

UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Diego Haji Carvalho Campos

**Machine learning techniques for porosity estimation and salt layers
segmentation**

Belo Horizonte
2025

Diego Haji Carvalho Campos

**Machine learning techniques for porosity estimation and salt layers
segmentation**

Versão Final

Dissertação apresentada ao Programa de Pós-Graduação em
Ciência da Computação da Universidade Federal de Minas
Gerais, como requisito parcial à obtenção do título de Mestre
em Ciência da Computação.

Orientador: Heitor Soares Ramos Filho

Coorientador: Tobias Maia Rabelo Fonte Boa

Belo Horizonte
2025

Campos, Diego Haji Carvalho.

C198m Machine learning techniques for porosity estimation and salt layers segmentation [recurso eletrônico] / Diego Haji Carvalho Campos – 2025.

1 recurso online (107 f. il., color.) : pdf.

Orientador: Heitor Soares Ramos Filho
Coorientador: Tobias Maia Rabelo Fonte Boa.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.

Referências: f. 84-88.

1. Computação – Teses. 2. Aprendizado do computador – Teses. 3. Redes neurais convolucionais – Teses. 4. Sismologia – Teses. 5. – Solos - Porosidade – Teses. I. Ramos Filho, Heitor Soares. II. Fonte-Boa, Tobias Maia Rabelo. III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da computação. IV. Título.

CDU 519.6*82(043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS

MACHINE LEARNING TECHNIQUES FOR POROSITY ESTIMATION AND SALT LAYERS
SEGMENTATION

DIEGO HAJI CARVALHO CAMPOS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Heitor Soares Ramos Filho - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Tobias Maia Rabelo Fonte Boa - Coorientador
Faculdade de Ciências e Tecnologia - UFG

Prof. Marcos Oliveira Prates
Departamento de Estatística - UFMG

Prof. Antonio Alfredo Ferreira Loureiro
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 13 de outubro de 2025.



Magistério Superior, em 16/12/2025, às 18:31, conforme horário oficial de Brasília, com fundamento no art. 5º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Antonio Alfredo Ferreira Loureiro, Professor(a)**, em 17/12/2025, às 09:08, conforme horário oficial de Brasília, com fundamento no art. 5º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Marcos Oliveira Prates, Professor do Magistério Superior**, em 17/12/2025, às 12:11, conforme horário oficial de Brasília, com fundamento no art. 5º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Tobias Maia Rabelo Fonte Boa, Usuário Externo**, em 17/12/2025, às 15:16, conforme horário oficial de Brasília, com fundamento no art. 5º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



A autenticidade deste documento pode ser conferida no site

[https://sei.ufmg.br/sei/controlador_externo.php?](https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0)

[acao=documento_conferir&id_orgao_acesso_externo=0](https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0), informando o código verificador **4830758** e o código CRC **5D58776D**.

Acknowledgments

I thank my family and my professors for their support and encouragement throughout this journey. I also thank Petrobras for the opportunity to take part in the IAGEO project, from which originated the idea of this dissertation.

Resumo

Esta dissertação investiga a aplicação de técnicas de aprendizado de máquina em duas tarefas críticas na caracterização de reservatórios de petróleo: a estimação de porosidade e a segmentação de dados sísmicos, com foco específico em ambientes de camadas de sal e sistemas turbidíticos. A pesquisa aborda desafios centrais relacionados à escassez de dados rotulados, à subjetividade dos intérpretes e à complexidade inerente dos modelos geológicos do subsolo. Para a tarefa de segmentação, arquiteturas de redes neurais convolucionais, particularmente U-Net e Feature Pyramid Networks, foram aplicadas a dados sísmicos para delinear feições geológicas como canais turbidíticos e limites de camadas de sal. Os modelos demonstraram desempenho robusto tanto em conjuntos de dados sintéticos quanto reais, evidenciando capacidade de generalização em diferentes regiões geológicas, ainda que com níveis variados de sucesso dependendo da proximidade da área de treinamento e da qualidade dos dados. Técnicas como o Bayesian SegNet foram incorporadas para estimar a incerteza dos modelos, aumentando a interpretabilidade e a confiabilidade das previsões. Na tarefa de estimação de porosidade, diversos modelos de regressão foram desenvolvidos, incluindo perceptrons multicamadas, autoencoders e modelos convolucionais baseados em variantes da U-Net. Uma contribuição metodológica relevante é o uso de modelos de Tradução de Imagem-para-Imagem para prever mapas espacialmente contínuos de porosidade a partir de atributos sísmicos, compensando a falta de dados densos de perfilagem de poços. Apesar de melhorias com técnicas como aprendizado por transferência e modelos de comitê, os resultados revelaram limitações persistentes relacionadas à escassez de rótulos e à variabilidade regional. A pesquisa demonstra a viabilidade das abordagens de aprendizado de máquina na caracterização do subsolo, demonstrando que modelos baseados em CNN são eficazes para segmentação sísmica e estimativa de porosidade. No entanto, também destaca que avanços adicionais são necessários para superar as restrições impostas pela limitação de dados rotulados e pela heterogeneidade geológica. Trabalhos futuros são sugeridos na direção da incorporação de atributos sísmicos adicionais, da exploração de arquiteturas baseadas em transformadores e da expansão do uso de geração de dados sintéticos para melhorar a robustez dos modelos. Este estudo fornece uma contribuição para a integração do aprendizado de máquina em fluxos de trabalho geocientíficos, oferecendo insights práticos para a modelagem de reservatórios e para pesquisas futuras em interpretação do subsolo orientada por dados.

Palavras-chave: aprendizado de máquina; segmentação; correção de viés; tradução de

imagem para imagem; estimação de porosidade; U-Net; FPN.

Abstract

This dissertation investigates the application of machine learning techniques for two critical tasks in petroleum reservoir characterization: porosity estimation and seismic data segmentation, specifically focused on salt layer environments and turbiditic systems. The research addresses key challenges associated with the scarcity of labeled data, interpreter subjectivity, and the inherent complexity of subsurface geological models. For the segmentation task, convolutional neural network architectures, particularly U-Net and Feature Pyramid Networks, were applied to seismic data to delineate geological features such as turbiditic channels and salt layer boundaries. The models demonstrated robust performance in both synthetic and real datasets, showing the ability to generalize across different geological regions, though with varying levels of success depending on proximity to the training area and data quality. Techniques like Bayesian SegNet were incorporated to estimate model uncertainty, improving the interpretability and reliability of predictions. In the porosity estimation task, several regression models were developed, including multilayer perceptrons, autoencoders, and convolutional models based on U-Net variants. A key methodological contribution is the use of Image-to-Image Translation models to predict spatially continuous porosity maps from seismic attributes, compensating for the lack of dense well log data. Despite improvements with techniques like transfer learning and ensemble models, the results revealed persistent limitations related to label sparsity and regional variability. The research confirms the viability of machine learning approaches in subsurface characterization, demonstrating that CNN-based models are effective for seismic segmentation and porosity estimation. However, it also highlights that further advancements are necessary to overcome the constraints imposed by limited labeled data and geological heterogeneity. Future work is suggested in the direction of incorporating additional seismic attributes, exploring transformer-based architectures, and expanding the use of synthetic data generation to improve model robustness. This study provides a contribution to the integration of machine learning into geoscientific workflows, offering practical insights for reservoir modeling and for further research in data-driven subsurface interpretation.

Keywords: machine learning; segmentation; bias correction; image-to-image translation; porosity estimation; U-Net; FPN.

List of Figures

1.1	Flowchart representing the overall steps of the solutions.	20
2.1	U-Net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.	24
2.2	Four main stages of the workflow for developing the SFM: data collection, training data preparation, model pre-training, and downstream task application. First, a seismic dataset is collected worldwide from several 3-D surveys, from which 2-D seismic data are carefully chosen for pre-training. The model is further constructed and trained through a self-supervised pre-training strategy on unlabelled seismic data. The versatility of the pre-trained model is used in applications such as classification, segmentation, inversion, signal processing and interpolation.	25
2.3	A schematic of the Bayesian SegNet architecture. The diagram shows the entire pipeline for the system which is trained end-to-end in one step with stochastic gradient descent. The encoders are based on the 13 convolutional layers of the VGG-16 network, with the decoder placing them in reverse. The probabilistic output is obtained from Monte Carlo samples of the model with dropout at test time. We take the variance of these softmax samples as the model uncertainty for each class.	27
2.4	(a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a Convolutional Network as if it were a featurized image pyramid. (d) The proposed Feature Pyramid Network is fast like (b) and (c), but more accurate. In this figure, feature maps are indicate by blue outlines and thicker outlines denote semantically stronger features	29
2.5	Example network architectures for ImageNet. Left: the VGG-19 model as a reference. Middle: a plain network with 34 parameter layers. Right: a residual network with 34 parameter layers. The dotted shortcuts increase dimensions. .	31

