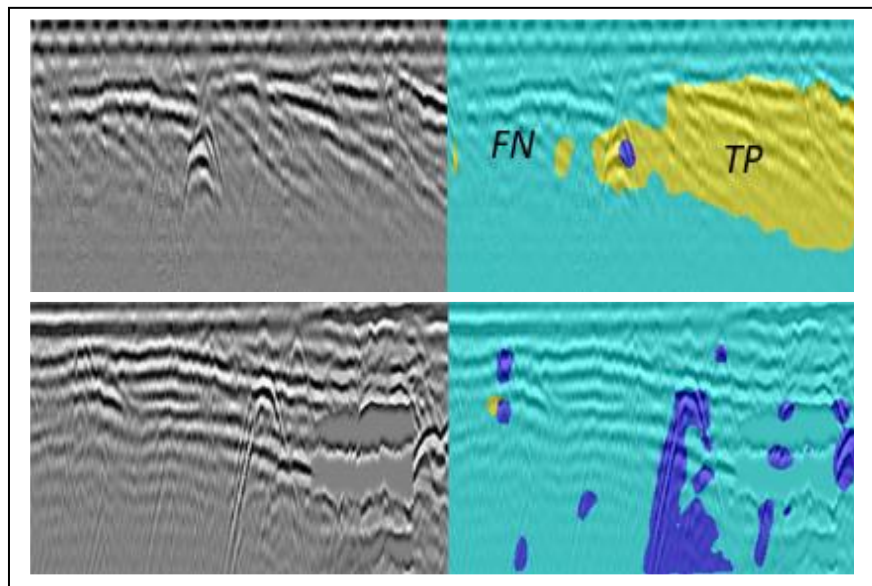




DEPARTMENT OF EARTH SCIENCES

DEEP LEARNING NEURAL NETWORK IN GEOLOGICAL INTERPRETATION OF GROUND PENETRATING RADAR IMAGES



Sven Svensson

Degree project for Bachelor of Science with a major in Earth Sciences
2024, 180 HEC

Abstract

The post-processing of GPR subsurface survey data is both time-consuming and involves the highly subjective task of geological interpretation. In this study, a Deeplab v3+ convolutional neural network is used to automatize the process in an attempt to improve efficiency and to reduce the impact of subjectiveness. For this, a semantic segmentation approach is used for sedimentology classification of radar facies in GPR images. Manual interpretation of 143 images with size 150x360 pixels, from a beach area with mostly sand and clay sediments, resulted in the identification of seven different radar facies. In three experiments, the neural network was evaluated with subsets of images with these facies. From a subset of 20 images with two radar facies, the neural network was able to correctly identify facies with a precision of >90%. From another subset of 44 images with two radar facies and a separate classification of remaining pixels, the facies were correctly identified with a precision in the range of 43%-49%. In a subset of 43 images with six radar facies, the network could identify five of the facies with a precision in the range of 21% - 54%, and one of the facies could not be identified. The lower precision for the dataset with six facies was probably due to a lower representation in the dataset. The study shows that a convolutional neural network is a feasible method for automatizing GPR image interpretation for sedimentology applications. It also demonstrates the importance of using a substantial and representative dataset for learning the neural network.

Contents

Abstract	1
1 Introduction.....	3
1.1 Related work	3
1.2 Scope of study and research questions.....	5
2 Background.....	6
2.1 GPR.....	6
2.2 AI deep learning neural networks.....	8
2.2.1 The composition of a neural network.....	8
2.2.2 The Convolutional Neural Network	9
2.2.3 Training a network	11
2.2.4 Training and evaluation metrics.....	12
2.3 Interpretation of GPR images.....	13
3 Materials and method	15
3.1 Materials	15
3.2 Method	17
4 Results	22
4.1 Training performance.....	22
4.2 Prediction performance	24
4.3 General metrics.....	26
4.4 Prediction views of test images	27
5 Discussion	28
6 Conclusions.....	30
7 Acknowledgements	31
8 References	32
Appendix 1.....	35
Main script semanticSegmTest.....	35
Function partitionTrainingData	37
Appendix 2.....	39

1 Introduction

Ground Penetrating Radar, GPR, is a well-proven cost-effective geophysical method for subsurface non-invasive surveys (Jol, 2008; Milsom & Eriksen, 2011). The GPR equipment is mobile, fast, and relatively small, (e.g. Guideline Geo AB, n.d.), which simplifies the implementation of surveys and the capturing of radar image data, also in areas where accessibility or stability issues limit other ground investigation methods. However, the post-processing of radar image data is time-consuming and involves geological interpretation, a task that is highly subjective and based on the expertise of the analysts (Bristow & Jol, 2003; Jol, 2008). To improve the process, it is interesting to evaluate Artificial Intelligence, AI, as a method. AI has shown promising results in image data processing and has the potential to both decrease subjectivity and improve overall efficiency of GPR surveys. This, in turn, can lead to better decision support in subsurface related investigations.

1.1 Related work

Several studies related to GPR utilize AI deep learning framework for subsurface interpretation and classification, see e.g. the overview in Küçükdemirci & Sarris (2022). Much of this work is directed to other disciplines than geology, such as civil engineering and infrastructure, military applications, archaeology, etc, and the focus is on localizing buried objects e.g. tunnels, pipes, cables, landmines, and archaeological features. Better in line with the geological field are studies that explore quantitative measures for interpreting distinct units, so called 'radar facies', of GPR images, but such studies are sparse. An example is Moysey et al. (2003) where synthetic models of four radar facies are used to train a deep learning network for the purpose of analysing aquifers. The accuracy for detection of radar facies in the synthetic images was approximately 90%. The network was also used to evaluate real data from an aquifer in Canada. For this data the accuracy of detection was in the range of 44-80%. No information is given on the size of training and test data in this paper. In Moysey et al. (2006) different measures of radar texture are analysed and compared for quantitative radar facies interpretation with deep learning. This study classified three radar facies with a detection accuracy in the range of 42-98%. No information on the size of the training data is given.

Numerous studies (Neal, 2004) confirm a relationship between primary radar reflections and primary beddings, i.e. the form and orientation of bedding and sedimentary structure. Here 'radar stratigraphy' has been used as a descriptive term that correlate GPR with specific sedimentary environments (Neal, 2004; van Overmeeren, 1998). Further, studies can be found that utilize a radar stratigraphy approach to interpret and categorize various sedimentary environments. In these studies, specific radar surfaces and facies based on the primary radar facies elements are identified and used in a categorization and geological interpretation of the surveyed environments (Figure 1). Examples of such studies are van Overmeeren (1998) that attempts to compile a radar facies 'atlas' based on surveys of different depositional sedimentary environments in the Netherlands. Aradóttir et al. (2022) that categorize and make a geological interpretation of different radar facies of a drumlin formation in Iceland. Similar work from various marine environments are e.g., Costas et al. (2006) with a survey from Spain, Magalhães, et al. (2017) with data from Brazil, Ribolini, et al. (2022) from Italy, and Shan et al. (2015) from a survey in China. An example of a radar facies approach for an aeolian environment can be found in Fu, et al. (2019).

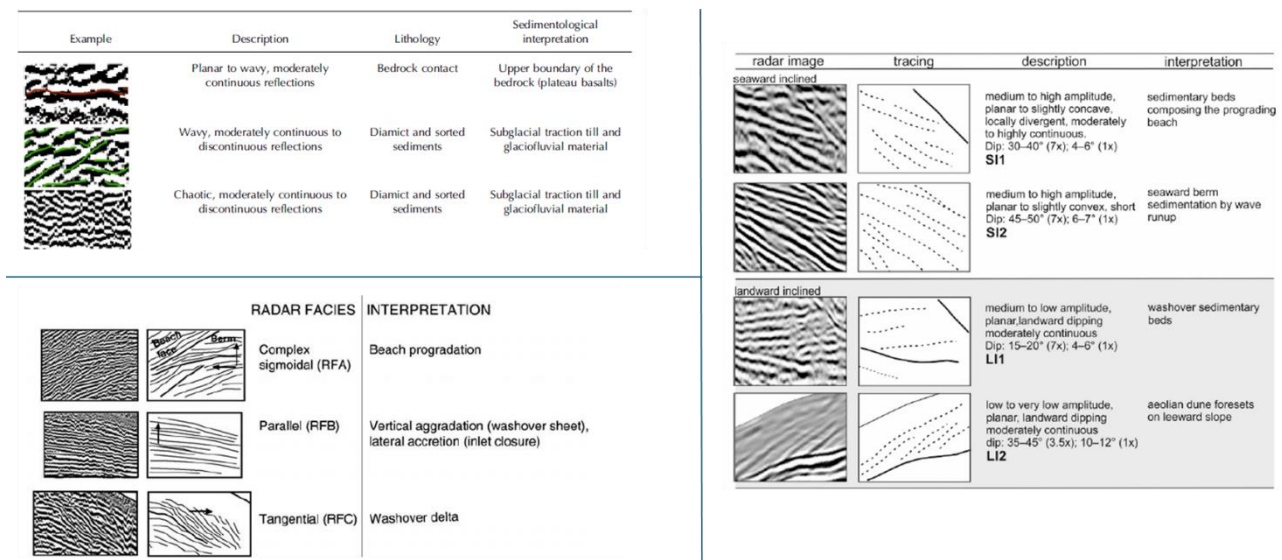


Figure 1. Examples of radar stratigraphy classification in radar facies. Top left from Aradóttir et al. (2022), bottom left from Costas et al. (2006), and right from Ribolini et al. (2021).

Considering the similarities between GPR and seismic reflection data images (van Overmeeren, 1998; Neal, 2004) additional views, ideas, and concepts can also be found in this area. Seismic reflection is a more mature technique than GPR and vast data exists from many surveys in the oil and gas industry. Several studies utilize deep learning methods for classification and interpretation of seismic images. In Kaur et al. (2023) two deep learning architectures, a convolutional neural network and a generative adversarial network, are compared regarding effectiveness in seismic facies classification into six classes based on huge datasets (25,000-60,000 images) from a basin area offshore New Zealand. The study showed a precision of 89,5-99,8%. A study by Liu et al. (2019) develops a classification framework that is used together with one convolutional neural network to classify four seismic facies with data from a basin area offshore China. The dataset sizes for training and test are in the range of 1500-2000 pixels. The resulting precision is in the range of 64-88%. Wrona et al. (2018) train and compare 20 deep learning models with four facies classes in the interpretation of seismic reflection data of the northern North Sea. Also in this study big datasets are used (10,000 images), and the precision achieved by the models was >95%. In Zhao (2018) the performance of a patch-based neural network is compared with a convolutional neural network in the classification of seismic reflection data from the North sea, offshore Netherlands. The datasets used for training are enormous (400,000 patches for the patch-based network and 40 full inline images for the convolutional network). This study uses nine seismic facies for classification but gives no information on resulting precision.

Normally, to reach an error level that is reasonable for a deep learning model, a vast amount of relevant training data is needed. The availability of published GPR datasets is therefore an important aspect for studies. It is evident that there is a lack of such archives (e.g. Zhang et al., 2021, and Küçükdemirci & Sarris, 2022). However, some examples of open data collected in coastal region surveys can be found, e.g. Howard & McPherson (n.d.) from Australia, and Thieler et al. (2013) and Forde et al. (2016), from the United States. Another open dataset is provided by Woodard et al. (2020) from a survey of a drumlin in Iceland.

1.2 Scope of study and research questions

This work examines the possibility to use AI in the form of a deep learning neural network as a method for sedimentology interpretation of GPR radar image data. The research questions addressed in the study are:

- What is the performance of a neural network regarding detection of sedimentology features, and what are the method's strengths and limitations?
- What characterizes useful training data, what is a proper amount of training data, and how do we retrieve it?

2 Background

2.1 GPR

Ground penetrating radar, GPR, is a technique that detects electrical discontinuities in subsurface by transmitting high-frequency electro-magnetic waves into the ground and receiving the response of reflected waves (Figure 2). The reflected waves are combined and further processed using various methods (Jol, 2008) to compose high-resolution images of the ground (Figure 3). GPR equipment comprise antenna, transmitter, receiver, and control electronics, normally contained in a mobile unit on wheels or meads that is moved over the surface to survey the subsurface (Figure 4).

