [?]

Simply, Inception extracts features at multiple levels, computing 1×1, 3×3, as well as 5×5 convolutional layers, which are concatenated afterwards as seen in Figure 2-3. This particularity of the model allows the training to not only cover greater depths, but allows better receptive field variability of inputs, which in turn leads to being able to detect hyperbolas of different scales when applied. As well, the usage of smaller convolution layers stacked to replace a larger input through factorization, as demonstrated in Figure 2-7, also reduces the process load to train the model. This design is due to the fact that convolutions with larger spacial filters are disproportionally computationally-heavy.

Although Inception's multi-layer convolution process is extremely desirable, by increasing the density of data of this model, it drastically increases the computational costs. The

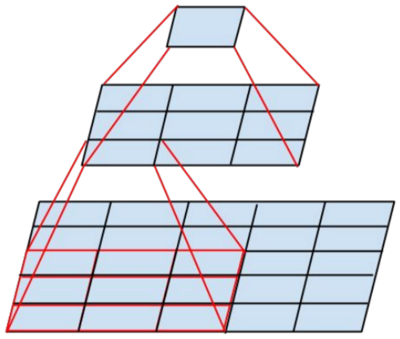


Figure 2-4: Factorizing and substituting convolution layers of smaller dimensions to cover the input matric. [**?**]

presence of large 5×5 convolutional operations are exponentially more expensive to compute, compared to their 3×3 and 1×1 counterparts.

Unlike many CNN models, Inception counters this computational bottleneck by skipping intermediary activation layers, making its computational cost much lower than the other two models compared. This makes Inception a likely candidate to train our hyperbola classifier problem in an environment where memory and computational capacity of the hardware is limited. Its accuracy doesn't seem to be highly impacted either. Inception V3 places quite highly for ImageNet's standards of Top-1 accuracy. [1]

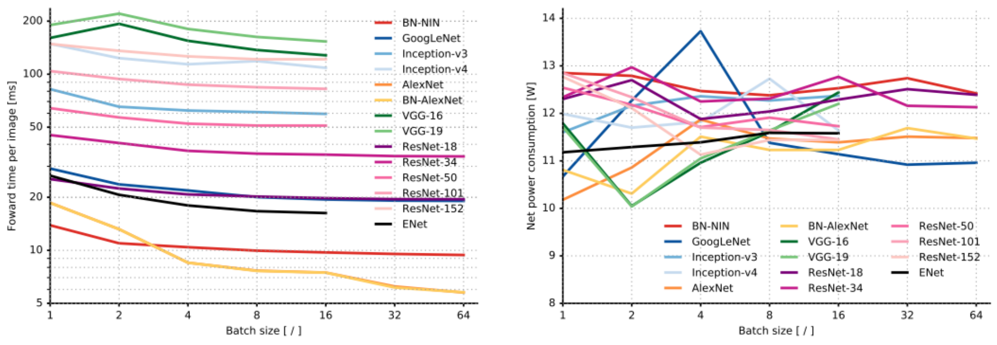


Figure 2-5: In the paper "AN ANALYSIS OF DEEP NEURAL NETWORK MODELS FOR PRACTICAL APPLICATIONS" show satisfactory results of Inception V3 compared to other models trained under the same conditions. [1]

As for the result of the model, the weights for Inception V3 are relatively small, coming in at 92MB, generated from a total depth of 159 stacked layers per training iteration.

2.3 MobileNet V1

MobileNet, as its name suggests, is a CNN model specifically tailored to run on resource constrained environment such as mobile platforms. This architecture is made of a total of 14 batches of convolution throughout the model, followed by one average pooling function as well as one fully connected layer, ending with the typical softmax function.

To lighten the processing, it applies a depth-wise separable process of convolutions similar to Inception (Figure 2-6). Essentially, a 3×13 factorized convolutional layer is applied to the input matrix linearly in the depth direction, then a 1×1 convolution concatenates the matrix back layer-wise called a point-wise convolution layer. Unlike Inception, it doesn't contain any larger convolution processes expensive to process. This, on one hand, will lead to fast training and runtime, but on the other hand, limits the receptive field to slightly smaller hyperbola detected in the images. In the case of hyperbola detection, these features will not be located in the entirety of the input GPR data, thus it isn't a impactful trade-off.