2.6	CycleGAN-based generator architecture of the model. Cross-connected structure of the pre-trained encoder and decoder with feature mapping layer and residual connection of residual block. (a) Generator in the model for translating images from domain X to Y. (b) Generator in the model for translating images from domain Y to X.	33
3.1	Image generated with stacking data.	36
3.2	Example of labeling	37
3.3	Training results for the segmentation model.	39
3.4	Validation results for the segmentation model.	39
3.5	Testing results for the segmentation model.	40
3.6	Testing results for the segmentation model for the section 287 YZ.	40
3.7	Comparison of the segmentation of the 200 XZ section with different amounts of samples.	43
3.8	Seismic section and mask constructed from the synthetic porosity model. In the left Figure, white regions are positive indications for the area of interest (i.e., positive for the channel).	44
3.9	Comparison of results of seismic section 50 YZ (CL2052). A) Seismic section; B) Results of channel segmentation using labels with interpretative subjectivity; C) Results of channel segmentation using labels from the porosity model. Both approaches were able to identify multiple morphologies of the area of interest. Light portions indicate positive for the area of interest, and dark portions are negative.	45
3.10	Comparison of results of seismic section 305 YZ (CL2562). A) Seismic section; B) Results of channel segmentation using labels with interpretative subjectivity; C) Results of channel segmentation using labels from the porosity model. Unlike the previous approach, the method using labels from the porosity model was able to delineate a channel feature in the upper left corner of the seismic section. Light portions indicate positive for the area of interest, and dark portions are negative.	46
3.11	Results of segmenting areas of interest in other regions of the Marimbá Field. The seismic section is on the left, and the segmentation result is presented on the right. The method could identify with great efficiency other turbiditic systems of the Marimbá Field. Light portions indicate positive for the area of interest, and dark portions are negative.	47
3.12	Overlapping of the region and segmentation of the section XZ 100	48
3.13	Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made with the seismic data.	49

3.14	Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made with the seismic data.	51
3.15	Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made with the instantaneous phase data.	51
3.16	Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made with the similarity data.	52
3.17	Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made by combining the model's predictions.	53
3.18	Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made using the Bayesian model's predictions for the segmentation masks.	54
3.19	Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made using the Bayesian model's predictions for the confidence interval.	55
3.20	Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, one OK and one bad, respectively. The predictions were made using the parameters adjusted to the Bayesian model's predictions for the segmentation masks.	55
3.21	Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, one OK and one bad, respectively. The predictions were made using the parameters adjusted to the Bayesian model's predictions for the confidence intervals.	56
3.22	Comparison between two predictions in the Buzios region with the Crossline's 3650. The first image is the seismic data, the second is the U-Net prediction, and the third is the SFM prediction.	57
3.23	Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, one good and one OK, respectively. The predictions were made using the parameters adjusted to the Bayesian model with the FPN and seismic data only for the predictions of segmentation masks.	59
3.24	Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, one good and one OK, respectively. The predictions were made using the parameters adjusted to the Bayesian model with the FPN and seismic data only for the predictions of confidence intervals.	59

3.25	Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, one Ok and one bad, respectively. The predictions were made using the parameters adjusted to the Bayesian model with the FPN and instantaneous phase data only for the predictions.	61
3.26	Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, both bad. The predictions were made using the parameters adjusted to the FPN model and instantaneous phase data only for the predictions.	61
3.27	Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, both good. The predictions were made using the parameters adjusted to the FPN model and instantaneous phase data only for the predictions.	62
3.28	Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, both good. The predictions were made using the parameters adjusted to the FPN model and instantaneous phase data only for the predictions.	63
4.1	Predictions of the MLP model with 47 hidden layers to the well data.	66
4.2	Predictions of the MLP model with 47 hidden layers for section 287 YZ. . . .	66
4.3	Porosity heatmap for section 287 YZ.	67
4.4	Geostatistical model heatmap for section 287 YZ.	67
4.5	Error heatmap for section 287 YZ.	68
4.6	Heatmap for section 287 YZ.	69
4.7	Error heatmap for section 287 YZ.	69
4.8	Geostatistical model Error heatmap for section 287 YZ.	70
4.9	AutoEncoder heatmap for section 287 YZ with the WeightedLoss.	71
4.10	Comparison of the error relative to the ground truth of the Feed Forward model with the geostatistical model in section 287 YZ using the inversion variables .	72
4.11	Comparison of the error relative to the ground truth of the Feed Forward model with the geostatistical model in section 287 YZ using the adjusted inversion variables	73
4.12	Reference image for the POV seismic section 200XZ	75
4.13	Result image for the seismic section 200XZ of the model created by [37]. . . .	76
4.14	Ground truth for the POV seismic section 200XZ	76
4.15	Comparison between ground truth and the autoencoder prediction using the ResNet-18 encoder	77
4.16	Comparison between ground truth and the autoencoder prediction using the U-Net encoder	78
4.17	Comparison between ground truth and the autoencoder prediction using the U-Net improved decoder	79
4.18	Comparison between ground truth and the autoencoder prediction using the U-Net ensemble	79

B.1	Training metrics with 12.5% of training images.	102
B.2	Validation metrics with 12.5% of training images	103
B.3	Test metrics with 12.5% of training images	104
B.4	Training metrics for the XZ section	105
B.5	Validation metrics for the XZ section	106
B.6	Test metrics for the XZ section	107

List of Tables

- 2.1 Comparison between related works and the approaches developed. 34
- 3.1 Training metrics. 41
- 3.2 Validation metrics. 42
- 3.3 Test metrics. 42
- 4.1 Comparison of results for section 287 YZ for the geostatistical model and the
Feed Forward model. 73
- 4.2 Comparison of results for K-fold Mean MAE error across all folds. 80

Contents

1	Introduction	17
1.1	Problem definition	19
1.2	Outline	19
1.3	Our contribution	21
2	Related Work	23
2.1	Image Segmentation	23
2.1.1	U-Net: Convolutional Networks for Biomedical Image Segmentation	23
2.1.2	Seismic Foundation Model (SFM): a new generation deep learning model in geophysics	25
2.1.3	Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder- Decoder Architectures for Scene Understanding	26
2.1.4	Feature Pyramid Network	28
2.1.5	Epistemic uncertainty	29
2.2	Porosity Prediction	30
2.2.1	Deep Residual Learning for Image Recognition	30
2.2.2	Image-To-Image Translation Using a Cross-Domain Auto-Encoder and Decoder	32
2.3	Summary	33
3	Segmentation	35
3.1	Model Selection	35
3.1.1	Machine Specification	35
3.2	Synthetic data evaluation	36
3.2.1	Labeling	37
3.2.2	Training	37
3.2.3	Minimal amount of samples	40
3.3	Marimbá field dataset	42
3.3.0.1	Expansion of the test area	44
3.4	Buzios Region dataset	44
3.4.1	Semi-Supervised training approach	50
3.4.2	Bayesian model	52
3.4.3	SFM	56

3.4.4	Feature Pyramid Network	57
3.4.5	Errors and Fixes	60
3.5	Discussions	63
4	Porosity Estimation	65
4.1	Bias correction model	65
4.1.1	Multi-layer Perceptron	65
4.1.2	Autoencoder model	67
4.1.3	Weighted Loss	69
4.2	Adjusting the data	71
4.2.1	Linear regression	71
4.2.2	MLP	72
4.3	Image-to-Image Translation	73
4.3.1	ResNet model	77
4.3.2	U-Net model	77
4.3.3	U-Net improved model	78
4.3.4	U-Net ensemble model	78
4.4	Discussions	80
5	Conclusion	82
	References	84
	Appendix A Apêndice	89
A.1	Convolutional Autoencoder	89
A.2	Weighted Loss	90
A.3	MLP Regression Model	91
A.4	MLP parameters	91
A.5	Image-to-Image Translation Resnet	92
A.6	Image-to-Image Translation Unet	94
A.7	Image-to-Image Translation Unet improved decoder	97
	Appendix B Apêndice	102
B.1	Minimal amount of samples	102
B.2	ANP Dataset	103

Chapter 1

Introduction

Incorporating Machine Learning (ML) in petroleum exploration and reservoir management has brought changes to the oil and gas industry, reshaping approaches to complex challenges. ML techniques, particularly in seismostratigraphy and porosity estimation, have improved the efficiency and accuracy of identifying and exploiting hydrocarbon resources [22] [40] [21]. This shift has been facilitated by the increased accessibility of modern ML methods and digital data-collection efforts, which have led to a resurgence of ML research for surface and subsurface characterization [22]. This context motivates the central theme of this dissertation: exploring how modern ML models can overcome existing limitations in subsurface characterization, particularly related to data scarcity. These challenges are particularly significant in geologically complex regions like pre-salt basins, where structural heterogeneity is high and traditional modeling techniques often fall short due to the limited availability of data capable of adequately representing the complexity of the study area. This dissertation delves into applying ML techniques, specifically in seismic data analysis for seismostratigraphy and porosity estimation, to improve the efficiency and accuracy of identifying and exploiting hydrocarbon resources.

Semantic segmentation is a fundamental task in computer vision, with the aim of assigning semantic labels to each pixel in an image [11]. It has been widely explored in various domains, such as understanding urban scene [6], crystallographic defect identification [30], and remote sensing image analysis [39]. Deep learning methods, particularly convolutional neural networks (CNNs), have shown promising results in semantic segmentation tasks [25] and have been applied in diverse areas, including medical image analysis [17], 3D building modeling [43], and understanding of indoor scene [4]. The development of deep learning architectures for semantic segmentation has been a focus of research, with studies proposing novel methods such as fully convolutional networks [20], deep diffusion networks [34], and panoptic segmentation models [5]. These architectures typically consist of an encoder-decoder structure, where the encoder extracts high-level features, and the decoder projects them back to the input resolution for pixel-wise classification [29].

Porosity estimation in petroleum reservoirs is a crucial task for the oil and gas industry, and machine learning (ML) has emerged as a powerful tool for this purpose.