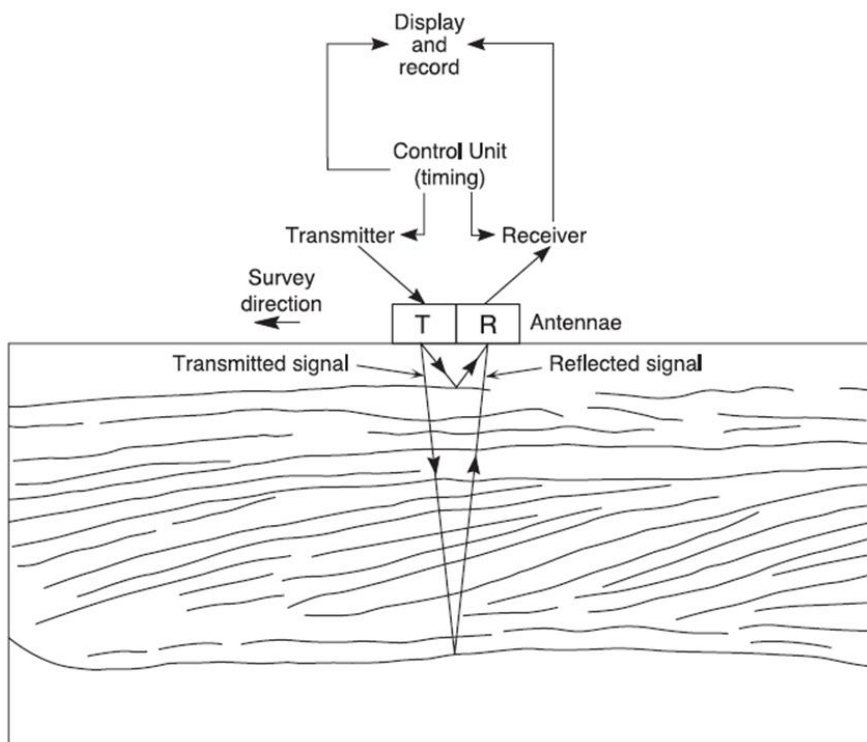


Figure 2. Principal view of GPR describing the transmitter and receiver function, the reflection of the radar signal and the control unit and display functionalities. From Neal (2004)

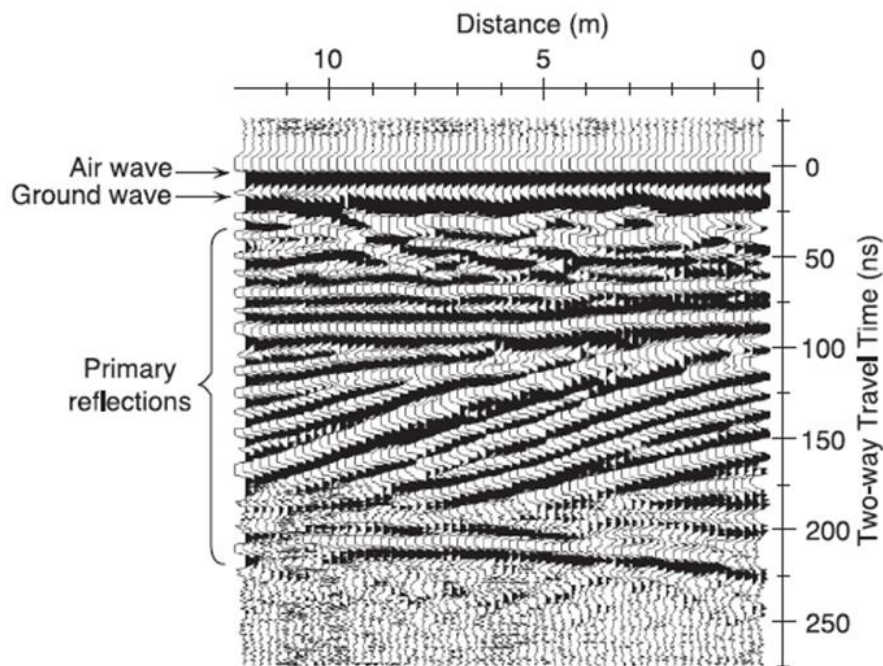


Figure 3. An example of a radar image produced by a GPR. The indicated air wave and ground wave are unwanted signals that are irrelevant for the subsurface analysis. The air wave represents a signal travelling directly between the transmitter and the receiver in air by the speed of light. The ground wave is travelling on the surface of the ground. Each vertical line represents the reflected signal from a radar pulse sent out by the transmitter and received by the receiver. From Neal (2004).



Figure 4. Picture describing a GPR in operation. From Guideline Geo AB (n.d.).

The wave propagation and the reflections in subsurface can be expressed by Maxwell's equations and constitutive relations based on the electrical and magnetic properties of the earth material (Jol, 2008; Milsom & Eriksen, 2011). For most GPR applications the dielectric permittivity, ϵ , and the electrical conductivity, σ , are important, whereas the magnetic permeability, μ , is normally neglected. The dielectric permittivity is a factor that controls the velocity of the electro-magnetic wave, whereas the

electrical conductivity results in energy losses that attenuates the wave. As such the conductivity has an impact on the depth of wave penetration, which typically is < 50 m.

Another factor that impacts the depth of wave penetration is the radar frequency, with a reduction in penetration with a higher frequency. On the other hand, an increase in frequency also increases the resolution in the reflected wave.

Both permittivity and conductivity vary in a wide range in normal ground depending on several factors, such as the type and composition of the geologic materials, the amount and type of fluid that occupies pore spaces, and for sediments the size, shape, orientation and packing of grains, etc. These variations are exploited by GPR when the electro-magnetic wave encounters a boundary with significant discontinuity in permittivity (and conductivity). The discontinuity leads to a sudden change in the propagation velocity of the wave, and thereby energy is reflected with an amount that is in proportion to the magnitude of the change (Jol, 2008; Milsom & Eriksen, 2011; Neal, 2004). The returning wave is detected by the GPR equipment with a signal amplitude that relates to the reflected energy. Normally the amplitude is coded in grey scale or in colour when presented in radar images (Figure 3).

2.2 AI deep learning neural networks

Artificial intelligence is “technology that enables computers and machines to simulate human intelligence and problem-solving capabilities” (IBM, n.d.a, section *What is AI?*). The concept of AI encompasses many disciplines, of which machine learning, and its sub-discipline deep learning, is of interest in this study. Deep learning utilizes neural networks in their task of ‘learning’ from large sets of data, in a way like the processes of the human brain. The neural networks are shown to be efficient models for statistical pattern recognition (Bishop, 2006) and as such suitable for image processing (e.g. IBM, n.d.b, and Camunas-Mesa et al., 2019).

2.2.1 The composition of a neural network

The neural network consists of layers of nodes or neurons that are interconnected with synaptic connections through which data and computation flow (Figure 5). The input layer receives the data to be evaluated and the output layer presents the result, e.g. in the form of statistics, or an interpreted image. The function of the neuron (Figure 6) is; 1) to receive inputs from several neurons in the preceding layer, 2) to multiply each input with an individual synaptic weight, 3) to sum the weighted inputs together and finally, 4) to pass the sum to an activation function that calculates the output to the next layer of neurons (Camunas-Mesa et al., 2019). The networks are normally run iteratively with automatically adjusted parameters (e.g., the weights) as part of the learning process. The iterations run until either an optimal model is created, or other parameter thresholds are fulfilled.

Deep neural network

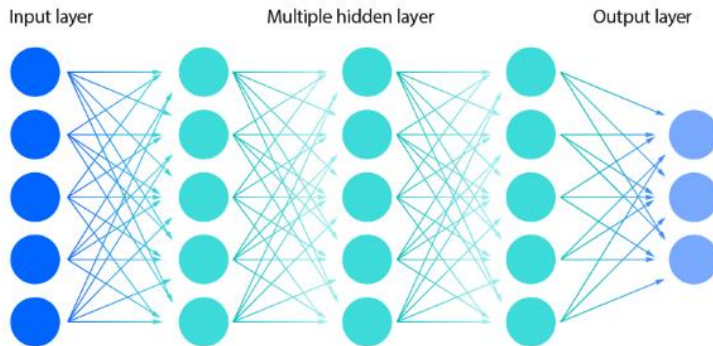


Figure 5. A schematic view that describes a deep neural network with its layers of neurons; the input layer, the hidden layers (not reachable by the user) and the output layer. From IBM (n.d.a).

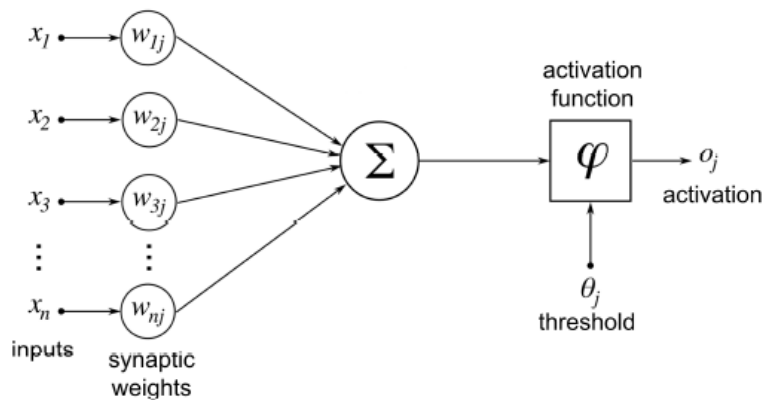


Figure 6. A flow diagram that describes the artificial neuron internal structure and function. Each input is multiplied by a weight, w_{x_i} , then summed together and finally mapped to the activation output with an activation function, φ , that is controlled by a threshold, θ_j . From Camunas-Mesa (2019).

2.2.2 The Convolutional Neural Network

A type of neural network that has been widely applied to image data and used in many GPR studies is the convolutional neural network, CNN (e.g. IBM, n.d.b; Zhang et al., 2021; Travassos et al., 2021). The architecture of the CNN (Albawi et al., 2017; Peemen, 2017; Yamashita et al., 2018) consists of convolution layers (including activation function), pooling layers, and fully connected layers for classification. Normally several pairs of convolution and pooling layers forms the first part of the CNN, after which one or more fully connected layers follows (Figure 7).

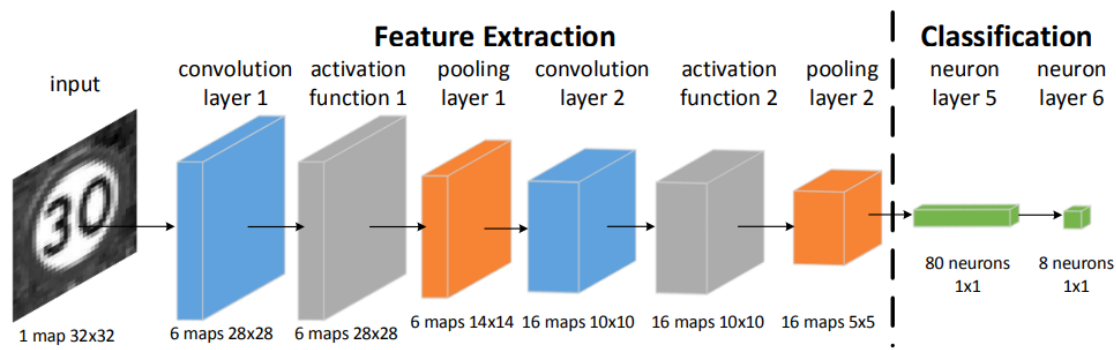


Figure 7. Picture describing the design and data flow of a convolutional neural network with stacked groups of convolution, activation, and pooling layers, followed by the classification layers. From Peemen (2017).

The convolution layers introduce so called 'feature maps' in the neural network. Feature maps are based on convolutional filters or kernels that detect specific features of the image, such as edges, colours, textures, etc. They are in their construction relatively invariant to translations and distortions of the input image, i.e. not that sensitive to spatial properties of the image (Bishop, 2006). Typically, since multiple composite aspects of features as patterns, parts, etc. needs to be detected there are also multiple convolution layers in the CNN. As the data propagates into deeper layers greater portions and larger elements of the image are recognized until the intended object finally is identified (Figure 8).

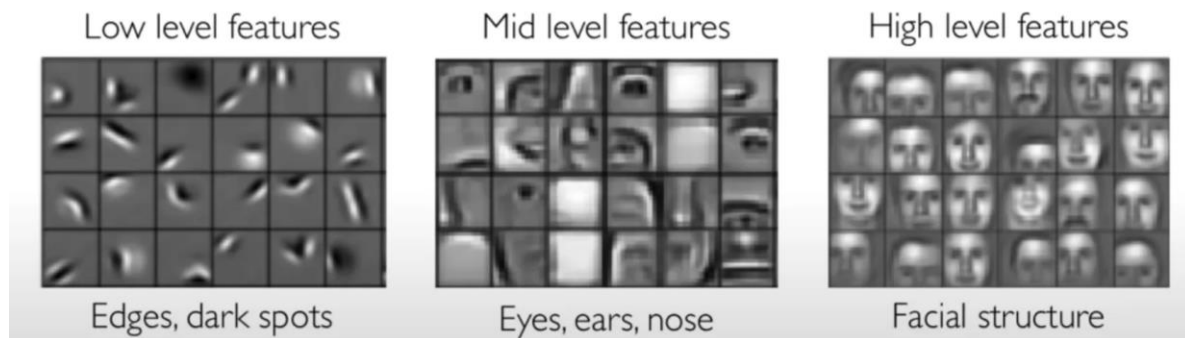


Figure 8. Figure explaining the successive detection of features in a convolutional neural network. From Knime (2022).

Before data propagate to the next layer, the result of the convolutional layer is transformed by a non-linear activation function that adjusts or cuts-off the output. The layer following a convolutional layer is normally a pooling layer, which acts to reduce the complexity of the feature map for following layers by condensing the spatial information, like a reduction of the resolution.

Following the layers of convolution and pooling, the processed data finally is input to one or more fully connected layers. These are generic neural network layers that have an architecture where each node is connected to every node in the previous and the following layer. The function of the fully connected layers is to generate a probabilistic categorization of the data that finally forms the output of the CNN. In this context the CNN is implemented either for object detection, or for semantic segmentation. Object detection means to localize and classify specific objects in the image. Semantic segmentation involves grouping pixels into regions based on their semantic meaning and classifying these regions (Figure 9).



Figure 9. Figure explaining object vs semantic segmentation. Modified from IBM (n.d.).

For semantic segmentation problems Fully Convolutional Networks, FCNs, are often used. They constitute a type of CNNs that show improved performance compared to traditional models. One example is the DeepLabv3+ net from Google (Chen et al., 2018).