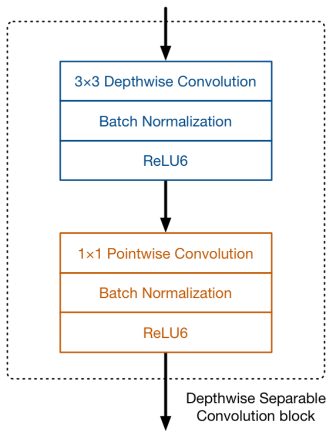

3×3 Depthwise Convolution
Batch Normalization
ReLU6
1×1 Pointwise Convolution
Batch Normalization
ReLU6
Depthwise Separable Convolution block

Figure 2-6: Visualizing the depth-wise separable convolution process. [?]

However, there is a trade-off in accuracy for this lightweight model. Using standards of ImageNet, in a comparison paper of these three models, MobileNet ranks on the lower end of the three models, achieving a Top-1 accuracy of 71.0—noticeably lower than both Inception V3 and VGG 19, as seen in Table 2-1.

However, since the size of weights of the model—an expected 22MB—are significantly smaller, there seems to be a worthwhile trade-off when comparing the ratio of the model size to the accuracy reached.

Model	Top-1
Inception V3	78.0
VGG 19	77.3
MobileNet V1	71.0

Table 2-1: ImageNet standards top-1 accuracy comparison between models.

2.4 VGG 19

VGG 19 consists of five convolutional layers as well, 3×3 stacked on top of each other in increasing depth, designed by the Visual Geometry Group of Oxford. ”19” stand for the number of weight layers in the network, the sum of five batches of convolutional layers as well as max pooling to reduce volume size and finally ending with two fully connected layers as well as a softmax process.

VGG’s 3×3 convolutional processes results in fixed reception fields, which is disadvantageous in terms of hyperbola detection accuracy. This may also lead to overfitting the model to be able to only detect certain types of hyperbola, highly similar to those provided in the labelled dataset.

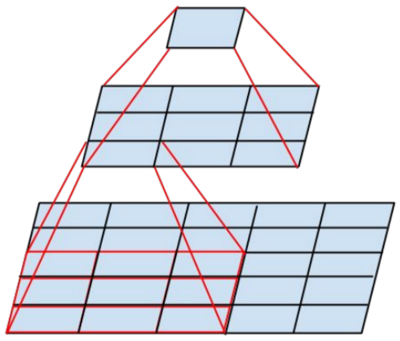


Figure 2-7: Factorizing and substituting convolution layers of smaller dimensions to cover the input matric. [?]

Based on the older AlexNet model—which is characterized by implementing a dropout function after each FC layer—the VGG model attempts to compensate for overfitting with a dropout process after each fully connected layer. However, the downside of VGG’s accuracy is in its difficult deployment and training process due to the great calculation requirements of such a computationally-heavy model—a disadvantage both in processing speed and memory requirements.

The network architecture weights themselves are quite large. Due to its depth and number of fully-connected nodes, the VGG model itself is over 574MB for VGG19. Hence, the biggest drawbacks of VGG, despite its good depth and accuracy, would be hard to deploy onto Sensors & Software Inc.'s GPR systems, and would only be something usable in EKKO Project, a post-processing software product. Due to this great drawback, the trade-off for accuracy of the VGG19 model is deemed too large and expensive to be used in this hyperbola classification situation.

2.5 Decision

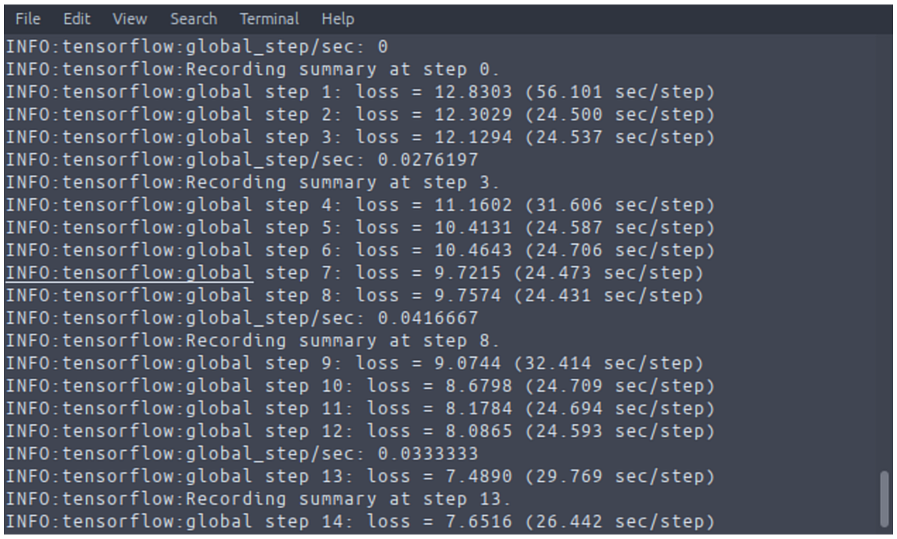
After an overall comparison of the three models, the most advantageous one seems to be MobileNet. It has a good trade-off between its small deployable size versus its accuracy in performance. However, MobileNet's small size is what puts this model above the rest; on a GPR device, memory and processing power is limited to less than 4GB at X GHz, not including the resource requirements for the pre-existing GPR data collection software. As it's similar to our day to day current mobile devices, MobileNet's potential deployment to an ARM system is more likely to succeed.