Recent ML applications have contributed to improving the understanding of hydrocarbon reservoirs. For instance, artificial neural networks have been utilized to understand porosity-permeability relationships on different geological scales, showing usefulness in understanding hydrocarbon reservoirs [1]. Additionally, support vector machines have been employed to predict hydrocarbon reservoir permeability, showing the importance of accurate permeability values in addressing petroleum engineering problems [9]. Furthermore, deep learning models, such as deep neural networks, have been explored for predicting movable fluid percentages in unconventional oil reservoirs, offering guidance for reservoir parameter measurement and development [36]. These advancements in ML techniques have demonstrated their efficacy in addressing key challenges within the oil and gas industry, ranging from porosity estimation to permeability estimation and movable fluid percentage modeling. The development of new permeability formulations using artificial intelligence approaches has been evidenced, further showing the role of ML in reservoir characterization [27]. Additionally, the application of ensemble models of neural networks has been instrumental in predicting petroleum reservoir properties from down-hole sensor data, aligning with the smart field phenomenon in the petroleum industry, where sensor data acquisition is widespread [3]. Furthermore, the utilization of XGBoost and artificial neural networks for porosity estimation has been compared, further showing the role of ML in reservoir characterization [2]. The main goal of this work is to develop and evaluate ML-based strategies that can be effectively applied to seismic data interpretation and porosity estimation in challenging geological settings. This includes combining segmentation techniques and also using image translation models to create frameworks capable of generalizing across different data conditions.

This dissertation explores some ML techniques, tailored to address specific petroleum exploration and reservoir management needs of our context. The work is conducted in collaboration with Petrobras and aims to automate various internal processes through ML techniques. Among the key contributions of this research are semantic segmentation with semi-supervised training and automated label selection, which tends to remove the interpreter bias, and also the use of Image-to-Image Translation (IIT) to generate continuous porosity maps from seismic inputs with synthetic data, allowing to speed up the porosity estimation process and removing computational bias from rounding calculations. By addressing challenges related to interpreter subjectivity, data scarcity, and the complexity of geological systems, this work contributes practical solutions directly applicable. Moreover, it provides a foundation for future research, encouraging further exploration of ML techniques in geoscience and supporting the development of more adaptable models for stratigraphical analysis.

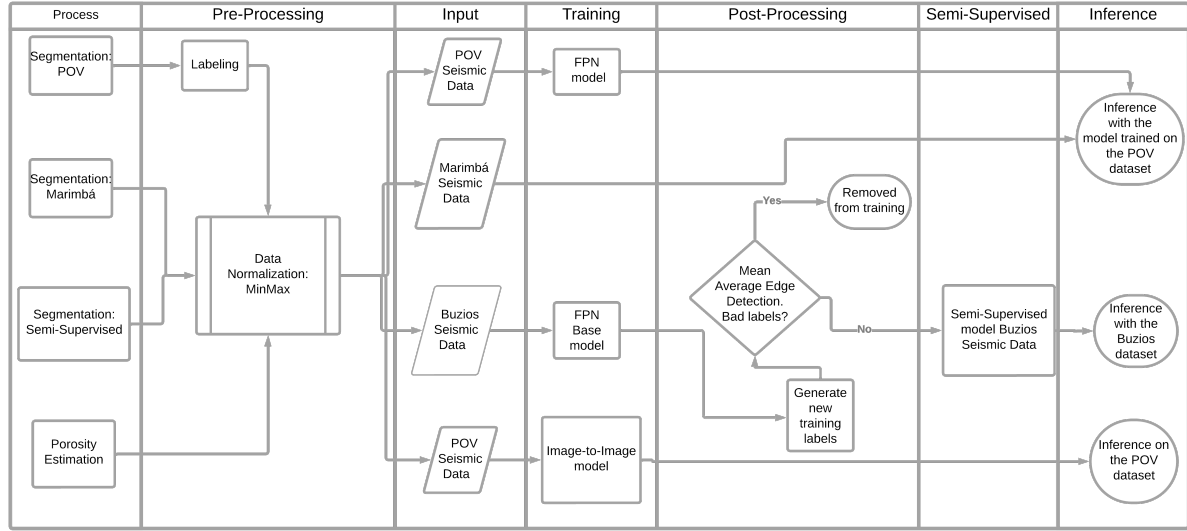
1.1 Problem definition

We divided our problem into two parts: segmentation for seismostratigraphy and ML techniques for porosity estimation, such as IIT. On the segmentation side, we need to separate our geological regions into four seismic facies. Our main challenge is the need for labels to train our model. One second challenge is the quality of the acquired data; the signal-noise ratio decreases steeply with depth. This low quality makes segmenting the lower regions, like pre-salt and basement, hard. We aim to find or create techniques to improve our predictions and circumvent these problems, especially the low quality. As it is a complex problem, we decided to tackle the segmentation in parts where we would train our model in different data, first synthetic data and then real data, to determine the best parameters and whether our model could be trained on synthetic data and generalized on real data. We can also have another problem if we cannot directly apply what we will do in the synthetic data to the real data, as we would need new labels that we do not possess, which will also be challenging to acquire. On the porosity estimation side, our main challenge is the necessity of labels. This lack of labels makes training machine learning models difficult as they need large amounts of data. We could use predictions from other regression models, but those predictions come with errors and biases, which could negatively influence our model. Another problem is the necessity to evaluate the region of interest neighborhood, as the neighbors can also present information used in the estimation. That necessity invalidates 1D models, as they cannot observe the neighbors. Unfortunately, we cannot use a 3D model as we do not have enough labeled data. The porosity search is the most difficult challenge, as the usually available labels are in the wells, and the wells are 1D, creating several issues manipulating the data for usability. One more issue is the low number of wells, which also limits the number of ground truth samples available, making it more problematic to evaluate the 2D results as, once again, we have a 1D column in a 2D "image" for validation and tests. Figure 1.1 shows our general idea for tackling the problem.

1.2 Outline

Chapter 2 highlights the use of ML in porosity estimation, emphasizing its applicability across different geological settings and scales. Studies have applied various ML techniques, such as artificial neural networks, supervised learning, and deep learning,

Figure 1.1: Flowchart representing the overall steps of the solutions.



Source: The author

for tasks ranging from predicting hydrocarbon reservoir permeability to water resources assessment. These studies demonstrate how ML has become a tool not only in hydrocarbon exploration but also in other environmental and geological applications. In addition, Chapter 2 discusses image segmentation approaches and their recent developments in machine learning, with a focus on models such as U-Net, ResNet, Feature Pyramid Networks (FPN), and Bayesian SegNet. In the chapter 3, the research employs a computational setup including an Intel core i5 13600K processor and an Nvidia RTX 2070 graphics card, to handle the demanding ML tasks. Semantic segmentation, a technique used in this research, is applied to identify turbiditic channels in seismic data, aiding hydrocarbon exploration. The U-Net model, a convolutional neural network architecture, is widely used due to its effectiveness in biomedical image segmentation tasks and adaptability to various domains.

Chapter 3 also details the methodology used, including the process of data labeling, training with synthetic and real datasets. It also discusses the practicality of the ML models trained on synthetic data when applied to actual data. While effective, this training is subject to interpreter subjectivity, which the research aims to mitigate through various experiments, including transfer learning (TL) and training on different datasets. TL, in particular, is employed to adapt models trained on one dataset (synthetic area) to new tasks (pre-salt area segmentation). The results highlight the limitations and possibilities of ML models when transitioning from synthetic to real-world geological data.

Chapter 4 details the development of regression models for porosity estimation. A critical challenge addressed in this research involves the limited availability of labeled data. While seismic datasets provide extensive spatial coverage, well-log data often used

as ground truth are limited and one-dimensional. This disparity complicates the training and validation processes for ML models designed to operate on two-dimensional seismic images. To overcome this, the research adopts strategies such as synthetic data generation, bias correction, and transfer learning, demonstrating how these techniques can compensate for data scarcity and improve prediction accuracy.

Finally, Chapter 5 presents the conclusions drawn from this research. The results confirm that CNNs, particularly U-Net-based and FPN architectures, are effective tools for seismic data segmentation and porosity estimation. However, the research also identifies key limitations, particularly regarding the scarcity of labeled well log data, which impacts model generalization and accuracy. Although advanced techniques such as transfer learning, ensemble models, and IIT yielded improvements, the results indicate that further work is required to address challenges related to sparse labeling, model bias, and regional variability. The conclusion also suggests directions for future research, including the incorporation of additional seismic attributes, exploration of transformer-based architectures, and development of larger, more diverse datasets to enhance model robustness and scalability.

1.3 Our contribution

We present an integrated pipeline for seismic segmentation and porosity estimation tailored to data-scarce and geologically complex settings.

For seismic segmentation, we adopt U-Net and FPN to delineate geobodies in seismic sections, and we reduce interpreter subjectivity by deriving training masks from a synthetic porosity model of the synthetic dataset. We show that stable training is achievable with about 12.5 percent of labeled slices, indicating good label efficiency, at least within the synthetic dataset. We then assess generalization on real data from the Marimbá Field and on the Búzios region. Transfer learning was evaluated only in the synthetic-to-Marimbá setting and was not reused for Búzios due to the difference in classes and their quantity. For Búzios we utilize a transfer learning from a previous experiment in the same region. Bayesian uncertainty via Monte Carlo dropout was explored to diagnose low-reliability areas but was not incorporated into the final segmentation outputs. We also provide qualitative comparisons against a Seismic Foundation Model to position our solutions with regard to recent baselines. We also develop a semi-supervised method that utilizes an edge detection technique to remove bad labels from the training pipeline as they could impact negatively the whole process. This was necessary due to the low amount of labels provided for the real data training.

For porosity estimation, we proceed in three stages. First, we attempt direct prediction with MLPs. Second, we target the petroelastic-inversion bias by learning residual corrections for selected variables using linear regression and MLPs, followed by a feed-forward predictor. When these strategies prove insufficient, we recast the task as IIT from seismic inputs to spatially continuous porosity maps, implementing autoencoder variants with U-Net and ResNet backbones, improved decoders, weighted loss to emphasize errors inside turbiditic channels, and simple ensembles to balance under and overestimation.