2.2.3 Training a network

The learning or training of the neural network is essentially the process of iteratively adjusting the synaptic weights based on the total network output as a response to the input. Compared to the traditional deep learning neural network, the training of a CNN also involves finding the filters or kernels for the convolutional layers (Yamashita et al., 2018). In general, training is automatized and performed stepwise by feeding the network with annotated data, measuring the error contained in the prediction and updating the network weights using a scheme known as error backpropagation. In this process a cost function is used, and the iterative learning process has the task to minimize the value of the cost function (Bishop, 2006; Yamashita et al., 2018).

The initial input data is divided into three parts: training, validation, and test (if independent test data have not already been set aside). The validation and test data are used in two additional phases in the preparation of the network apart from the training. Validation involves the task of evaluating the network during training, to fine-tune parameters, and to handle model selection. Test is the last process phase where performance of the final, trained, and fine-tuned neural network is evaluated (Yamashita et al., 2018). All these phases require data and normally the total available data is split between them by a user-defined percentage, often 70-80% for training and validation and the rest for test. It is important that the datasets are truly separate since otherwise the neural network would risk becoming 'overfitted', a term used to describe that the model learns properties very specific to the training set. The problem is related to the algorithm used for calculating the adjustments of weights in the network and increases when a reduced dataset is used for training a complex model. Overfitting can be resembled as the model is memorizing irrelevant noise on the cost of generalized properties. Overfitting is a main challenge in machine learning and results in reduced performance for the operational neural network when it later is fed with data it has never been exposed to (Yamashita et al., 2018).

Training a neural network is a computing intensive task that is very time consuming for a traditional CPU based computer. However, with the parallel nature of neural networks they can naturally be mapped to the architecture used in graphics processors, GPU, giving magnitudes of improved performance (NVIDIA, 2024). Recently this technology has seen a rapid development that has led to reduced system cost, hence neural network studies are today feasible also with limited budget.

Transfer learning (IBM, n.d.c) is a concept that can limit the time for training. It is based on the principle of reusing a neural network that has already been trained for a specific task and adapting it to a new task. Usually this is a much faster and easier way forward than to train the network from scratch.

One way to overcome the earlier mentioned obstacles in acquiring training data is to use one of many standardized datasets such as VOC 2012 (Everingham et al., 2012), ImageNet, COCO, CIFAR-10, MNIST, etc. These datasets are collections of images in different classes that can be used in transfer learning. A GPR study that use this approach is Pham & Lefevre (2018), who use the CIFAR-10 for pre-training a CNN before training it with real GPR data.

Another way to accomplish a larger set of training data is to use data augmentation. This is an established method that artificially increases the size of the training dataset by randomly transforming and adding more variety to the original data. The process is performed as a part of the training process and only generates temporary data, i.e. it does not increase the actual stored dataset.

2.2.4 Training and evaluation metrics

The training of neural networks is normally done in epochs, which are instances of the full learning sequence for which the entire training dataset is provided to the net. An epoch is divided into iterations where a subset, or mini-batch, of the complete training dataset is used. After each iteration, the neuron weights and training parameters are calculated and adjusted based on the error backpropagation scheme.

Related to training and evaluation of semantic segmentation neural networks some notations and metrics are explained below.

Ground truth is the pixel labelled training dataset that defines the classes that the network is designed to predict.

Intersection over Union, IoU, is a measure of the overlap between a ground truth segment and a predicted segment, Figure 10. This means IoU can tell us whether a prediction is valid or not (Camps-Valls et al., 2021; Mathworks, n.d.). The mathematical definition of IoU is:

$$IoU = \frac{area(Predicted \cap Ground\ truth)}{area(Predicted \cup Ground\ truth)}$$

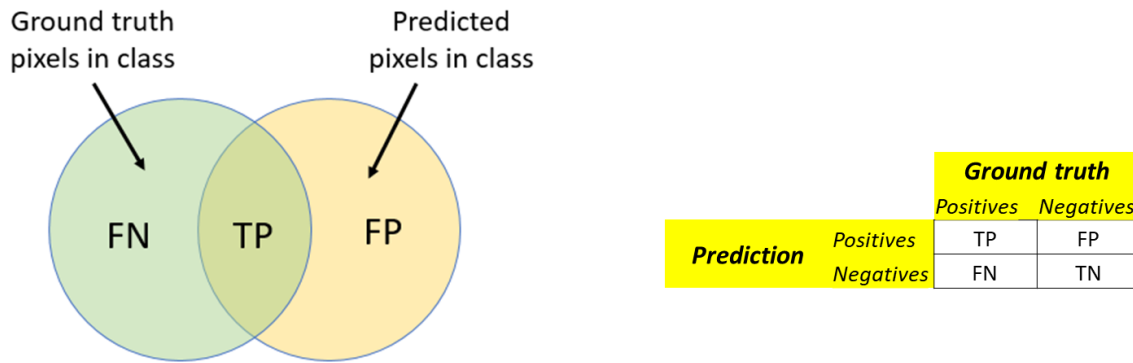


Figure 10. Definition of Intersection over Union, IoU, for semantic segmentation. FN are the False Negative pixels, i.e. ground truth pixels that are not detected. FP are the False Positive pixels, i.e. predicted pixels that does not belong to ground truth. TP are the True Positive pixels, i.e. predicted pixels that belong to ground truth. Modified from Mathworks (n.d.).

Accuracy or Precision is a measure of the percentage of correctly identified pixels for each class (Camps-Valls et al., 2021, and Mathworks, n.d.). Mathematically it is described as:

$$P = \frac{TP}{(TP + FN)}$$

where P is the Precision, TP are True Positives, and FN are False Negatives (Figure 10).

The accuracy does however not give an indication on the statistical confidence of the network in its predictions. A high confidence in predictions means that the network performs better as compared to a lower confidence. Training Loss is a measure that describes how far from a perfect prediction the network is. A definition of the loss function can be found in Bishop (2006).

The metric *Boundary F1, BF*, is a measure that correlate better than IoU with how humans assess patterns and structures of an image. It is a matching score that indicates how well the predicted boundary of each class aligns with the true boundary (Mathworks, n.d.). A definition of BF can be found in Martin et al. (2004).

2.3 Interpretation of GPR images

The interpretation of GPR images is essentially based on an understanding of the nature and origin of reflections and reflection patterns of the subsurface. Features such as the water table, sedimentary structures, and lithological boundaries are examples of subsurface discontinuities that are visible to GPR and of particular importance to sedimentology applications. There is also a known relationship, the radar stratigraphy, between primary radar reflections and the form and orientation of bedding and sedimentary structure that correlate GPR with specific sedimentary environments (Neal, 2004; van Overmeeren, 1998).

The terminology for radar stratigraphy in this report come from Neal (2004) and van Overmeeren (1998) and have analogies from seismic reflection.

To describe the overall characteristics of a GPR reflection pattern the term 'radar facies' is used. It refers to subsets or subareas of a radar image that share a common visible reflection pattern characteristic (Figure 11), a characteristic that is produced by a specific geological formation or sedimentary sequence.

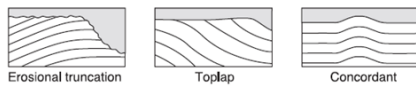
'Radar surfaces' are the boundaries of radar facies and describe the discontinuities between them (Figure 11). The radar surfaces therefore represent depositional breaks in a sedimentary sequence.

The effects on the radar response, i.e. the resulting reflection pattern, from structural and textural features of the observed subsurface are termed 'radar facies elements'. The primary radar facies elements are:

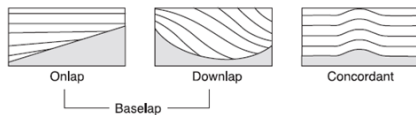
1. Reflection amplitude, controlling the intensity (grayscale or colour) of individual picture elements
2. Reflection continuity, described by category (iv) of Figure 11
3. Reflection configuration, described by category (i)-(iii) of Figure 11
4. External form (geometry) of radar facies unit, e.g. sheet, wedge, lens, mound, trough, etc

Radar surfaces

i) UPPER BOUNDARY

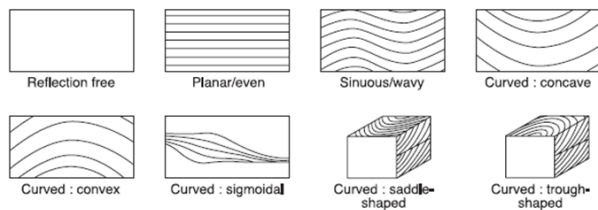


ii) LOWER BOUNDARY

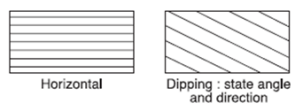


Radar facies

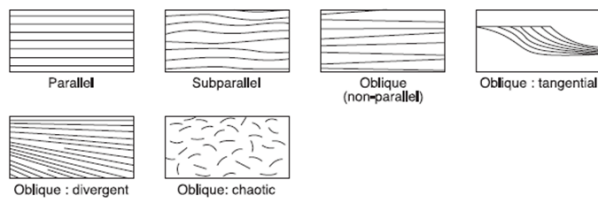
i) REFLECTION CONFIGURATION : SHAPE



ii) REFLECTION CONFIGURATION : DIP



iii) REFLECTION CONFIGURATION : RELATIONSHIP BETWEEN REFLECTIONS



iv) REFLECTION CONTINUITY

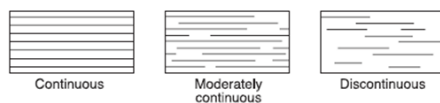


Figure 11. Picture showing the radar stratigraphy elements; radar surfaces and radar facies. Modified from Neal (2004).

3 Materials and method

3.1 Materials

Real data was acquired from a GPR survey at Askimbadet, a beach area in Göteborg, Figure 12. The survey was performed in April 2024 in the GU-course Applied Geophysics, GVG470. The sediment composition in the area according to SGU (2023) includes post glacial clay and post glacial sand. Two deformation zones intersect the area. There are also a few man-made ground constructions that involves gravel.

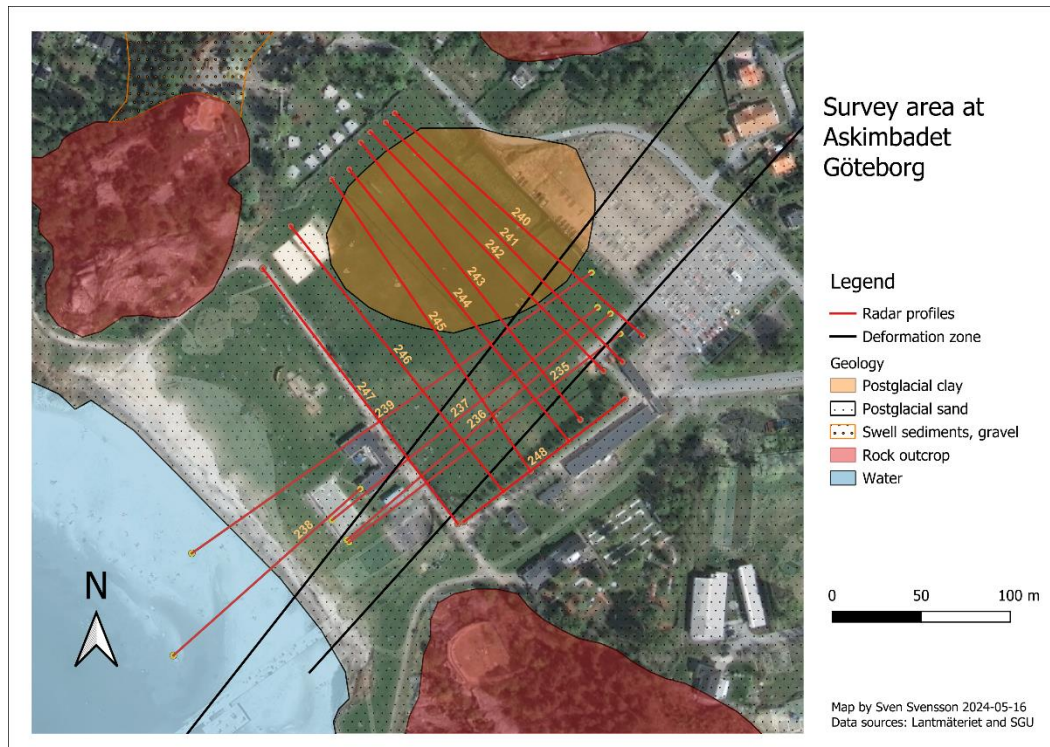


Figure 12. Map describing the survey area at Askimbadet, southwest of Göteborg. The 14 radar profiles 235-248 are drawn in red lines and are lined in NE-SW and NW-SE directions. The radar profiles mostly pass through a subsurface consisting of post glacial sand and postglacial clay, but also a few areas with walking paths based on gravel.

The complete set of survey data consists of 14 radar profiles, which were taken in two perpendicular directions, Figure 12. The lengths of the profiles were in the range of 115-204 m. The radar equipment used was of brand Malå (Guideline Geo AB, n.d.) with an integrated 250 MHz antenna. The velocity of the radar signal was estimated to 0.06 m/ns based on standard figures of moist sand, representing the ground conditions of the surveyed area. Radar images showed no or limited reflections below approximately 1.7 meters depth, at $v = 0.06$ m/ns, see Figure 13.