As for the other two, considerations have been made, notably for Inception V3's highly efficient training to model size ratio, making it a very good candidate for usage in higher performing environments. However, the final decision was based on future potential application and usage, as well as ease of training, and hence—although very close—the Inception model was ruled out.

Finally, although VGG is quite accurate out of the three models examined, deployment and training in such a short time frame is simply not feasible enough, and hence, VGG 19 was ruled out.

3 Concluding Results

Through the proof-of-concept classifier, experimental results show that MobileNet is about 3 times as fast as Inception and 10 times as fast as VGG, running at an average of 27 seconds per batch of 24 images, as seen in Figure 3-1. Despite slightly lower accuracy, it still performed reasonably accurate, with over 6000 steps of training using a pre-processed labelled dataset of 171 images. Its loss—initially at 12.83—was reduced to an average of 1.8 per step, reaching as low as 1.12.



```
INFO:tensorflow:global_step/sec: 0
INFO:tensorflow:Recording summary at step 0.
INFO:tensorflow:global step 1: loss = 12.8303 (56.101 sec/step)
INFO:tensorflow:global step 2: loss = 12.3029 (24.500 sec/step)
INFO:tensorflow:global step 3: loss = 12.1294 (24.537 sec/step)
INFO:tensorflow:global_step/sec: 0.0276197
INFO:tensorflow:Recording summary at step 3.
INFO:tensorflow:global step 4: loss = 11.1602 (31.606 sec/step)
INFO:tensorflow:global step 5: loss = 10.4131 (24.587 sec/step)
INFO:tensorflow:global step 6: loss = 10.4643 (24.706 sec/step)
INFO:tensorflow:global step 7: loss = 9.7215 (24.473 sec/step)
INFO:tensorflow:global step 8: loss = 9.7574 (24.431 sec/step)
INFO:tensorflow:global_step/sec: 0.0416667
INFO:tensorflow:Recording summary at step 8.
INFO:tensorflow:global step 9: loss = 9.0744 (32.414 sec/step)
INFO:tensorflow:global step 10: loss = 8.6798 (24.709 sec/step)
INFO:tensorflow:global step 11: loss = 8.1784 (24.694 sec/step)
INFO:tensorflow:global step 12: loss = 8.0865 (24.593 sec/step)
INFO:tensorflow:global_step/sec: 0.0333333
INFO:tensorflow:global step 13: loss = 7.4890 (29.769 sec/step)
INFO:tensorflow:Recording summary at step 13.
INFO:tensorflow:global step 14: loss = 7.6516 (26.442 sec/step)
```

Figure 3-1: Initial training, loss at 12.83 and rate of 27 seconds per batch.

From around 1300 steps to 4000 steps, MobileNet V1 performs quite well in improving itself, as seen in Figure 3-2 and 3-3.

With further iterations and adjusting the learning rate along the way, the boxes detecting hyperbola are tightened. As well, less obvious hyperbola are detected with overall higher confidence percentage, going from 79% in Figure 3-4 to both 88% in Figure 3-5 after 6000 steps.

In the end, the model's output graph totalled to about 22.2MB, which opens up the possibility of a very lightweight memory-efficient deployment onto Sensors & Software Inc.'s GPR products.