This work differentiates from the previously presented by using extra techniques, such as edge detection, for better segmentation. We also combine this with a semi-supervised strategy to circumvent our label limitation. This combination has not been presented before in previous papers, especially in the context of seismic segmentation, and this was a fundamental part of the solution, as we would only have been able to solve our problem with it. On the porosity search side, the main difference is our approach to the problem, where we used the IIT model to create porosity maps from the seismic input, something that has yet to be tried.

Chapter 2

Related Work

This chapter presents previous works that are correlated with this one, directly or indirectly. Those works are divided in two major sections: image segmentation for seismic stratigraphy, and image translation for porosity prediction. The purpose of this chapter is to show the relationship between what has been made before and what we are doing it right now.

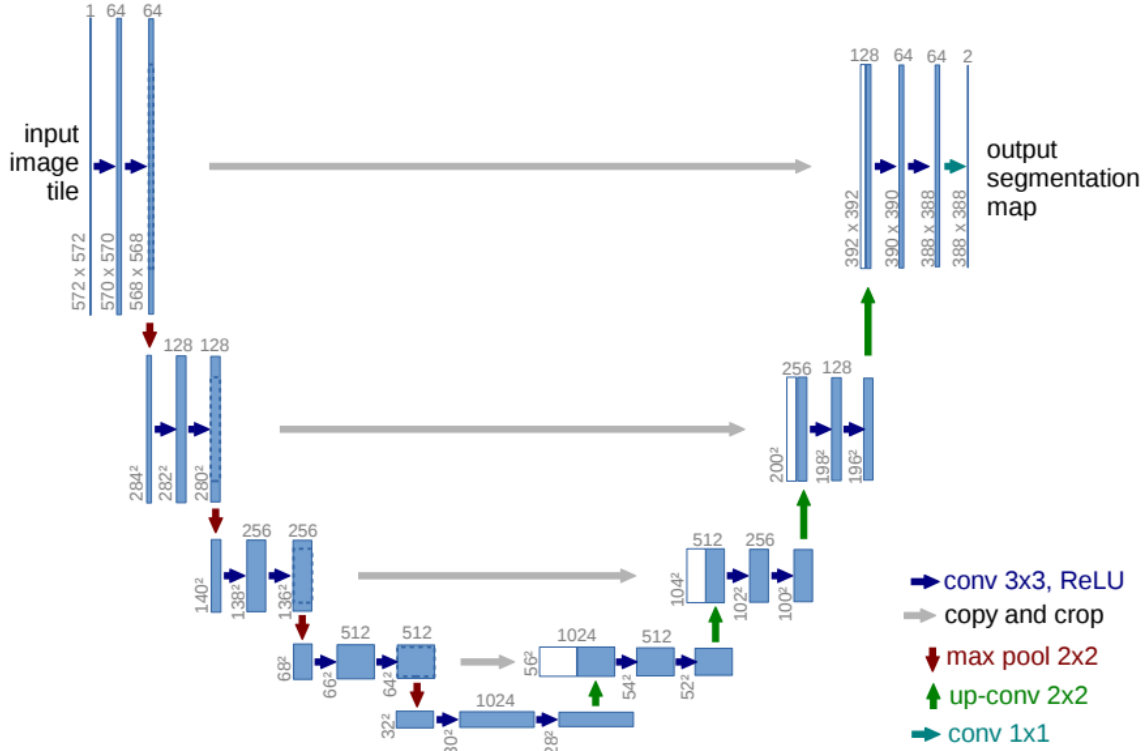
2.1 Image Segmentation

2.1.1 U-Net: Convolutional Networks for Biomedical Image Segmentation

The U-Net model [31] is a convolutional neural network architecture primarily designed for biomedical image segmentation tasks. It comprises a contracting path to capture context and a symmetric expanding path for precise localization. The contracting path follows the typical architecture of a convolutional network, with repeated application of convolutions and max-pooling operations to reduce the spatial dimensions. The expanding path, on the other hand, consists of up-convolutions for upsampling and concatenation with the correspondingly cropped feature map from the contracting path to provide localization information. This architecture allows the network to learn from very few annotated samples and produce precise segmentation maps [42] [35] [41]. This architecture is available on Figure 2.1.

The U-Net model has been widely used and extended in various domains, including medical image segmentation, weather forecasting, and geoscientific imagery. For example, the U-Net++ model redesigns skip connections to exploit multiscale features in image segmentation, consistently outperforming baseline models for semantic segmentation across

Figure 2.1: U-Net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

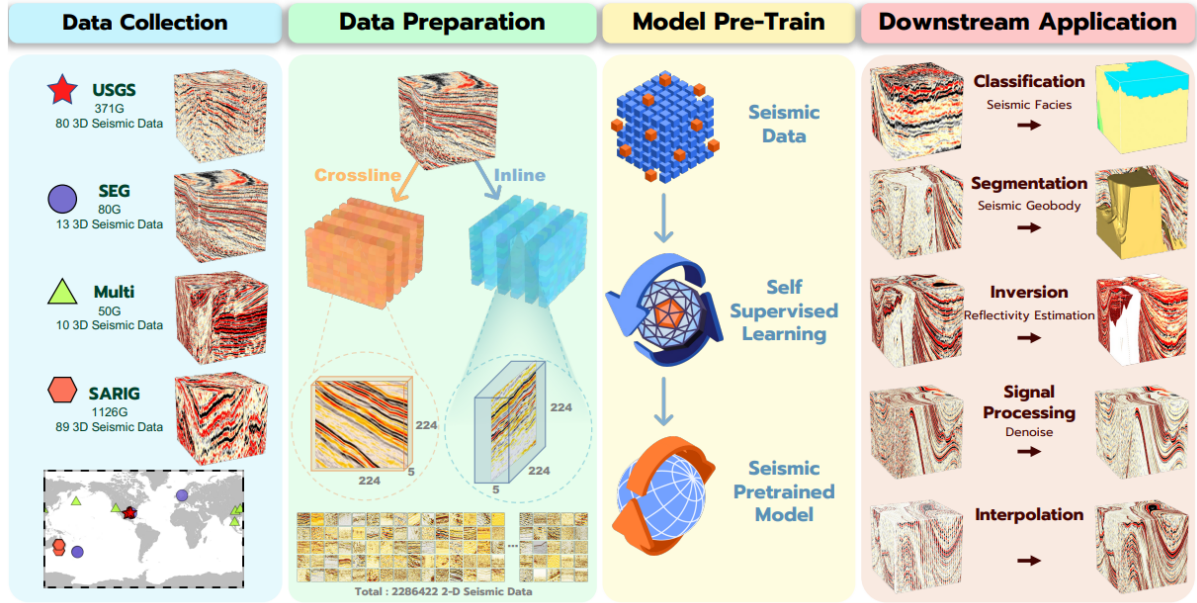


Source: [31]

different datasets and backbone architectures [41]. The H-DenseU-Net model, a densely connected hybrid U-Net, has been proposed for liver and tumor segmentation from CT volumes, demonstrating its effectiveness in segmenting liver tumors [16]. Furthermore, the TMD-U-Net model, a triple-U-Net with multiscale input features and dense skip connection, has been introduced for medical image segmentation, incorporating modifications in network node interconnection and the use of dilated convolutions [35].

In addition, the U-Net architecture has been adapted for various applications, such as detection of road cracks, urban land use, classification of cover and automatic segmentation for cervical cancer brachytherapy, showcasing its versatility and applicability in different domains [38] [7].

Figure 2.2: Four main stages of the workflow for developing the SFM: data collection, training data preparation, model pre-training, and downstream task application. First, a seismic dataset is collected worldwide from several 3-D surveys, from which 2-D seismic data are carefully chosen for pre-training. The model is further constructed and trained through a self-supervised pre-training strategy on unlabelled seismic data. The versatility of the pre-trained model is used in applications such as classification, segmentation, inversion, signal processing and interpolation.



Source: [32]

2.1.2 Seismic Foundation Model (SFM): a new generation deep learning model in geophysics

The paper on the Seismic Foundation Model (SFM) [32] presents the development of a deep learning-based model for geophysical tasks, focused explicitly on seismic facies classification and other seismic data applications. They based the training approach for SFM on a transformer architecture using self-supervised learning, with a dataset of over 2.2 million 2-D seismic images derived from 192 globally collected 3-D seismic volumes. The overview of this process is available in Figure 2.2

The authors collected a large and diverse dataset of seismic images from various global sources. The data was prepared by slicing 3-D seismic volumes into 2-D images along inline and crossline directions, resulting in a highly diverse training dataset. The model is pre-trained using a generative self-supervised learning approach, which does not rely on labeled data. Specifically, they used the Masked Autoencoders (MAE) method, where parts of the seismic data were masked and reconstructed, helping the model learn