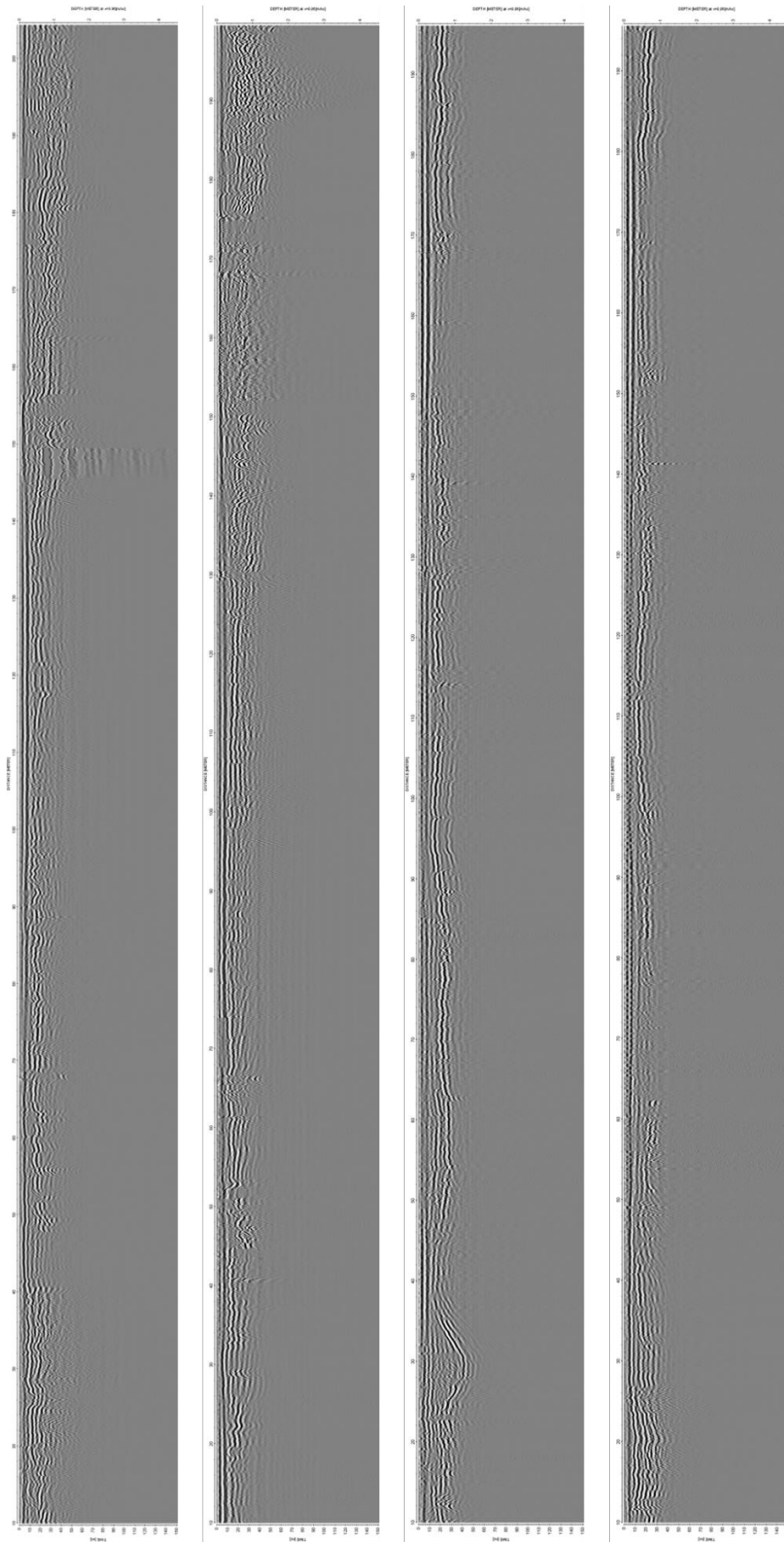


Figure 13. Examples of radar profiles from the survey at Askimbadet in April 2024. The profile id:s, from left to right, are 235, 237, 240, and 242. The equipment used is a Malå radar with an integrated 250 MHz antenna. The profiles are all ~200 m long, with a radar penetration depth of ~1.7 m, at velocity $v=0.006$ m.

For transfer learning the open VOC 2012 dataset (Everingham et al., 2012) was used. This dataset has been widely used as a benchmark for image segmentation tasks and consists of 2913 images of which each has a pixel-level segmentation annotation into one of 20 different categories, divided into the classes: Person, Animal, Vehicle, and Indoor.

The software ReflexW, v. 10.4 (Sandmeier, 1998) was used for preprocessing of raw radar data. Further, Python v. 3.11.7 was used for image preparation tasks.

MatLab R2024a and Neural Network Toolbox ver. 23.2 was used for the implementation of the neural network. The toolbox was utilized for all training, for the adaptation of the network to GPR images, for tests and for generation of results. The process of labelling pixels in the defined radar facies types was performed using the Matlab Image Labeller application.

For most processing demanding tasks, a PC running Windows 11 with an INTEL i7-12700H, 2.3MHz CPU and an NVIDIA RTX A2000 8GB GPU was used. Some of the work was also done on a PC running Windows11 with an INTEL i5-8350U 1.70 GHz CPU.

3.2 Method

The study was designed to evaluate the performance of a neural network regarding detection of sedimentology features, and the method’s strengths and limitations. Further, the design was prepared to evaluate properties of useful learning data, and the impact of the size of learning data.

Due to its strength in image processing and pattern recognition a neural network of CNN-type was the chosen method for the study. Further, the characteristics of GPR images with a composition based on radar facies elements rather than on specific objects promoted a semantic segmentation approach. This led to the choice of Deeplab v3+ semantic segmentation network (Chen et al., 2018), a model with proven performance, which is available in MatLab Neural Network Toolbox, and is used in similar work as e.g. the seismic study by Kaur et al. (2023).

The workflow described in Figure 14 was applied for the study, of which the main steps are described in the following sections.

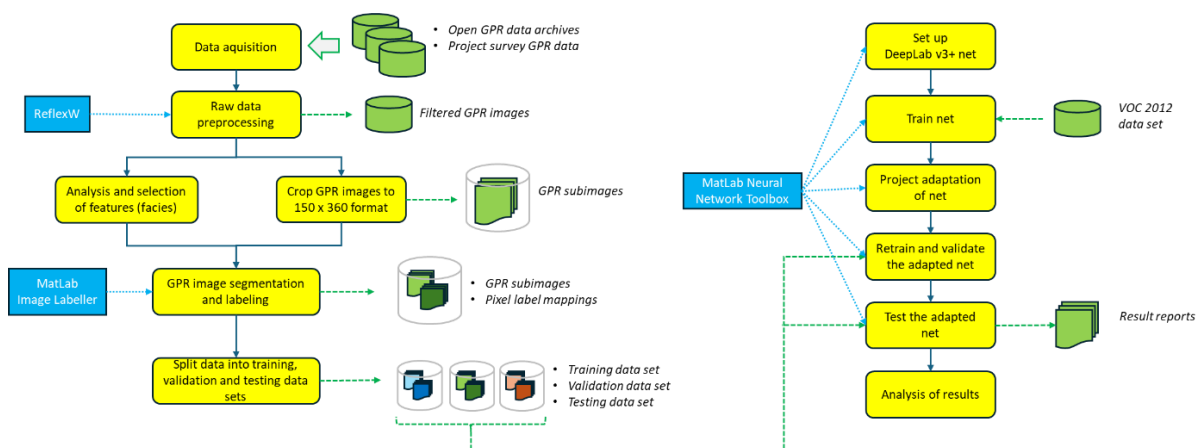


Figure 14. Figure describing the workflow used in the study. The left part describes the acquisition of data, raw data preprocessing and preparation of the training, validation, and testing datasets. The right part describes the set-up of the convolutional neural network, the training, validation, and test of the network, and finally the generation and analysis of results.

Radar data acquisition and preprocessing

Raw radar data was loaded and processed in ReflexW, (Sandmeier, 1998), with the filtering options: *Energy decay*, *Subtract-mean (Dewow)*, and *Move start time*. Energy decay was used to compensate for the natural physical attenuation of the radar signal during propagation, and in this way weaker reflections at increasing depth are gradually amplified. Dewow was used to reduce possible low-frequency noise in the data. Move start time was used for static correction in time of the received signal, i.e. to move the starting point of the signal. The velocity of the radar signal was estimated to 0.06 m/ns based on standard figures of moist sand, representing the ground conditions of the surveyed area. The filtering resulted in radar images with improved readability and resolution as compared with raw data.

Analysis and facies selection

This step involved the manual task of analysing the filtered images and identifying candidates of primary radar facies in order to assign labels to the input data. In this process each image subarea that was found to be representative for a specific radar facies was judged on its clarity of appearance. Seven radar facies types were then selected based on frequency of occurrence and average clarity of appearance, Figure 15.

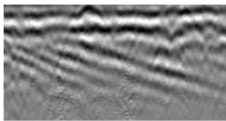

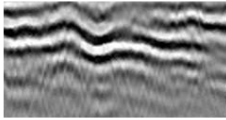

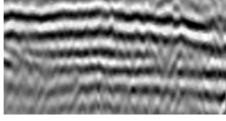

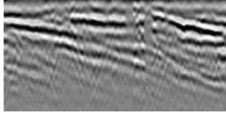



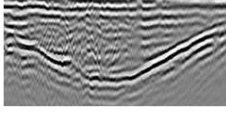

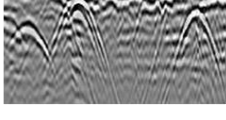

Radar image	Tracing	Facies description	Interpretation
		Dipping 20-30°. Low to medium amplitude. Continuous.	Cross bedded river dune sediments. Sand.
		Sinuous/wavy. Medium to high amplitude. Continuous/moderately continuous.	Stream and river sediments. Sand and gravel.
		Subparallel. Medium amplitude. Moderately continuous.	Cover sands.
		Oblique – tangential. Low to medium amplitude. Moderately continuous.	Cross bedded estuarine deposits. Sand.
		Oblique – chaotic. Medium to high amplitude.	Unsorted, mixed material. Sand and gravel.
		Curved concave/convex. Medium to high amplitude. Continuous/moderately continuous.	Channel shapes (convex). Beach ridges (concave). Sand.
		Hyperbola	Boulders

Figure 15. Picture showing the stratigraphy classification of radar facies in the study. The description is based on Neal (2004), and the interpretation is based on van Overmeeren (1998).

Cropping of images

This step was to cut up the original radar images into smaller sub images. The reason was to fit the selected size of the input layer in the Deeplab v3+ net. In addition this simplified the pixel labelling procedure by limiting the area to work on. Since the original images showed no or limited reflections below the upper 150 rows of pixels (this corresponds approximately to 1.7 meters depth at an estimated velocity of $v = 0.06$ m/ns) the cropping was limited to this part. The resulting sub image size was 150x360 pixels (Figure 16). Cropping and generation of sub image files was made with a Python script (Appendix 2). In total 143 images were generated.

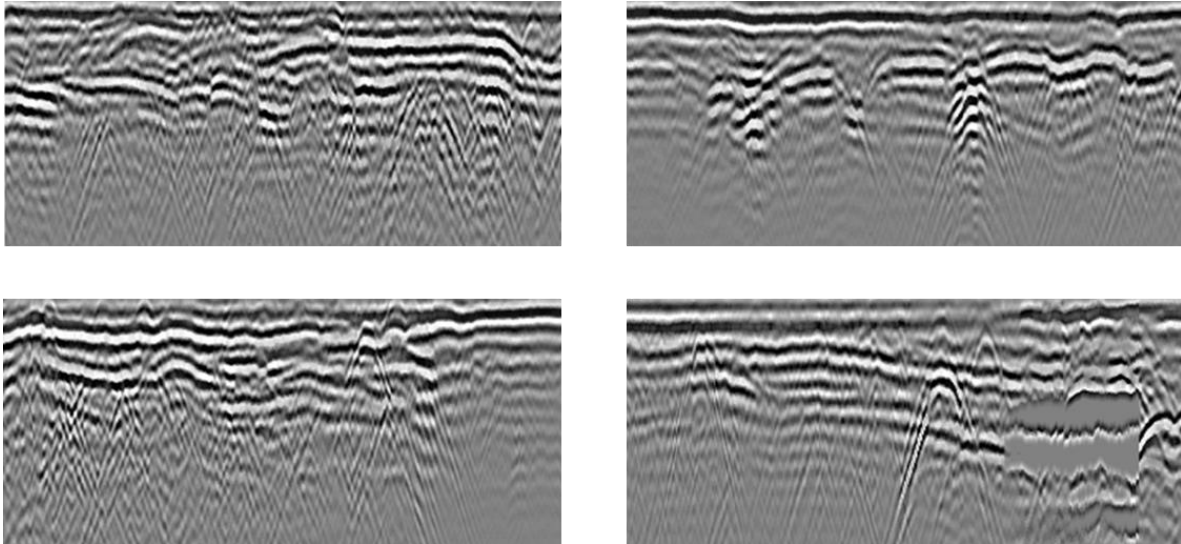


Figure 16. Examples of images cropped into size 150x360 pixels that are used to generate the ground truth dataset for training and validation, and for the test data set.