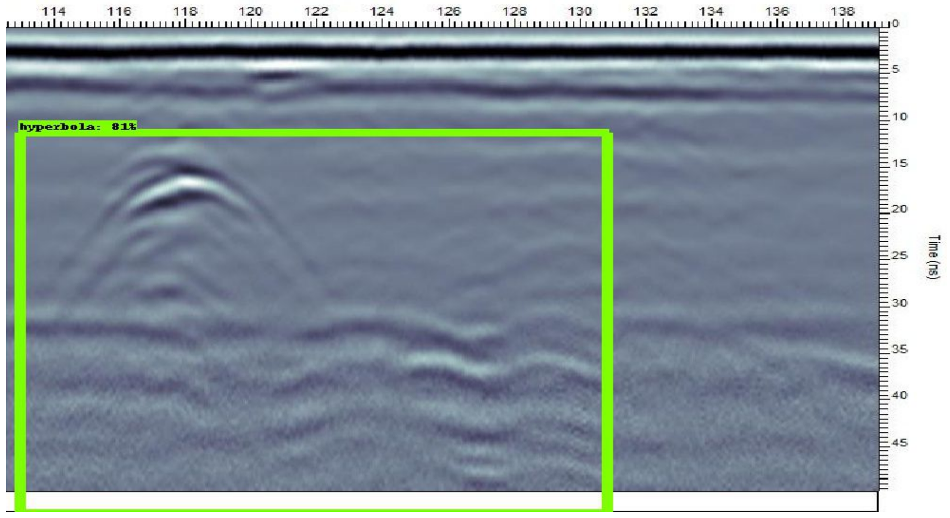


Figure 3-2: Initial detection tested at 1300 steps, which took approximately 10 hours to train.

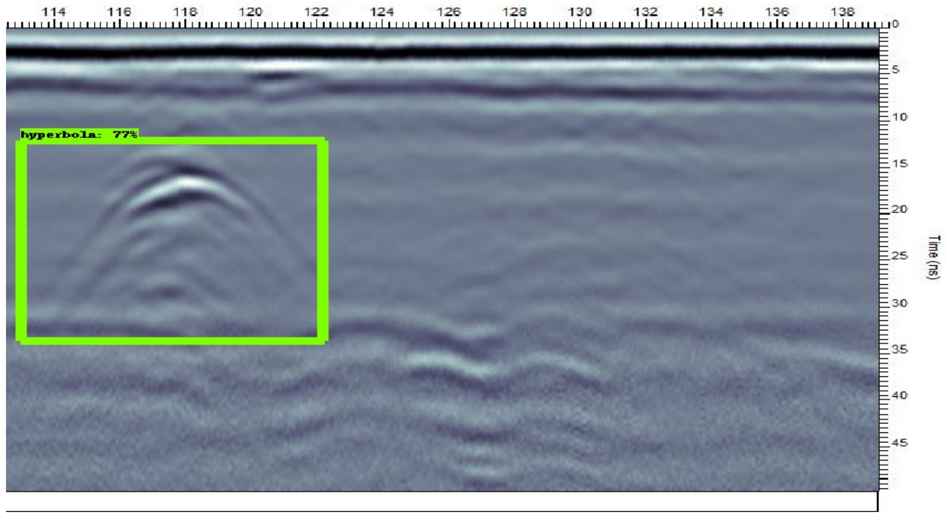


Figure 3-3: Results of the same image, after over 4000 steps of training over the course of 2 days.