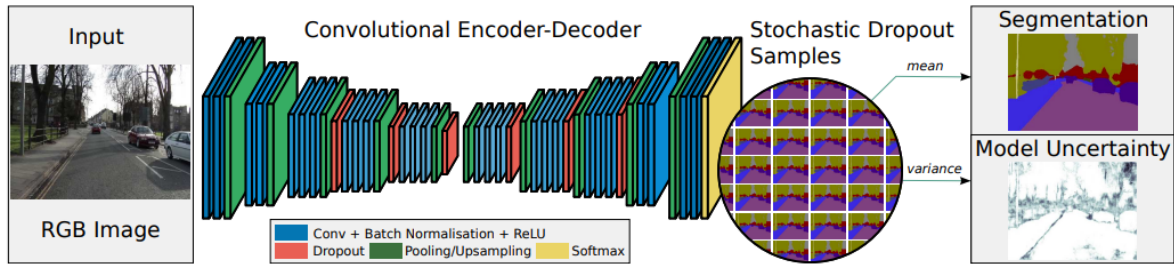
robust feature representations. They built the SFM on a Transformer architecture, which enables dynamic attention mechanisms and captures global seismic data features. The architecture was chosen over CNNs for its better generalization ability for global context extraction, critical for seismic data interpretation. Seismic facies classification is essential for interpreting seismic data to understand subsurface structures and geological features. In this task, seismic data is divided into facies based on the characteristics of seismic reflections. The seismic facies classification task was performed using the publicly available Parihaka seismic survey dataset. The dataset was partitioned into training and validation subsets with crossline seismic images selected to ensure diversity. They pre-trained the SFM against baseline models like U-Net, Deeplab, and a Transformer trained from scratch. SFM was tested in two modes: "SFM Frozen" (with pre-trained parameters) and "SFM Fine-tune" (with fine-tuning for the specific task). The SFM models (Frozen and fine-tuned) significantly outperformed baseline models like U-Net and Deeplab in terms of mean Intersection over Union (mIoU) and Class Pixel Accuracy (CPA). The SFM Fine-tune model, in particular, showed superior performance across all seismic facies classes, demonstrating better continuity and accuracy in classifying seismic reflections. The results from seismic facies classification indicated that the SFM, pre-trained on a large and diverse dataset, provides significant improvements in generalization, performance, and scalability. The model's ability to maintain accuracy even when applied to validation data far from the training samples shows its strong generalization capabilities.

2.1.3 Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding

SegNet is a deep convolutional encoder-decoder architecture for pixel-wise semantic segmentation [13]. It consists of a series of encoder and decoder layers, where the encoder extracts features through convolution, batch normalization, and max-pooling. In contrast, the decoder uses max-pooling indices for up-sampling to recover the input resolution. Using max-pooling indices enables better retention of boundary details in segmented images and reduces the model's parameters, allowing for end-to-end training using stochastic gradient descent. They based SegNet's encoder on the 13 convolutional layers of the VGG-16 network. The Figure 2.3 shows the general idea of the SegNet architecture.

Bayesian SegNet extends the SegNet model to provide pixel-wise uncertainty

Figure 2.3: A schematic of the Bayesian SegNet architecture. The diagram shows the entire pipeline for the system which is trained end-to-end in one step with stochastic gradient descent. The encoders are based on the 13 convolutional layers of the VGG-16 network, with the decoder placing them in reverse. The probabilistic output is obtained from Monte Carlo samples of the model with dropout at test time. We take the variance of these softmax samples as the model uncertainty for each class.



Source: [13]

estimates, which are crucial for decision-making in safety-critical applications like autonomous driving. Bayesian SegNet achieves this by using Monte Carlo dropout during test time, approximating the posterior distribution over the network's weights. Dropout layers are applied after specific convolutional layers, and the model is sampled multiple times to estimate the mean and variance of class probabilities. The variance provides an estimate of model uncertainty. Bayesian SegNet is particularly effective for smaller datasets, as it can model uncertainty from data scarcity, providing more reliable predictions in challenging conditions. **CamVid Dataset:** The performance of Bayesian SegNet was tested on the CamVid road scene understanding dataset. Bayesian SegNet outperforms conventional SegNet by achieving a higher mean Intersection over Union (IoU) and class accuracy across all evaluated classes. The model showed significant improvements for thin and small object classes, such as poles and pedestrians, which are typically more difficult to segment. Bayesian SegNet sets a new benchmark for scene segmentation on the CamVid dataset. **SUN RGB-D Dataset:** On the SUN RGB-D indoor scene understanding dataset, Bayesian SegNet performs better than baseline models such as Fully Convolutional Networks (FCN) and conventional SegNet. The model achieves the highest class average accuracy and provides reliable uncertainty estimates for indoor scenes, where object occlusion and varied poses increase the segmentation difficulty. Bayesian SegNet also performs well on the Pascal VOC 2012 segmentation challenge, achieving competitive results with significantly fewer parameters than other deep learning architectures like FCN and Dilation Network. Bayesian SegNet provides a 2-3% improvement in segmentation performance across multiple architectures when applying Monte Carlo dropout. In conclusion, Bayesian SegNet improves segmentation accuracy and provides a reliable measure of model uncertainty, making it a valuable tool for real-time applications

in which confidence in predictions is critical.

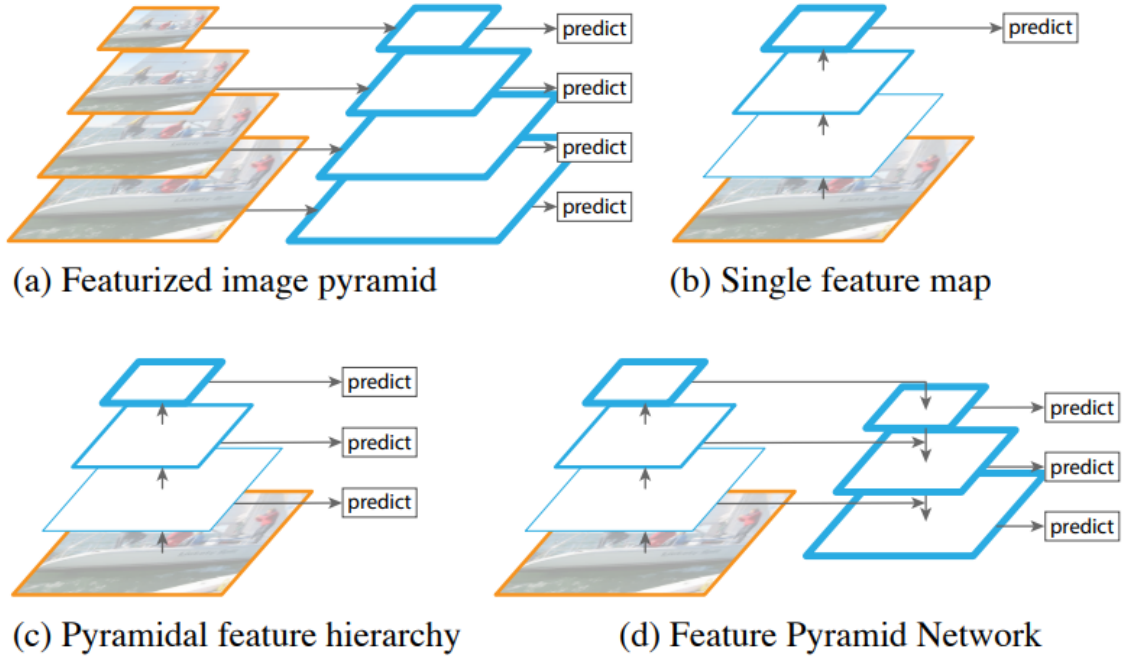
2.1.4 Feature Pyramid Network

The FPN, proposed by [18], is a method that leverages the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to build feature pyramids with strong semantics at all scales. Traditional object detection methods, which only rely on the top-most feature maps, face limitations in detecting objects across a wide range of scales. FPN introduces a top-down architecture that combines high-level semantic feature maps at different levels, enriched with spatially finer information, significantly enhancing multi-scale object detection performance. FPN creates a multi-scale feature representation, where feature maps from different layers of the backbone network (e.g., ResNet) are combined. The top-down pathway in FPN upsamples the high-level feature maps, while lateral connections merge these maps with the corresponding lower-level feature maps containing finer spatial details. This combination enables object detectors to perform well on objects of varying sizes without needing image pyramids. The resulting feature maps are scale-invariant, preserving semantic richness and high resolution, which is beneficial for detecting both small and large objects. Figure 2.4 shows a superficial look of the architecture, when compare to other pyramid networks.

The Microsoft COCO (Common Objects in Context) dataset, introduced by [19] is a large-scale benchmark designed to advance object recognition by focusing on scene understanding in realistic environments. It contains over 300,000 images with approximately 2.5 million labeled object instances across 91 categories, where objects are annotated using precise per-instance segmentation masks rather than only bounding boxes.

The MS COCO dataset plays a central role in the evaluation of the FPN framework, as it provides a highly challenging benchmark characterized by a large number of object categories, complex scenes, and significant variation in object scales—particularly a high prevalence of small objects. This makes COCO especially suitable for assessing the effectiveness of FPN, whose core contribution is improving multi-scale feature representation through a top-down architecture with lateral connections. The authors use COCO not only because it is a widely accepted standard for object detection benchmarking, but also because its difficulty highlights the limitations of prior methods that struggle with scale variance. By demonstrating consistent improvements in Average Precision (AP), especially for small objects, the paper leverages COCO to validate that FPN successfully addresses one of the most critical challenges in modern object detection: robust performance across different object sizes.

Figure 2.4: (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a Convolutional Network as if it were a featurized image pyramid. (d) The proposed Feature Pyramid Network is fast like (b) and (c), but more accurate. In this figure, feature maps are indicated by blue outlines and thicker outlines denote semantically stronger features



Source: [18]

2.1.5 Epistemic uncertainty

[8] shows that a neural network with dropout applied before every weight layer is mathematically equivalent to a probabilistic approximation of a deep Gaussian process. This interpretation treats dropout as an approximate Bayesian inference, allowing neural networks to capture uncertainty by integrating their weights. The derivation explains how dropout effectively minimizes the Kullback–Leibler divergence between a deep Gaussian process’s approximate and actual posterior. This new insight shows that dropout integrates over models probabilistically, avoiding overfitting while preserving model uncertainty. They also discuss how the dropout objective function relates to standard neural network optimization with L2 regularization, illustrating that dropout sampling can be interpreted as approximating a Gaussian process with a variational distribution. The key result here is that dropout networks can capture uncertainty with minimal computational overhead, which can be applied to any network architecture or non-linearity.