Semantic segmentation and generation of ground base datasets

The outcome of the previous analysis and facies selection was used to pick out representative images for semantic segmentation. To enable an evaluation of what properties are useful for learning and the impact of the size of learning data, three different datasets, DS1, DS2, and DS3, were created, see Table 1. These datasets are based on the facies types in Figure 15. The distribution of classes within the datasets are depicted in Figure 17

Table 1. Generated semantic segmentation datasets, DS1, DS2, and DS3, together with the radar facies that are included in each dataset and the number of images generated.

Dataset id	Facies included (from Figure 15)	No of images	Comments
DS1	<i>dipping</i> <i>hyperbola</i>	20	
DS2	<i>dipping</i> <i>oblique-tangential</i> <i>hyperbola</i> <i>other (remaining parts)</i>	44	Facies types <i>dipping</i> and <i>oblique-tangential</i> are both classified as <i>dipping</i> based on similarities. Parts of images that are not within the included facies are classified as <i>other</i> .
DS3	All except <i>hyperbola</i>	43	

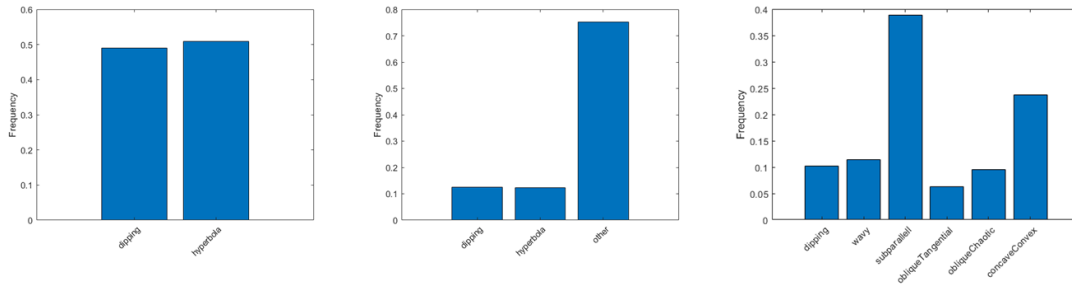


Figure 17. Class distributions of datasets. From left to right; DS1, DS2, and DS3.

The generation of ground base datasets for semantic segmentation was performed by labelling image pixels according to the defined classes in Table 1. An example of this is shown in the screenshot from Matlab image Labeller in Figure 18. When labelling was completed, each image dataset (DS1, DS2 and DS3) was split into three, for training, validation, and testing, with a share of 60%, 20%, and 20% respectively.

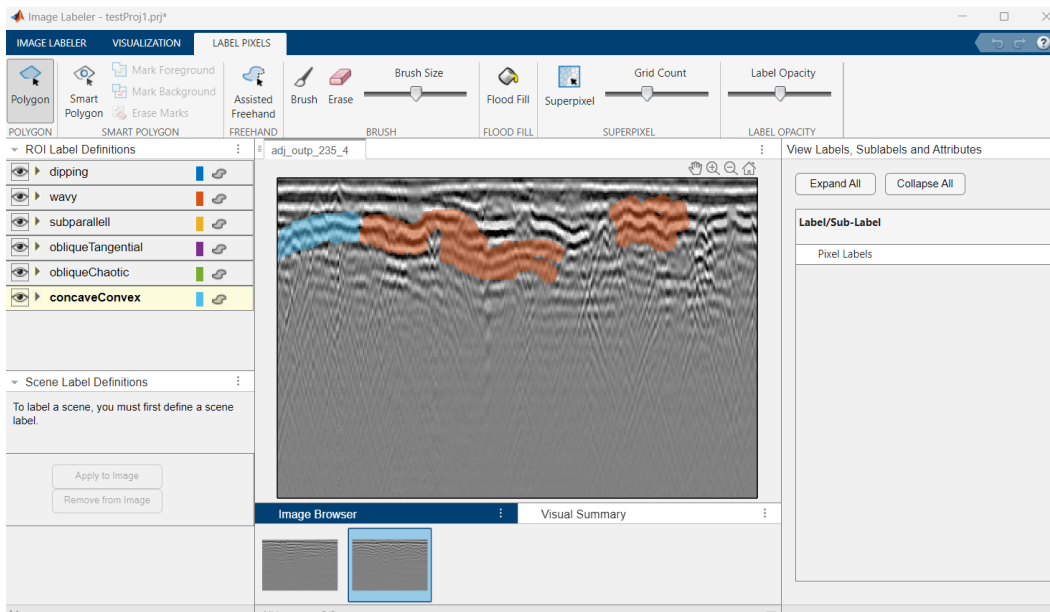


Figure 18. Screenshot describing the application Matlab Image labeller. The class labels are shown to the left.

Deeplab v3+ training with VOC 2012

To prepare for transfer learning, the VOC 2012 dataset was downloaded from Everingham et al. (2012) and training of the Deeplab v3+ network was performed using the Matlab script from Itakura (2019).

Transfer learning with real data

In this step the Deeplab v3+ net was first adapted to the requirements of the actual project. This involved modifying the output layer of the network to match the classes to be categorized. Next, to increase the variety of training data, data augmentation was performed on the real dataset.

As is evident from Figure 17, the classes in the datasets DS2 and DS3 are not evenly distributed. To improve training, the classes were therefore balanced, an operation that was made to flatten the distribution among the labelled pixels. The adapted net was then trained and validated using the real datasets, (see Matlab script in Appendix 1).

Test of the adapted net and final analysis

The net was finally tested and evaluated. This was done by conducting three separate experiments. In each experiment, metrics in form of accuracy, loss, BF and IoU was generated (see Matlab script in Appendix 1) and manual analysis of the results was made.

4 Results

This section presents results from training and test of the neural network for each of the three experiments that were conducted on the datasets DS1, DS2, and DS3 respectively. The section is separated into; 1) training performance, which describes the progress of the training together with accuracy and loss, 2) prediction performance, which presents confusion matrices that describes how well the net predicts each class, 3) general metrics that shows accuracy, IoU, and mean BF score for the classes, and 4) prediction views of test images.

4.1 Training performance

Resulting performance for training is presented in Figure 19, Figure 20, and Figure 21, for the three experiments with datasets DS1, DS2, and DS3, respectively. In general, the training accuracy and loss curves flattens out well before the final epoch of training. The final validation accuracy is moderate to low, and decreases with increasing number of classes, i.e. highest accuracy for the dataset DS1 and lowest accuracy for DS3. Final validation loss is low for all experiments, in the range of 0.04-0.62.

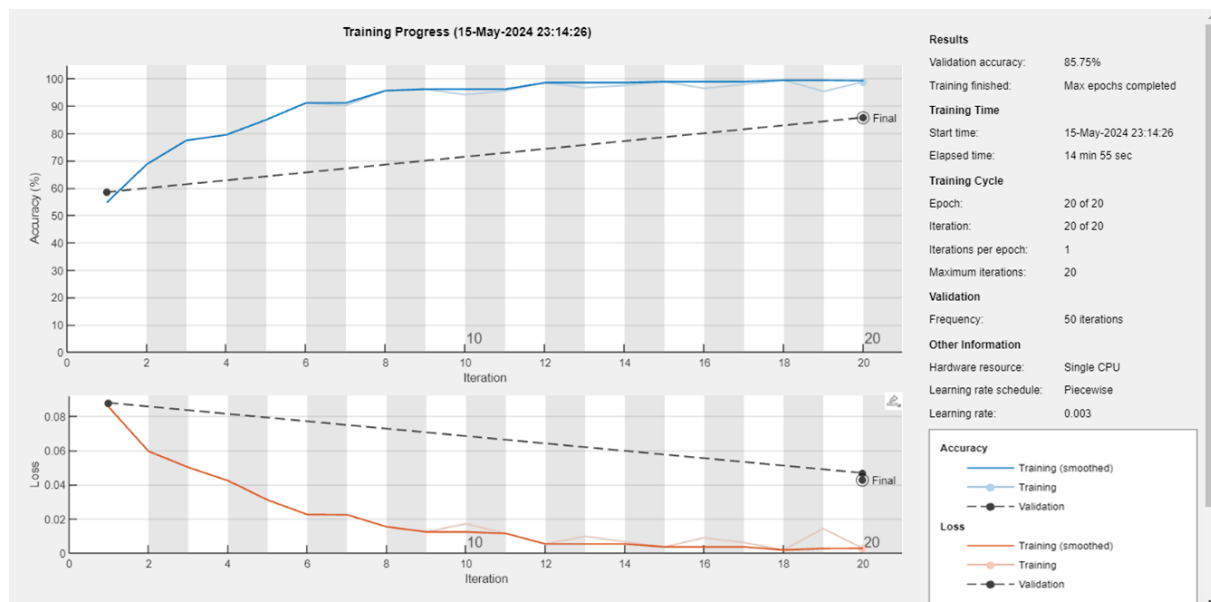


Figure 19. Training performance for dataset DS1 over 20 iterations of training. The top graph shows the Accuracy, with a slowly increasing value until a final validation accuracy of 85.75% is reached. The bottom graph shows the Loss, with a slowly decreasing value until a final validation loss of 0.04 is reached. Loss is flattening out after ~12 iterations. Training was performed on a single CPU, taking ~15 minutes.

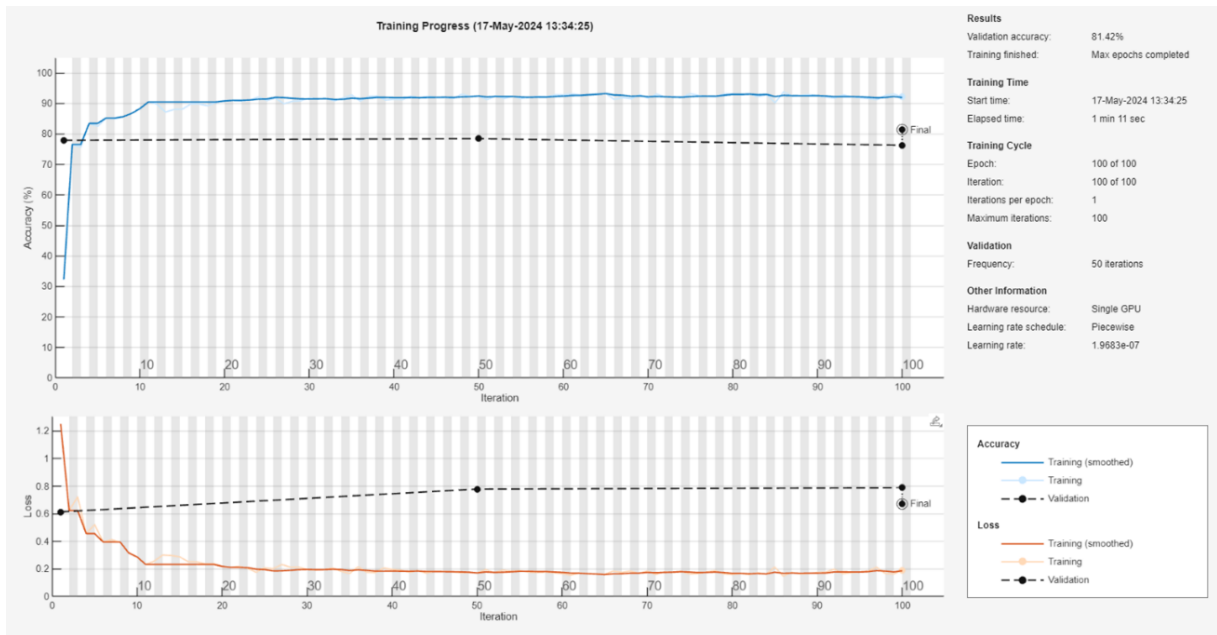


Figure 20. Training performance for dataset DS2 over 100 iterations of training. The top graph shows the Accuracy, with a quick step from start, and then a slow increase until a final validation accuracy of 81.42% is reached. The bottom graph shows the Loss, with a quick drop from start, and then a slow decrease until a final validation loss of 0.62 is reached. Loss is flattening out after ~25 iterations. Training was performed on a GPU, with 100 iterations taking ~1 minutes.

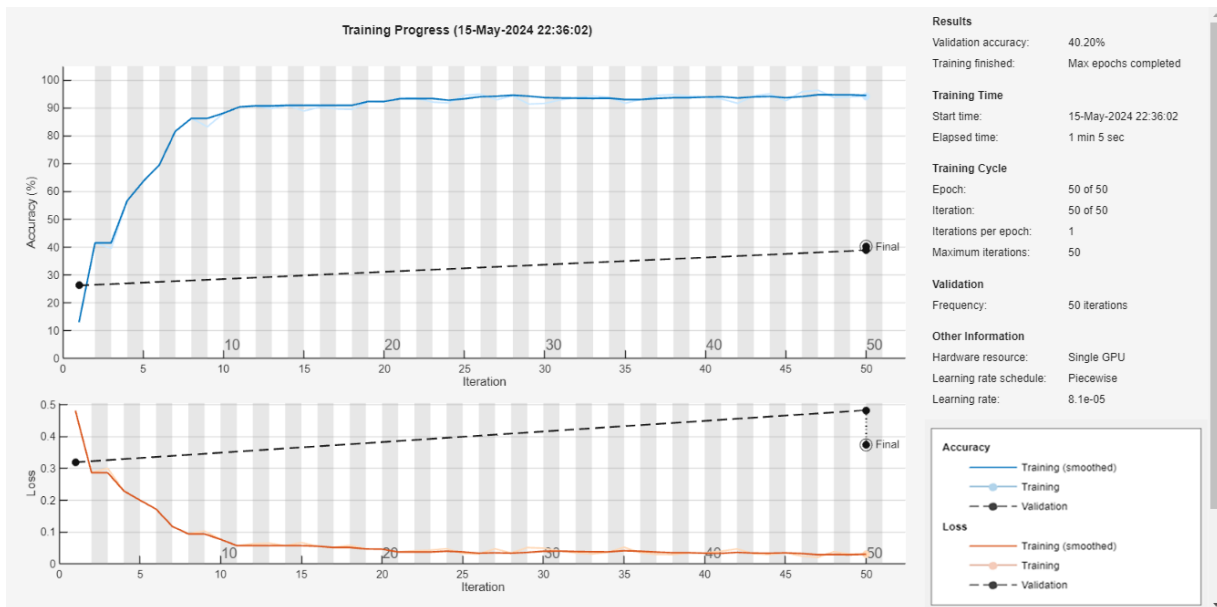


Figure 21. Training performance for dataset DS3 over 50 iterations of training. The top graph shows the Accuracy, with a slowly increasing value until a final validation accuracy of 40.2% is reached. The bottom graph shows the Loss, with a slowly decreasing value until a final validation loss of 0.38 is reached. Loss is flattening out after ~20 iterations. Training was performed on a GPU, with 50 iterations taking ~1 minutes.