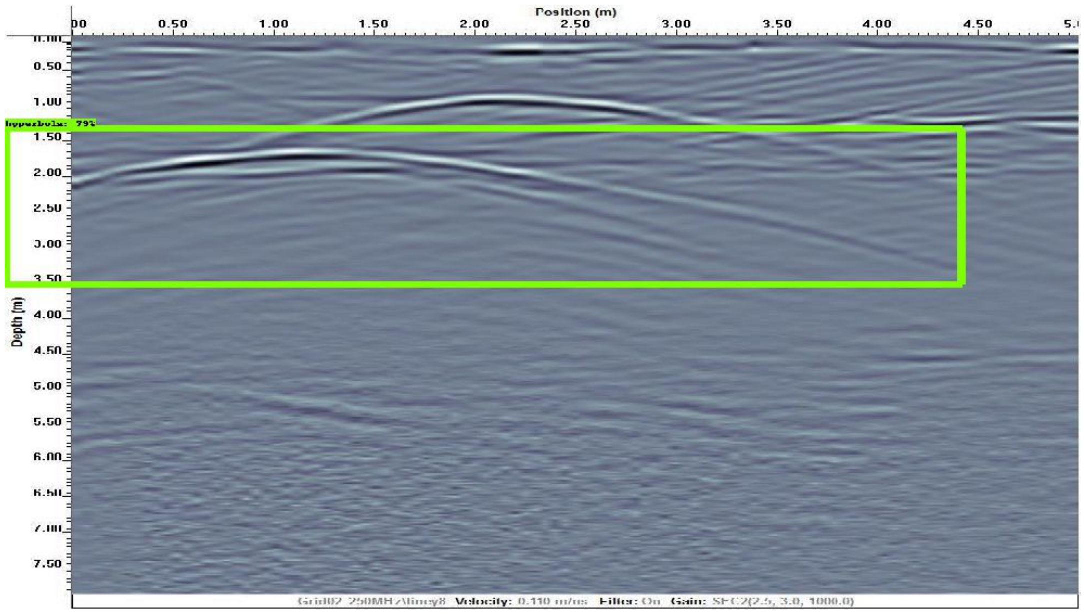


Figure 3-4: Phase one testing

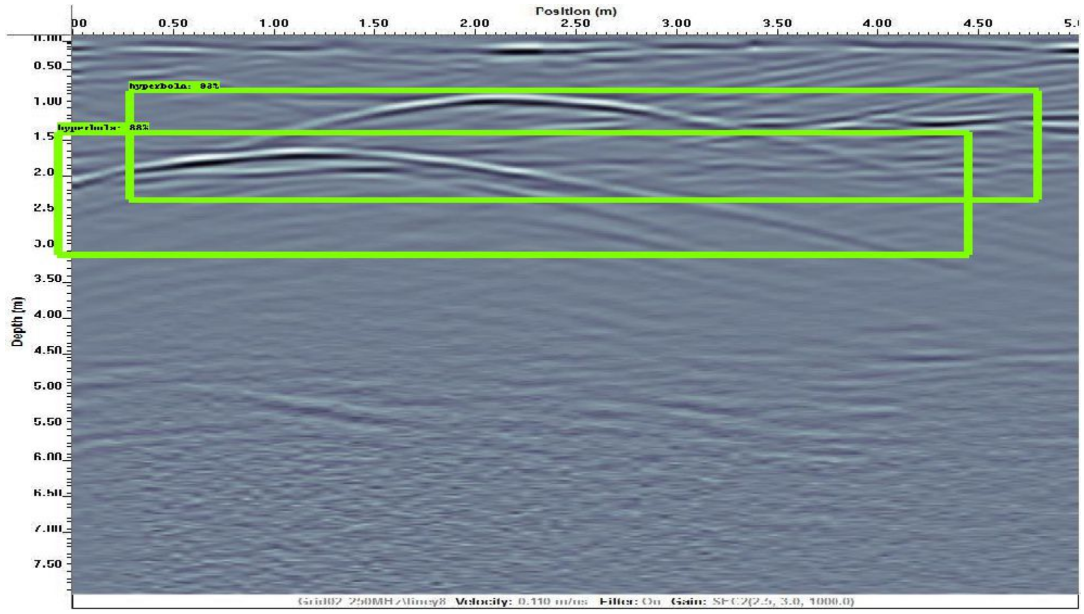


Figure 3-5: Phase two testing on the same dataset

3.1 Conclusion

All in all, the choice of MobileNet V1 as the CNN training architecture was a successful decision. Over the course of a couple of days and with merely 171 labelled images, the hyperbola classifier's performance has exceeded my expectations. MobileNet is indeed both lightweight and accurate enough for rudimentary hyperbola detection, despite the large variability in how hyperbolas present themselves in GPR data.

4 Recommendations

My proof-of-concept consisted of three training iterations, totalling to over 6000 steps. Although successful, this is but a limited comparison of a small selection of models—though three very well-known models.

As well, hyperbola detection and identification has proven itself to be a problem difficult to tailor to its variability. Throughout the training, the challenge of not overfitting the model to take a balanced account of different hyperbolas has presented itself multiple times. At this stage, since the current proof-of-concept was trained using a manually created dataset, it is only a rough representation of hyperbola identification. Needless to say, to attempt to capture the full complexity of the problem, a better breadth of images—a wide variety of hyperbolas in different environments, different soil calibrations, different collection speeds and GPR stacking—is required to avoid the model to over-fit to noisy or coincidental features in the GPR data.

In terms of expanding the model, without thorough understanding of the effects of each transformation, certain computational gains during training can be lost. If scaled or applied to the problem naively, MobileNet's advantage in its performance to size relation over other similarly performing models may immediately be voided.

References

[1] A. P. Alfredo Canziani, Eugenio Culurciello, “An analysis of deep neural network models for practical applications,” 2016.

[2]