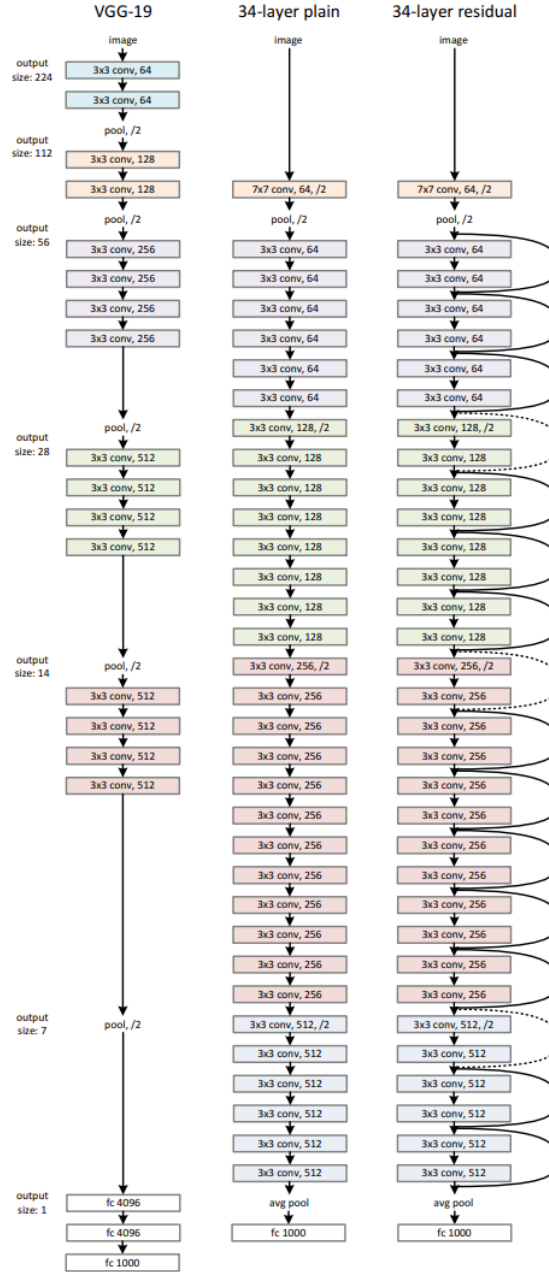
The authors provide mathematical details to show how this approximation is valid for many models, including those using alternative dropout forms like drop-connect and multiplicative Gaussian noise. They extend the Bayesian interpretation of dropout to derive methods for estimating model uncertainty. The authors describe a method called Monte Carlo dropout, where multiple forward passes are performed through the network during inference, and the results are averaged to obtain predictive uncertainty. This method allows deep neural networks to express uncertainty in their predictions by treating the dropout masks as random variables. They provided the theoretical grounding for this method, explaining how the uncertainty estimates can be derived from the dropout masks sampled during inference. The authors also explore the implications of this approach for both regression and classification tasks. They show that the model’s predictive mean and variance can be calculated using Monte Carlo sampling, providing an empirical way to capture uncertainty in neural network predictions. This technique is computationally efficient, requiring only a modest increase in inference time. They discussed how this method can be applied to existing models trained with dropout, allowing practitioners to estimate uncertainty without modifying their neural network architectures. They also presented a detailed experimental evaluation of the uncertainty estimates obtained using dropout neural networks. The experiments focus on regression and classification tasks using MNIST and atmospheric CO2 concentration data datasets. The authors demonstrate that the uncertainty estimates produced by Monte Carlo dropout lead to significant improvements in predictive accuracy, particularly in log-likelihood and root mean square error (RMSE) compared to state-of-the-art methods. They highlighted the ability of dropout-based models to capture uncertainty in both training and test data, providing more reliable predictions for out-of-distribution samples. In conclusion, the experiments validate the theoretical claims, showing that dropout can effectively represent uncertainty in various tasks. The results also indicate that dropout uncertainty can improve reinforcement learning performance by allowing agents to balance exploration and exploitation more effectively.

2.2 Porosity Prediction

2.2.1 Deep Residual Learning for Image Recognition

[10] focuses on applying deep residual learning in image recognition tasks. The

Figure 2.5: Example network architectures for ImageNet. Left: the VGG-19 model as a reference. Middle: a plain network with 34 parameter layers. Right: a residual network with 34 parameter layers. The dotted shortcuts increase dimensions.



Source: [10]

authors propose a novel architecture called ResNet, which utilizes residual connections to enable the training of deep neural networks. The ResNet architecture achieved state-of-the-art performance on the ImageNet dataset, demonstrating the effectiveness of deep residual learning in image recognition tasks. The general architecture of the ResNet is on Figure 2.5.

The ResNet architecture addresses the problem of vanishing gradients that often

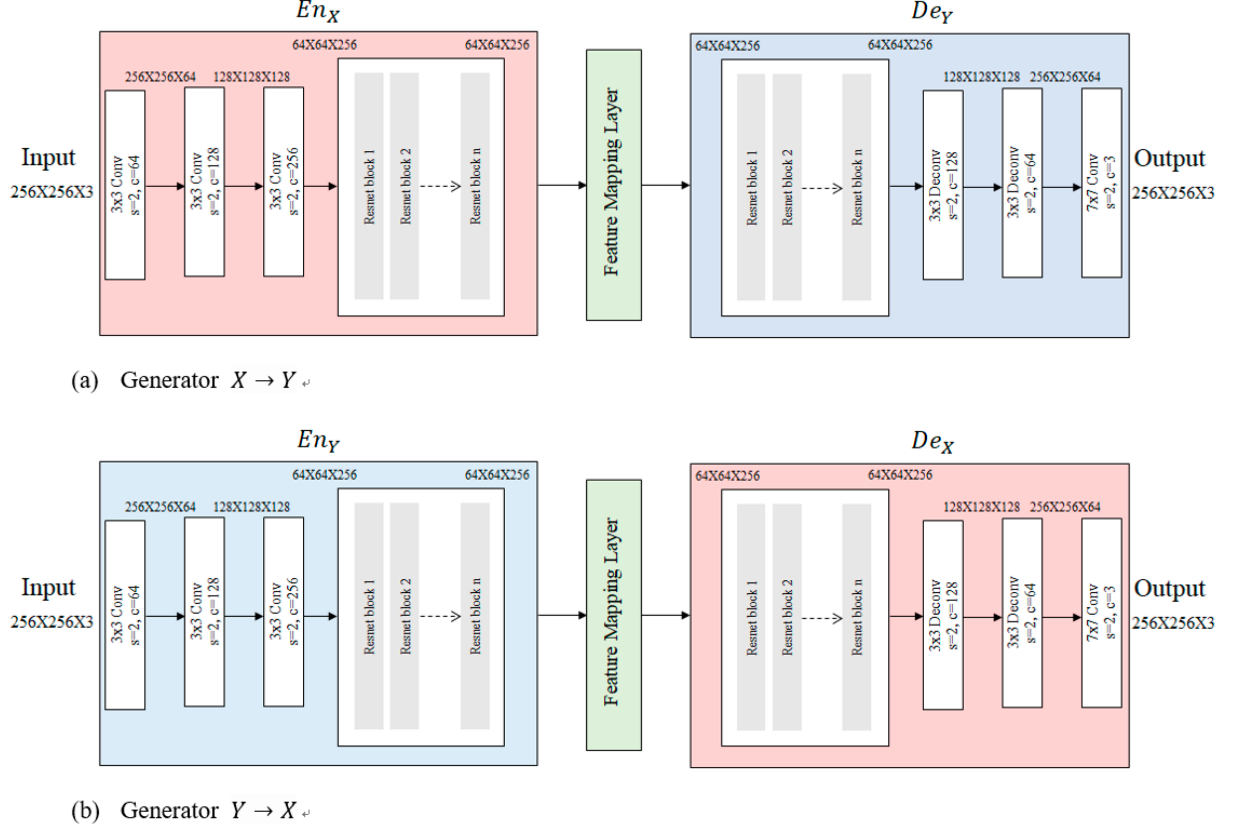
occur in deep neural networks by introducing skip connections, which allow the network to learn residual mappings instead of directly learning the desired underlying mapping. This approach enables the training of much deeper networks without suffering from degradation in performance. The authors conducted extensive experiments to evaluate the performance of ResNet on various benchmark datasets, including ImageNet, CIFAR-10, and CIFAR-100. The results showed that ResNet achieved significantly better accuracy than previous state-of-the-art models, demonstrating the effectiveness of deep residual learning in image recognition tasks.

2.2.2 Image-To-Image Translation Using a Cross-Domain Auto-Encoder and Decoder

[37] presents a distinctive methodology for IIT using a cross-domain auto-encoder and decoder, which enhances the ability to maintain content integrity while translating images between different domains. This paper introduces a dual structure composed of an auto-encoder for encoding images into a common feature space and a decoder that reconstructs the encoded features into images within a different domain. The key innovation lies in the cross-domain capability of the decoder, which can interpret and reconstruct features in multiple target domains from a single encoded representation. The overview of the architecture is on Figure 2.6

This approach significantly improves the flexibility and efficiency of image translation tasks. It is highly applicable in fields where multiple outputs from a single input are beneficial, such as style transfer, multi-lingual text imaging, or simultaneous enhancement of various image qualities. The model is trained with a combination of loss functions that preserve content, enhance style accuracy, and encourage feature consistency across domains, leading to high-quality translations that are both diverse and precise. Extensive testing shows that their model achieves high scores in conventional metrics and satisfies subjective quality assessments, indicating its practical effectiveness in real-world applications.