4.2 Prediction performance

Prediction performance is presented by the confusion matrices in Figure 22 and Figure 23. The matrices show the scores for true positives (TP) and false positives (FP) for each class. TP represents correct detection of the network with respect to ground truth, whereas FP represents false detections. In general, the increase of classes from experiment 1 to experiment 3 has a negative impact on the performance.

For experiment 1, on dataset DS1, the score is >90% for true positives (TP) and <10% for false positives (FP).

The TP-scores for experiment 2, on dataset DS2, is <50% for the classes of interest, i.e. 'dipping' and 'hyperbola'. The FP-score within these two classes is low, <2%, but >50% when the class 'other' is included.

For experiment 3, on dataset DS3, the overall performance is low. Only the class 'subparalell' has a TP-score above 50%. The remaining classes have TP-scores in the range of 21-38%, except for 'oblique-tangential' that has no TP detected at all. Most FP detections fall into the class 'concave-convex' with as high as 71% for the class 'wavy'. Significant is also the confusion to the class 'subparalell' for the classes 'dipping' and 'oblique-chaotic', with FP-scores of 34% and 45% respectively.

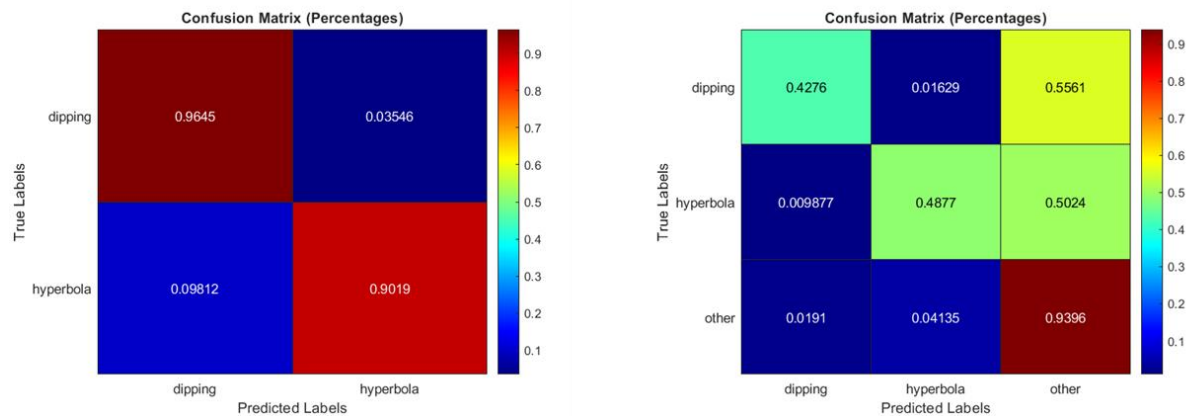


Figure 22. Confusion matrix describing prediction performance for each class with dataset DS1 to the left and dataset DS2 to the right. The falling diagonal represents correct predictions. For the DS1 dataset the correct prediction for the class 'dipping' is ~96%, and for the class 'hyperbola' ~90%. Wrong predictions are ~4% for 'dipping' and ~10% for 'hyperbola'. For the DS2 dataset the correct prediction for the class 'dipping' is ~43%, for 'hyperbola' ~49%, and for 'other' ~94%. Wrong predictions for the 'dipping' and 'hyperbola' classes are significant to the class 'other', with ~56% and ~50% respectively. Remaining wrong predictions are relatively low, in the range of ~4% to <1%.

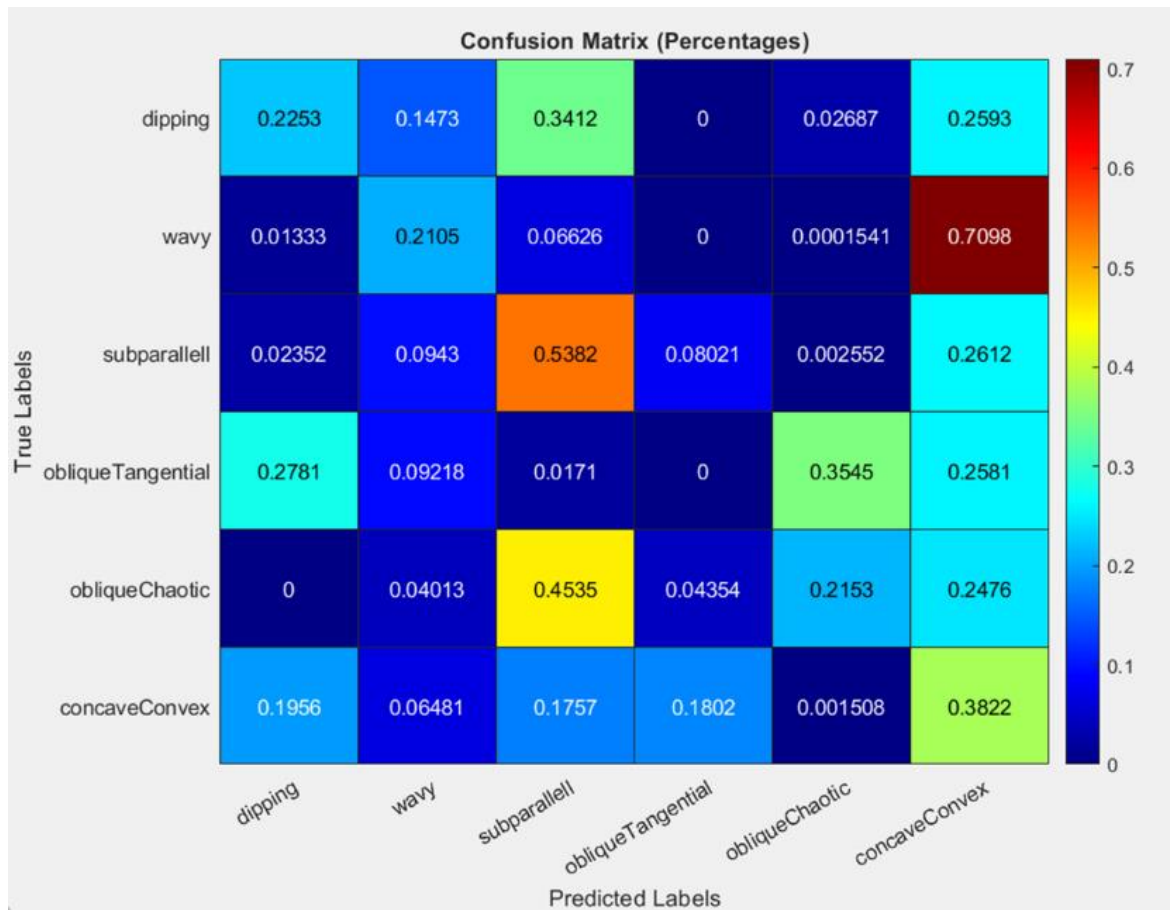


Figure 23. Confusion matrix describing prediction performance for each class of dataset DS3. The falling diagonal represents correct predictions that are generally low for all classes, in the range of ~54% down to 0%. Wrong predictions are generally high for all classes, with 9 cells in the matrix indicating >20%.

4.3 General metrics

The general metrics in form of accuracy, IoU, and mean BF score is presented in Figure 24. The accuracy metrics was also presented in the previous section.

In general, the increase of classes from experiment 1 to experiment 3 has a negative impact on the IoU. Both classes in experiment 1 have an IoU >80%. In experiment 2, the classes of interest, 'dipping' and 'hyperbola', have an IoU of ~40%. Experiment 3 has an overall low IoU, with most classes having an IoU <20%.

The mean BF score shows no visible correlation with the number of classes, and neither with IoU or the accuracy.



Figure 24. Metrics for each class in the datasets DS1, DS2, and DS3. The graphs show the accuracy, the IoU, and the mean BF score. In general, the results of DS1 show the highest values and the lowest spread among the classes.

4.4 Prediction views of test images

To get a visual understanding of the neural network prediction results, eight combined images from experiment 2 are presented in Figure 25. The original images (left in each sub image) are from the test dataset of the experiment and include various radar facies of which the main classes 'dipping' and 'hyperbola' are predicted by the net (remaining segments are predicted as 'other'). The figures demonstrate the fact that the prediction accuracy for the 'dipping' and 'hyperbola' classes is in the range of 40-50%. Some of the facies patterns are detected (TP), but there are also patterns that are not detected (FN). In the pictures, can also be found examples of wrong predictions (FP).

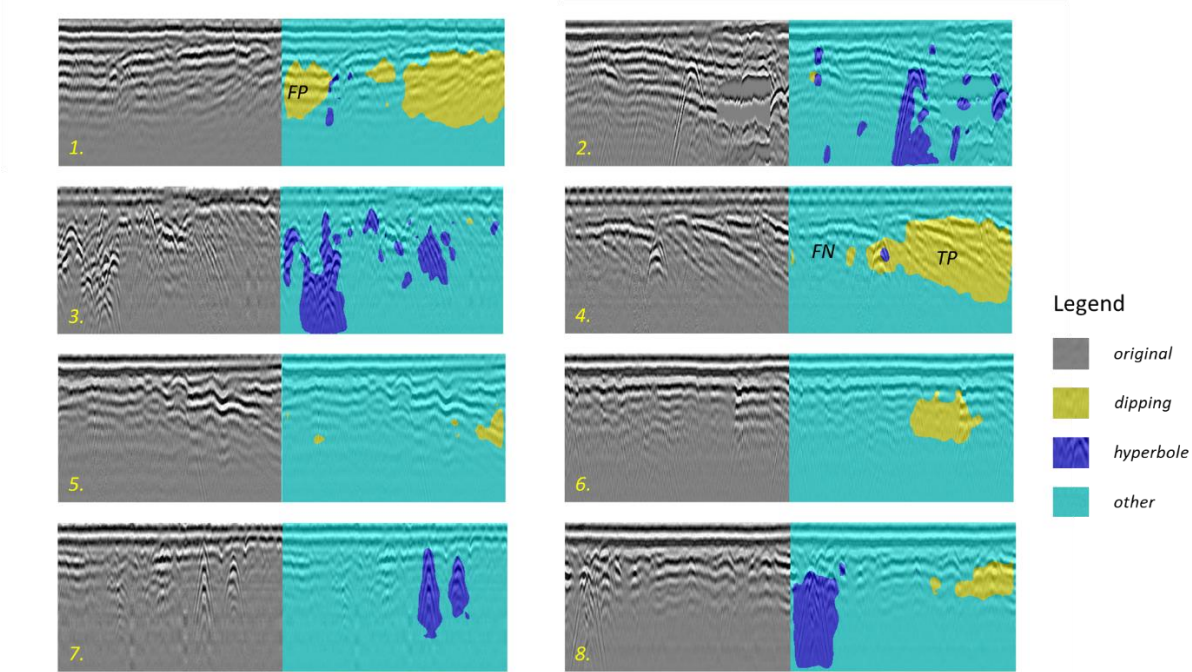


Figure 25. Prediction view of images from experiment 2, with dataset DS2. Each of the eight sub images shows a combined view, with the original image from the training dataset to the left, and the predicted output from the neural network to the right. In sub images 1 and 4 examples of areas with TP, FP, and FN predictions are indicated.

5 Discussion

In the study performed it is clear that the availability of substantial and representative data for training and validation is a key factor for reaching relevant performance of a neural network. This is well established knowledge from general research in deep learning and therefore was a starting position of this work. However, to find high quantity of relevant GPR data is difficult, which was evident in the study.

In general, the results show that the Deeplabs v3+ convolutional neural network has the capability to detect various radar facies (Figure 15). However, the detection was far from perfect, with a substantial amount of fault detections, which is visualized in Figure 25. Further, the three experiments that were conducted indicated a performance drop of the network with increasing number of classes to predict. A reasonable explanation for this is that the limited amount of data, suffers from the split into an increasing number of classes to predict.

The amount of GPR data was limited and acquired from only one site (Askimbadet). Although seven radar facies types could be defined based on the original radar images, the representation of them in the full material was low. Together, these factors can explain the rather low precision metrics observed in the study. A way to describe this is: “a neural network will only be able to identify facies types associated with GPR reflection patterns that are well represented in the training data”, Moysey et al. (2003).