Figure 2.6: CycleGAN-based generator architecture of the model. Cross-connected structure of the pre-trained encoder and decoder with feature mapping layer and residual connection of residual block. (a) Generator in the model for translating images from domain X to Y. (b) Generator in the model for translating images from domain Y to X.



Source: [37]

2.3 Summary

The table 2.1 shows the main similarities and differences between other authors and our work. It sums up the parts where we took inspirations from and also where we had to diverge.

Work	Similarities	Main differences
[31] (U-Net)	U-Net used for seismic facies segmentation and as a decoder backbone in IIT models. Training with limited labeled data and data augmentation.	Applied to seismic data with weak labels and masks derived from synthetic porosity models; encoder implemented via SMP with ResNeXt50 backbone; stable performance observed with only 12.5% of labeled slices.
[18] (FPN)	Adoption of FPN as a multi-scale segmentation architecture for seismic sections.	Focused on seismic facies instead of natural object detection; tested with additional seismic attributes such as instantaneous phase and similarity, with region-specific adjustments.
[13] (Bayesian SegNet)	Use of Monte Carlo dropout for uncertainty estimation in segmentation tasks.	Uncertainty applied diagnostically to identify low-confidence regions, rather than being integrated into the final segmentation output.
[32] (SFM)	Served as a qualitative reference baseline for seismic segmentation with self-supervised pretraining.	Not directly adopted; final solution relies on CNNs (U-Net and FPN) instead of transformer-based masked autoencoders.
[8] (Epistemic Uncertainty)	Concept of estimating epistemic uncertainty through dropout at inference time.	Applied specifically to real and synthetic seismic data to assess model reliability and distance from training regions.
[10] (ResNet)	Use of ResNet-18 as encoder backbone in autoencoder architectures for IIT.	Adapted for regression of continuous porosity maps instead of classification of natural images.
[37] (Cross-domain IIT)	Inspired the design of IIT models preserving structural consistency across domains.	GANs were not used; the final architecture employed U-Net/ResNet autoencoders with improved decoders, weighted loss, and ensemble prediction to mitigate over- and underestimation.

Table 2.1: Comparison between related works and the approaches developed.

Chapter 3

Segmentation

3.1 Model Selection

As shown in chapter 2, given the versatility and high capability of the U-Net along with its low computational costs, we opted for that model. The main reason is that U-Net can generate good results with a small number of samples, which is also very important in this context, where we have none or very few samples labeled by experts along the whole area. No adaptations were made in the model at this point, we used the network as designed by the segmentations models pytorch (SMP) library ¹.

3.1.1 Machine Specification

All experiments were performed in a machine with the following specifications:

- Intel core i5 13600K
- 64Gb 5600 Mhz
- Nvidia RTX 2070
- SSD Samsung 860 EVO 256 GB
- SSD XPG GAMMIX S11 Pro 512 Gb
- HD Seagate ST4000DM004 4Tb

¹https://github.com/qubvel/segmentation_models.pytorch

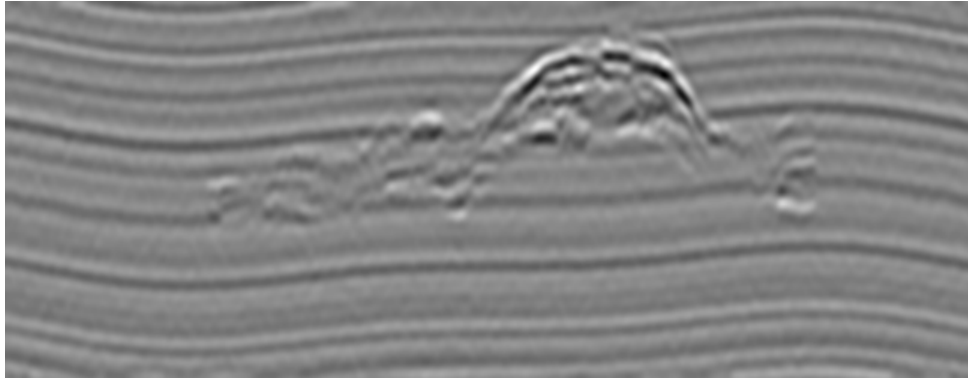
3.2 Synthetic data evaluation

In the problem we are working on, we identified semantic segmentation as a technique to provide a solution to automatically identify turbiditic channels to segment the channel region from the others. Therefore, efforts will be made to estimate the properties of interest for exploration of hydrocarbons within the region of interest (ROI). Selecting ROI will reduce computational effort and produce a more expressive model since the region is observable outside the channels and presents a low correlation between the secondary variables (seismic) and those of interest (e.g. porosity) due to the synthetic data having its values outside the ROI zeroed for control.

Combining seismic traces into a single trace, commonly known as stacking in seismic data processing, attenuates random noise and amplifies the coherent signal in the gather [15]. This process involves adding the seismic variables FAR, UFAR, and NEAR in the study context. Figure 3.1 is an example of the result. Stacking is essential in seismic data processing procedures, such as prestack migration, as it enables detecting foul traces/shots and restoring or replacing noisy seismic observations [15].

The data provided for modeling and learning was generated from the thesis [28]. The data is synthetic and created using the Marimbá field as a reference. By synthesizing the Marimbá field, it was possible to create a controlled environment to test our hypothesis and the ground truth values for them. Considering that without this controlled environment, we cannot know that our models for porosity prediction are performing correctly. However, we do not have infinite time or computational resources to keep evaluating every single point in the data provided, and as previously established, we do have an ROI. Therefore, we decided to segment the ROI.

Figure 3.1: Image generated with stacking data.

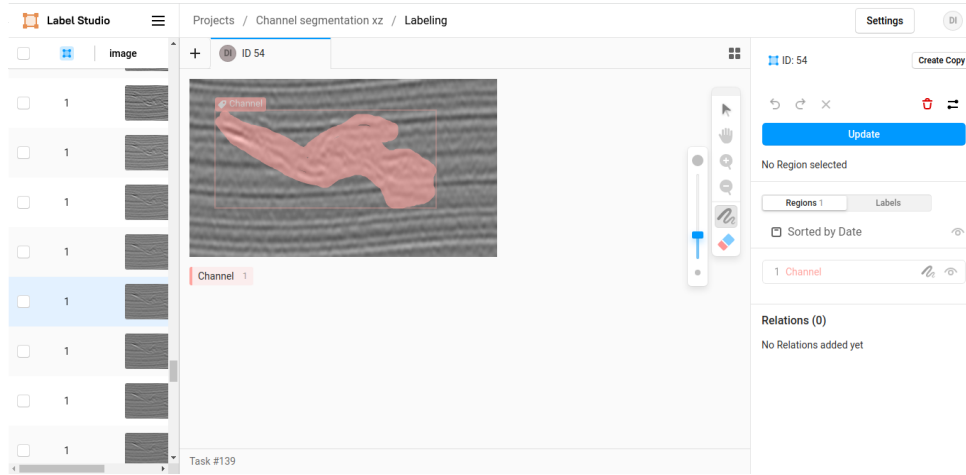


Source: The author

3.2.1 Labeling

Initially, the seismic data is loaded, and then all the images of each section are extracted from NumPy arrays by fixing, one at a time, X and Y axes forming the sections XZ and YZ. Then, we used the Label Studio² labeler to label the channels of one of the sections. An example of a weak label is in the Figure 3.2. That means that the label might have some mistakes or flaws. We used the labels and obtained favorable results. Later on, we also compare the segmentation masks generated from our weak labels with the ground truth labels to see the efficiency of those labels. We used a semantic image segmentation model with the labeled channels for channel identification training.

Figure 3.2: Example of labeling



Source: Fonte-Boa, Tobias.

3.2.2 Training

We use the SMP library to speed up the process. The encoder is based on resnext50 (<https://bit.ly/3I0aM3I>) and uses knowledge transfer from "ImageNet" weights. Dice loss is used for the loss function, and Intersection over Union (IoU) is the primary metric. IoU is a number from 0 to 1 that specifies the overlap between the predicted and labeled bounding box. An IoU of 0 means no overlap between the boxes, and an IoU of 1 means that the union of the boxes is the same as their overlap, indicating that they are entirely

²<https://labelstud.io/>

overlapping. We used in the training of 100 epochs, 646 (XZ slices) training images and 434 (YZ slices) images for validation, and the template (XZ and YZ slices) had 1080 images for testing. We obtained the images called templates through the porosity of the entire synthetic region provided by the geology team. Therefore, we treat this data as our reference. The region with greater porosity than zero was considered a turbiditic channel.

The following *data augmentation* techniques were used to increase the training base.

- horizontal flip
- affine transforms
- perspective transforms
- brightness/contrast/colors manipulations
- image blurring and sharpening
- gaussian noise
- random crops

All the techniques listed above are in the Albumentations³ package.