Apart from the amount of data and how well it represents the properties of interest, we can ask if we use the right data to train the network with. I.e., what is the quality of the data we use? In essence this relates to the subjective interpretation made during pixel labelling. In this study different radar facies were defined based on identified GPR image subsets that each share a common reflection pattern or characteristic (Figure 15). However, during the work of categorizing images with pixel labelling, distinct subsets were not always easily identified. The consequence could be that some inconsistency was introduced to the dataset, which eventually led to lower quality of the training data, and lower precision in the network prediction.

Comparing the precision metrics of this study with other GPR studies, we can find similarities and differences. Moysey et al. (2003) reports numbers in the range 44-80% for real data, of which the lower numbers are in the same level as our results from experiment 2 on dataset DS2 (42% and 48%). Compared with our experiment 1 on dataset DS1 with a precision of 90% and 96%, the higher numbers reported by Moysey et al. (2003) are below our results. Note however that the DS1 dataset have two classes as compared with four in Moysey et al. (2003), which could have an impact. The study Moysey et al. (2006) reports 42% for one of the facies, but also numbers, 60%-98% that are well above our experiment 2 results. The impact from dataset sizes cannot be judged since no information is given by Moysey et al. Note also that in Moysey et al. (2003) synthetic radar images are used for training.

Compared with the seismic studies, by Kaur et al. (2023), and Wrona et al. (2018), with reported precision metrics of ~90% and higher, the results from our study are low. Two reasonable explanations for this difference is: 1) the huge amount of data used in the seismic studies, and 2) the coarse resolution and the nature of spatial consistency of seismic facies that produce clear distinctions and features over large image areas. The difference in the facies characteristics between GPR and seismics is of physical nature and therefore a factor that could not be improved. However, an increase in the amount of GPR data could probably reduce this difference.

To enhance the learning process, transfer learning with the VOC 2012 (Everingham et al., 2012) dataset was initially used. However, this approach was abandoned early in the work when it was obvious that transfer learning was redundant. The reason was the small time that was needed for training from scratch of our network for GPR images. With increased amount of data, transfer learning is probably still a good way to increase efficiency and save time in the learning process.

Based on the available data, from a beach environment with mostly sands and clays, a sedimentology approach was chosen for this study. Since the network is trained for the defined radar facies for sediments it is not possible to reuse it without modifications for other ground environments (like palsas, glaciers, or gravel and bedrock dominated ground). However, the network can be used for transfer learning by adding new classes and reusing parts of the training data. This would be beneficial as the network then will cover more ground environments.

Further improvements of a deep learning approach for GPR could be to generate specific datasets with known features. One way to accomplish this is to use synthetic datasets (Pham & Lefevre, 2018; Moysey et al., 2003; Küçükdemirci & Sarris, 2022). This approach is both a way to define the training data, as well as to increase the amount of training data. Another way could be to construct a physical ground model by building a geophysical test site or 'sandbox' with predefined layers and objects of interest. This principle is used by Dérobert & Pajewski (2018) in a study for civil engineering applications.

Another area of improvement is related to reducing the subjectiveness in interpretation. The current method for interpretation of GPR images is manual and highly subjective. It is essentially based on the individual expert's skills and ability to recognize reflection patterns and other characteristics in the image (Moysey, 2006; Bristow & Jol, 2003). In a future scenario, big GPR data could become freely available. That would provide an opportunity for the research community to standardize and categorize radar facies in a joint effort that eventually will remove the subjectivity aspect. A standardized dataset for GPR images could in this way be compiled, in a manner like the databases existing today for deep learning, e.g., VOC 2012 (Everingham et al., 2012), ImageNet, COCO, CIFAR-10, MNIST, etc.

6 Conclusions

Despite the varying performance resulting from the study, it has shown that deep learning is a feasible way to automate GPR image interpretation. There is however a major obstacle in acquiring substantial and representative data for the learning process, to achieve reasonable performance of the neural network. Nevertheless, even with limited datasets as is used in this study, we get an indication that neural network prediction can be made to some extent.

Further, a trained neural network for interpreting GPR images is an effective solution compared with the current manual and highly subjective method. A well trained neural network will also not add on subjectivity in the interpretation of several images. However, producing the training data for neural networks is still a time consuming and subjective process, although it is a one-time effort. The quality of the training data is essential to achieve high performance in the predictions. The resulting output from the neural network is never better than the quality of the training data.

7 Acknowledgements

I would like to thank Martin Persson for bringing up the idea of this project and for good discussions on GPR and practical survey topics. Further, thanks to Heather Reese for valuable comments and suggestions on the report. I would also like to thank my classmates that provided early valuable comments when proof-reading the report.

8 References

- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, 1–6. <https://doi.org/10.1109/ICEngTechnol.2017.8308186>
- Aradóttir, N., Benediktsson, I. O., Ingólfsson, O., Sturkell, E., Brynjólfsson, S., Farnsworth, W. R., & Phillips, E. (2022). Drumlin formation within the Bustarfell drumlin field, northeast Iceland: integrating sedimentological and ground-penetrating radar data. *Journal Of Quaternary Science*, *2022*, *38*(3), 386–402.
- Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Bristow, C.S., & Jol, H. M. (2003). *Ground penetrating radar in sediments* (Vol. 211). Geological Society.
- Camps-Valls, G., Tuia, D., Zhu, X. X., & Reichstein, M. (2021). *Deep learning for the earth sciences : a comprehensive approach to remote sensing, climate science and geosciences*. Wiley.
- Camunas-Mesa, L. A., Linares-Barranco, B., & Serrano-Gotarredona, T. (2019), Neuromorphic Spiking Neural Networks and Their Memristor-CMOS Hardware Implementations. *Materials*, *12*, 2745.
- Chen, L-C., Zhu, Y., Papandreou, G., Schroff, F., & Hartwig, A. (2018). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *arXiv.org*. <https://doi.org/10.48550/arxiv.1802.02611>
- Costas, S., Alejo, I., Rial, F., Lorenzo, H., & Nombela, M. A. (2006). Cyclical evolution of a modern transgressive sand barrier in northwestern Spain elucidated by GPR and aerial photos. *Journal of Sedimentary Research*, *76*(9), 1077–1092. <https://doi-org.ezproxy.ub.gu.se/10.2110/jsr.2006.094>
- Dérobert, X., & Pajewski, L. (2018). TU1208 Open Database of Radargrams: The Dataset of the IFSTTAR Geophysical Test Site. *Remote sensing*, *10*(4), 530. <https://doi.org/10.3390/rs10040530>
- Everingham, M., Van-Gool, L., Williams, C. K. I., Winn, J., Zisserman, A. (2012). *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>
- Forde, A.S., Smith, C.G., & Reynolds, B.J. (2016). Archive of ground penetrating radar data collected during USGS field activity 13BIM01—Dauphin Island, Alabama, April 2013. *U.S. Geological Survey Data Series*, *982*. <https://dx.doi.org/10.3133/ds982>
- Fu, T., Wu, Y., Tan, L., Li, D., & Wen, Y. (2019). Imaging the structure and reconstructing the development of a barchan dune using ground-penetrating radar. *Geomorphology (Amsterdam, Netherlands)*, *341*, 192–202. <https://doi.org/10.1016/j.geomorph.2019.05.014>
- Howard, F.J.F., McPherson, A.A. (n.d.), Ground Penetrating Radar (GPR) Data - Old Bar Beach Survey, Downloaded 24-05-05 from <http://www.ga.gov.au/metadatagateway/metadata/record/100224>
- IBM. (n.d.a). *What is artificial intelligence (AI)?*. Downloaded 2024-04-04 from <https://www.ibm.com/topics/artificial-intelligence>
- IBM. (n.d.b). *What are convolutional neural networks?*. Downloaded 2024-04-04 from <https://www.ibm.com/topics/convolutional-neural-networks>

- IBM. (n.d.c). *What is transfer learning?*. Downloaded 2024-04-04 from <https://www.ibm.com/topics/transfer-learning>
- IBM. (n.d.d). *What is object detection?*. Downloaded 2024-05-02 from <https://www.ibm.com/topics/object-detection>
- Itakura, K. (2019). *Semantic Segmentation Using Pascal-VOC dataset*. Downloaded 2024-05-02 from <https://github.com/Kentaltakura/Semantic-segmentation-using-Pascal-VOC-with-MATLAB>
- Jol, H.M. (2008). *Ground Penetrating Radar Theory and Applications* (1st ed.). Elsevier.
- Kaur, H., Pham, N., Fomel, S., Geng, Z., Decker, L., Gremillion, B., Jervis, M., Abma, R., & Gao, S. (2023). A deep learning framework for seismic facies classification. *Interpretation (Richmond)*, 11(1), T107–T116. <https://doi.org/10.1190/INT-2022-0048.1>
- Knime. (2022). *Convolutional Neural Networks & Computer Vision*. Downloaded 2024-04-08 from <https://www.knime.com/blog/convolutional-neural-networks-computer-vision>
- Küçükdemirci, M. & Sarris, A. (2022). GPR Data Processing and Interpretation Based on Artificial Intelligence Approaches: Future Perspectives for Archaeological Prospection. *Remote Sensing (Basel, Switzerland)*, 14(14), 3377. <https://doi.org/10.3390/rs14143377>
- Liu, Z., Cao, J., Lu, Y., Chen, S. & Liu, J. (2019), A seismic facies classification method based on the convolutional neural network and the probabilistic framework for seismic attributes and spatial classification. *Interpretation*, 7(3), SE225–SE236, doi: [10.1190/INT-2018-0238.1](https://doi.org/10.1190/INT-2018-0238.1).
- Magalhães, A. J. C., Lima-Filho, F. P., Guadagnin, F., Silva, V. A., Teixeira, W. L. E., Souza, A. M., Raja Gabaglia, G. P., & Catuneanu, O. (2017). Ground penetrating radar for facies architecture and high-resolution stratigraphy: Examples from the Mesoproterozoic in the Chapada Diamantina Basin, Brazil. *Marine and Petroleum Geology*, 86, 1191–1206. <https://doi.org/10.1016/j.marpetgeo.2017.07.027>
- Guideline Geo AB. (n.d.). *Malå*. Downloaded 2024-05-03 from <https://www.guidelinegeo.com/mala-ground-penetrating-radar-gpr/>
- Martin, D.R., Fowlkes, C. C., & Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5), 530–549. <https://doi.org/10.1109/TPAMI.2004.1273918>
- Mathworks (n.d.). Help Center. Downloaded 2024-05-19 from https://se.mathworks.com/help/vision/ref/evaluatesemanticsegmentation.html?s_tid=doc_ta
- Milsom, J., & Eriksen, A. (2011). *Field geophysics*. Oxford : Wiley-Blackwell.
- Moysey, S., Caers, J., Knight, R. J., & Allen-King, R. M. (2003). Stochastic estimation of facies using ground penetrating radar data. *Stochastic Environmental Research and Risk Assessment*, 17(5), 306–318. <https://doi.org/10.1007/S00477-003-0152-6>
- Moysey, S., Knight, R. J., & Jol, H.M. (2006) Texture-based classification of ground-penetrating radar images, *Geophysics.*, (71), 111. <https://doi.org/10.1190/1.2356114>
- Neal, A. (2004). Ground-penetrating radar and its use in sedimentology; principles, problems and progress. *Earth-Science Reviews*, 66(3-4), 261–330. <https://doi.org/10.1016/j.earscirev.2004.01.004>
- NVIDIA (2024). *Artificial Neural Network*. Downloaded 2024-04-09 from <https://developer.nvidia.com/discover/artificial-neural-network>

- Peemen, M. C. J. (2017). *Improving the efficiency of deep convolutional networks*. [Phd Thesis, Electrical Engineering, Technische Universiteit Eindhoven].
- Pham, M-T., & Lefevre, S. (2018). Buried Object Detection from B-Scan Ground Penetrating Radar Data Using Faster-RCNN. *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, 2018*, 6804–6807. <https://doi.org/10.1109/IGARSS.2018.8517683>
- Ribolini, A., Bertoni, D., Bini, M., & Sarti, G. (2021). Ground-Penetrating Radar Prospections to Image the Inner Structure of Coastal Dunes at Sites Characterized by Erosion and Accretion (Northern Tuscany, Italy). *Applied Sciences*, 11(23), 11260. <https://doi.org/10.3390/app112311260>
- Sandmeier, K.J. (1998). *REFLEX W 32 and 64 bit Version 10.4*. Downloaded 2024-04-04 from <https://www.sandmeier-geo.de/reflexw.html>
- SGU (2023). *Kartvisaren Jordarter 1:25 000-1:100 000*. <https://apps.sgu.se/kartvisare/kartvisare-jordarter-25-100.html>
- Shan, X., Yu Xinghe, Y., Clift, P. D., Tan, C., Jin, L., Mingtao, L. & Wen, L. (2015). The ground penetrating radar facies and architecture of a paleo-spirit from Huangqihai Lake, north China; implications for genesis and evolution. *Sedimentary Geology*, 323, 1–14. <https://doi.org/10.1016/j.sedgeo.2015.04.010>
- Thieler, E.R., Foster, D.S., Mallinson, D.J., Himmelstoss, E.A., McNinch, J.E., List, J.H., & Hammar-Klose, E.S. (2013). Quaternary geophysical framework of the northeastern North Carolina coastal system. *U.S. Geological Survey Open-File Report 2011–1015*. <https://pubs.usgs.gov/of/2011/1015/>.
- Travassos, X. L., Avila, S. L., & Ida, N. (2021). Artificial Neural Networks and Machine Learning techniques applied to Ground Penetrating Radar: A review. *Applied Computing & Informatics*, 17(2), 296–308. <https://doi.org/10.1016/j.aci.2018.10.001>
- Van Overmeeren, R. A. (1998). Radar facies of unconsolidated sediments in the Netherlands; a radar stratigraphy interpretation method for hydrogeology. *Journal of Applied Geophysics*, 40(1-3), 1–18. [https://doi.org/10.1016/S0926-9851\(97\)00033-5](https://doi.org/10.1016/S0926-9851(97)00033-5)
- Woodard, J. B., Zoet, L. K., Benediktsson, Í. Ö. et al. (2020). Insights into drumlin development from ground-penetrating radar at Múlajökull, Iceland, a surge-type glacier. *Journal of Glaciology*, 66, 822–830. <https://doi.org/10.1017/jog.2020.50>
- Wrona, T., Pan, I., Gawthorpe, R. L., & Fossen, H. (2018). Seismic facies analysis using machine learning. *Geophysics*, 83(5), 111.
- Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights Imaging*, 9, 611–629. <https://doi.org/10.1007/s13244-018-0639-9>
- Zhang, W., Li, H., Li, Y., Liu, H., Chen, Y., & Ding, X. (2021). Application of deep learning algorithms in geotechnical engineering: a short critical review. *The Artificial Intelligence Review*, 54(8), 5633–5673. <https://doi.org/10.1007/s10462-021-09967-1>
- Zhao, T. (2018). Seismic facies classification using different deep convolutional neural networks, 2018 *SEG Annual Meeting*.

Appendix 1

MatLab code for setting up, training, validation and testing of the neural network.

Main script semanticSegmTest

```
%-----  
% Name: semanticSegmTest  
% Revision: C  
% Date: 2024-05-16  
% Author: Sven S  
% Description:  
% This script is a modified version of the script from Itakura, K. (2019).  
% "Semantic Segmentation Using Pascal-VOC dataset", https://github.com/  
% KentaItakura/Semantic-segmentation-using-Pascal-VOC-with-MATLAB,  
% and the Matlab help document "Train and Deploy Fully Convolutional  
% Networks for Semantic Segmentation", https://se.mathworks.com/help/  
% deeplearning/ug/train-and-deploy-fully-convolutional-networks-for-  
% semantic-segmentation.html  
%  
% The script assumes imagedata is residing in subdirectory imgData and  
% pixel label data in pixelLabelData. Labels definitions are assumed to be  
% in the file labelDefinitions.mat, residing in the archive folder. See  
% directory structure below.  
%  
% + MATLAB Drive\subproject\archive---+-labelDefinitions.mat  
%                               I  
%                               +---pixelLabelData  
%                               +---imgData  
%-----  
  
% This version uses recategorized images from Archive6 using label  
% categories: 'dipping', 'hyperbola' and 'others'.  
  
% Clean environment  
clear;clc;close all  
  
% Directory path assuming current dir is MATLAB drive  
projPath=cd;  
  
% Set subproject  
whichSubproj='askimProj';  
whichArchive='dataArchive6';  
  
% Load the label definitions and set classes and label IDs  
labelDefs=load(fullfile(projPath, whichSubproj, whichArchive, 'labelDefinitions.mat'));  
classes=labelDefs.S.Name;  
labelIDs=labelDefs.S.PixelLabelID;  
  
% Generate pixel label datastore  
pixelLabelPath=fullfile(projPath, whichSubproj, whichArchive, 'pixelLabelData');  
pxds = pixelLabelDatastore(pixelLabelPath, classes, labelIDs);  
  
% Generate image datastore  
imagePath=fullfile(projPath, whichSubproj, whichArchive, 'imgData');  
imds=imageDatastore(imagePath);  
  
%{  
% Optional: Run this code to testshow overlaid images  
for i = 1:44  
    I = readimage(imds,i);  
    C = readimage(pxds,i);  
    B = labeloverlay(I,C);  
    figure;imshow(B);  
end
```

```

%}

% Calculate frequency of classes
tbl = countEachLabel(pxds);
frequency = tbl.PixelCount/sum(tbl.PixelCount);

% Show the distribution of classes
dist=figure;figure(dist);
bar(1:numel(classes),frequency);
xticks(1:numel(classes));
xticklabels(tbl.Name);
xtickangle(45);
ylabel('Frequency');

% Create subsets of image and pixel label data for train, validate and test
shares=[0.6 0.2 0.2];
[imdsTrain, imdsVal, imdsTest, pxdsTrain, pxdsVal, pxdsTest] =
partitionTrainingData(imds,pxds,labelIDs, shares);

% Specify the network image size. This is typically the same as the traing image sizes.
imageSize = [240 360 3];

% Specify the number of classes.
numClasses = numel(classes);

% Create the DeepLab v3+.
network = deeplabv3plusLayers(imageSize, numClasses, 'resnet18');
% Below are optional standard nets to use
% lgraph = segnetLayers(imageSize,numClasses,3)
% lgraph = unetLayers(imageSize,numClasses,"EncoderDepth",4);

% Balance classes using class weighting
pxLayer = pixelClassificationLayer('Name','labels','Classes',tbl.Name);
network = replaceLayer(network,'classification',pxLayer);

% Prepare training, validation and test data stores
% First training data:
%   Data augmentation to provide more examples to the net
augmenter = imageDataAugmenter('RandXReflection',false,...
'RandXTranslation',[-5 5],'RandYTranslation',[-5 5]);
%   Datastore for training data
pximds = pixelLabelImageDatastore(imdsTrain,pxdsTrain, ...
'DataAugmentation',augmenter,'OutputSizeMode','resize','OutputSize', imageSize);
%   Datastore for validation data
pximdsVal = pixelLabelImageDatastore(imdsVal,pxdsVal, ...
'DataAugmentation',augmenter,'OutputSizeMode','resize','OutputSize',imageSize);

% Define training options:
options = trainingOptions('sgdm', ...% optimizer
'LearnRateSchedule','piecewise',...% learning rate is reduced every epoch. if you make
the rate constant over the traning process, please specify as "none".
'LearnRateDropPeriod',10,...% reduce the learning rate at the factor of 0.3 every 10
epoch
'LearnRateDropFactor',0.3,...% reduce the learning rate at the factor of 0.3
'InitialLearnRate',1e-2, ...% specify initial learning rate
'L2Regularization',0.0001, ...% L2 regularization 0.0001
'ValidationData',pximdsVal,...% validation data
'MaxEpochs',100, ...% max epoch 50
'MiniBatchSize',16, ...% mini-batch size
'Shuffle','every-epoch', ...% shuffle the data at each epoch
'VerboseFrequency',500,...% display the training status at every 500 iterations when
using trainNetwork
'Plots','training-progress',...% display the training curve
'ValidationPatience', Inf); % when the validation accuracy is not decreased for 10
times in a row, training stops

```

```

% Train the network - to stop execution here, set doTrain = false
doTrain = true;
if doTrain
    [net, info] = trainNetwork(pximds, network, options);
end

%{
% Optional: As a quick sanity check, run the trained network on one test image.
for i = 3:6 %size(imdsTest.Files)
    I = imresize(readimage(imdsTest,i), [240 360]);
    C = semanticseg(I, net);
    B = labeloverlay(I,C);
    figure(i)
    imshowpair(I,B, 'montage')
end

% Optional: Compare the results in C with the expected ground truth stored in pxdsTest.
expectedResult = read(pxdsTest); pxdsTest.reset;
actual = uint8(C);
expected = uint8(imresize(expectedResult{1}, imageSize));
imshowpair(actual, expected, 'montage'); title('left:inference result, right: hand-made
label');
%}

% Evaluate the trained network
pxdsResults = semanticseg(augmentedImageDatastore(imageSize, imdsTest), net, ...
    'MiniBatchSize', 10, ...
    'WriteLocation', [pwd, '\SegmentationResult'], ...
    'Verbose', false);

% Use evaluateSemanticSegmentation to measure semantic segmentation
% metrics on the test set results.
pxdsTest.ReadFcn=@(x) imresize(imread(x), [240 360]);
metrics = evaluateSemanticSegmentation(pxdsResults, pxdsTest, 'Verbose', false);
% To see the dataset level metrics, inspect metrics.DataSetMetrics .
metrics.DataSetMetrics
metrics.ClassMetrics

% Convert the confusion matrix table to a numeric matrix
confMatMatrix = table2array(metrics.NormalizedConfusionMatrix);
% Display the normalized confusion matrix as a heatmap
hm=figure; figure(hm);
h = heatmap(classes, classes, confMatMatrix);
title('Confusion Matrix (Percentages)');
xlabel('Predicted Labels');
ylabel('True Labels');
% Adjust the colormap to emphasize differences in percentages
colormap('jet');

```

Function partitionTrainingData

```

%-----
% Name: partitionTrainingData
% Date: 2024-05-09
% Author: Sven S
% Description:
% This function partitions the input image and pixel label datasets
% (parameters 'imds' and 'pxds') into training, validation and testing subsets.
% The partitioning ratios are defined by the parameter 'shares' as:
% [training%, validation%, testing%].
% The function is a modified version of the function partitionPascalVOCDData
% from Itakura, K. (2019). Semantic Segmentation Using Pascal-VOC dataset.
% https://github.com/KentaItakura/Semantic-segmentation-using-Pascal-VOC-with-MATLAB

```

```

%-----
function [imdsTrain, imdsVal, imdsTest, pxdsTrain, pxdsVal, pxdsTest] =
partitionTrainingData(imds,pxds,labelIDs, shares)
% Get number of files and shuffle indices
numFiles = numel(imds.Files);
shuffledIndices = randperm(numFiles);

% Get shares for training, validation and test
trainingShare = shares(1);
validationShare = shares(2);
testShare = shares(3);

% Use % defined by shares(1) of the images for training.
numTrain = round(trainingShare * numFiles);
trainingIdx = shuffledIndices(1:numTrain);
% Use defined by shares(2) of the images for validation
numVal = round(validationShare * numFiles);
valIdx = shuffledIndices(numTrain+1:numTrain+numVal);
% Use the rest for testing
testIdx = shuffledIndices(numTrain+numVal+1:end);

% Create image datastores for training, validation and test
trainingImages = imds.Files(trainingIdx);
valImages = imds.Files(valIdx);
testImages = imds.Files(testIdx);
imdsTrain = imageDatastore(trainingImages);
imdsVal = imageDatastore(valImages);
imdsTest = imageDatastore(testImages);

% Extract class and label IDs info
classes = pxds.ClassNames;

% Create pixel label datastores for training, validation and test
trainingLabels = pxds.Files(trainingIdx);
valLabels = pxds.Files(valIdx);
testLabels = pxds.Files(testIdx);
pxdsTrain = pixelLabelDatastore(trainingLabels, classes, labelIDs);
pxdsVal = pixelLabelDatastore(valLabels, classes, labelIDs);
pxdsTest = pixelLabelDatastore(testLabels, classes, labelIDs);
end

```

Appendix 2

Python script for cropping the raw radar images into several sub images to fit the DeepLab v3+ net input layer.

```
#-----
# Name: cropRawdata
# Revision: C
# Date: 2024-04-26
# Author: Sven S
# Description:
# This script crops radar images in .BMP format, with optional size, into several
# sub images of the size specified by variables sub_width and sub_height.
#
# The script assumes image data is residing in subdirectory defined by the variable
# rel_rawdata_directory and the generated output files will be written in subdirectory
# defined by rel_output_directory. See directory structure below.
#
# + current_directory----I---cropRawdata_revX.py
#                         I
#                         +---rel_rawdata_directory
#                         +---rel_output_directory
#-----
from PIL import Image
import os
from skimage import io, exposure

# Define the relative path to the directory containing the rawdata files
rel_rawdata_directory = 'rawdata'
# Define the relative path to the directory to write output files
rel_output_directory = 'output'

# Get the current directory
current_directory = os.getcwd()

# Define the full path to the rawdata and the output directories
rawdata_directory = os.path.join(current_directory, rel_rawdata_directory)
output_directory = os.path.join(current_directory, rel_output_directory)

# Define the size of each sub-image
sub_width = 360
sub_height = 150

# Iterate over each rawdata file
for filename in os.listdir(rawdata_directory):
    if filename.endswith(".BMP"):
        # Open the rawdata file
        image = Image.open(os.path.join(rawdata_directory, filename))
```

```

# Get the dimensions of the original image
width, height = image.size

# Calculate the number of sub-images horizontally and vertically
num_horizontal = width // sub_width
num_vertical = height // sub_height

# Offsets in rawdata
horiz_offset = 50
vert_offset = 40

# Iterate over each sub-image
for i in range(num_horizontal):
    for j in range(num_vertical):
        # Calculate the coordinates of the sub-image
        left = i * sub_width + horiz_offset
        upper = j * sub_height + vert_offset
        right = (i + 1) * sub_width + horiz_offset
        lower = (j + 1) * sub_height + vert_offset

        # Crop the sub-image
        sub_image = image.crop((left, upper, right, lower))

        # Perform contrast adjustment to enhance contrast
        sub_image = exposure.rescale_intensity(sub_image)

        # Construct the filename for saving the sub-image
        output_filename = f'outp_{filename[:-4]}_{i}_adj.BMP'

        # Save the sub-image with the constructed filename
        sub_image.save(os.path.join(output_directory, output_filename))

```