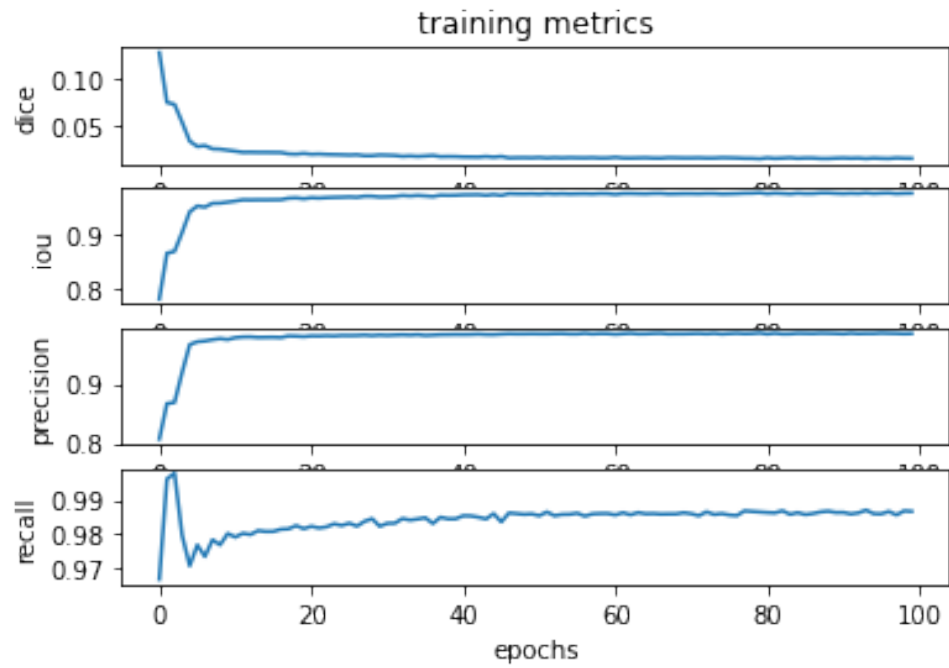
Finally, the trained network is run with the test images to determine where there are channels.

During training, the model U-Net achieved 95% Iou, 95% accuracy, and 98.5% recall. In validation, it achieved 68% in Iou, 92% in precision, and 74% in the recall. In the test, it got 86% on Iou, 99% accuracy, and 86% on recall. Due to the labels, the model generates a more expanded mask of the channels; that is, the labeling process incorporates pixels that the labeling specialist imagines are from the channel, with a tendency to increase this volume slightly. Furthermore, in the general volume, it generates a slightly more expanded model for the same reason.

The Figures 3.3, 3.4 and 3.5 show the evolution of the IoU, precision and recall metrics during training, validation and testing, respectively. The Figure 3.6 qualitatively presents the segmentation results applied to a YZ slice. The leftmost Figures represent the input images, the central figures represent the images segmented through the template, and the rightmost Figures represent the model prediction. It can be observed, as previously mentioned, that the predicted volume is, in general, more expanded than the template volume due to the bias introduced in the labeling process.

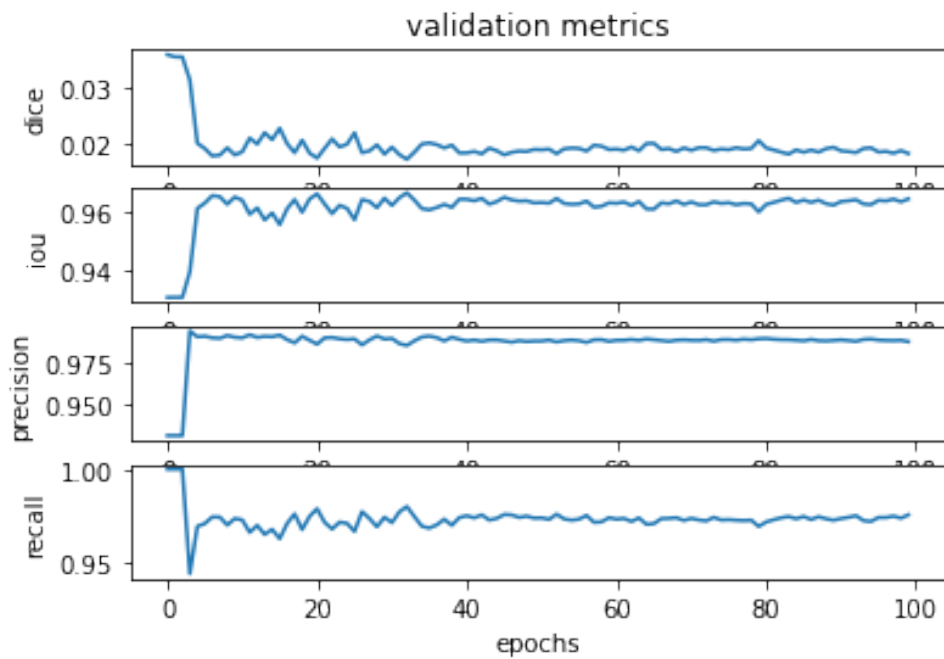
³<https://pypi.org/project/albumentations/>

Figure 3.3: Training results for the segmentation model.



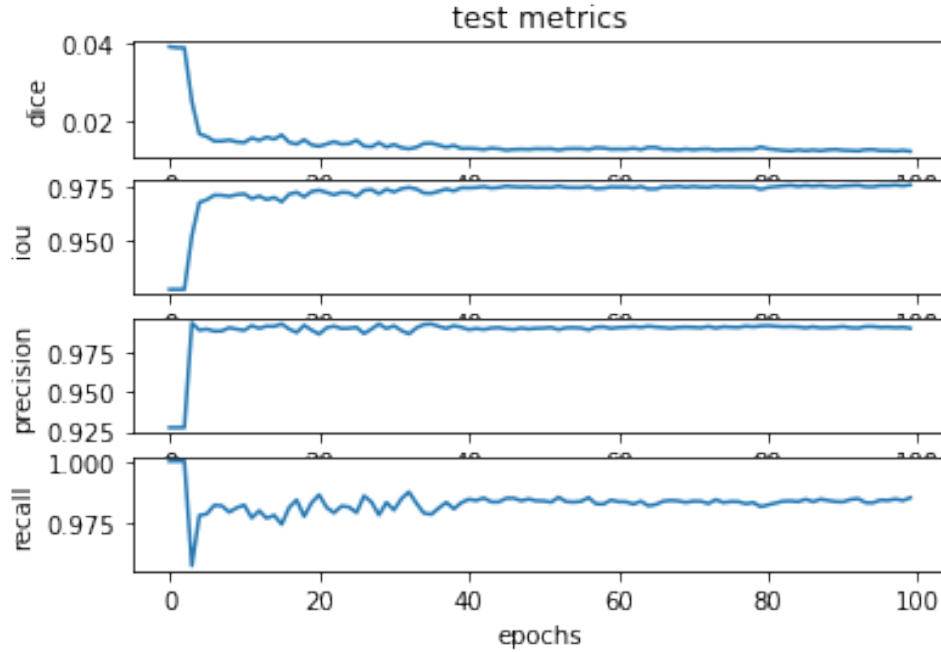
Source: The author

Figure 3.4: Validation results for the segmentation model.



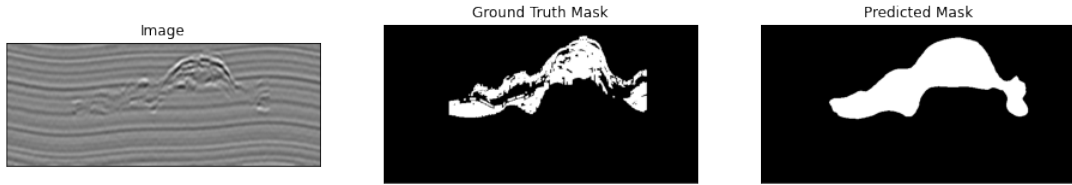
Source: The author

Figure 3.5: Testing results for the segmentation model.



Source: The author

Figure 3.6: Testing results for the segmentation model for the section 287 YZ.



Source: The author

3.2.3 Minimal amount of samples

Considering that labeling images can be expensive, as we would need a specialist to do so, especially if we are talking about thousands of images, we decided to verify with how many images, or a percentage of them, we could train and obtain a meaningful segmentation model. The training method is the same one used previous to the synthetic data. We used 646 (XZ cuts) training images and 434 (YZ cuts) images for validation and the ground truth (XZ and YZ cuts) with 1080 images for testing. We carried out the same training, but with each new training, we reduced the number of training images by half, keeping the validation and test images at the maximum. This method helps determine the minimum number of images needed to perform training and achieve convergency.

The results of the model metrics can be seen in the Tables 3.1, 3.2 and 3.3 for training, validation, and testing, respectively. The metric used to evaluate the quality of the model is *IoU*, which means *Intersection over Union*, which is a metric that evaluates the overlap of two detection boxes, or in this case, segmentation. Observing the mentioned Tables, we see that even when we reduce the number of labeled images from 646 (100%) to 20 (3,125%), the quantitative metrics used for evaluation, the result has a slight variation. This variation is mainly due to the significant imbalance between the number of points outside the channel and the number of points in the channel. However, if we analyze how these metrics evolve during training and testing, we see that using only 20 images is the evolution of the noisiest metric.

Figures B.1, B.2, and B.3 show the evolution of quantitative metrics during training, validation, and testing for 12.5% of labels. We suppressed the results for the other labels so the analysis did not become too repetitive. However, we observed that the metrics stabilize from 12.5% of labels and become very noisy for a smaller number of labels. In this way, we conclude that the model presents better stability when 12.5% of the images are labeled. This stability indicates that this problem has excellent potential for transfer learning since our model is pre-trained to perform semantic segmentation in other domains. Even with a small number of labeled images, it can specialize in segmenting the turbiditic channel.

We can do a qualitative analysis by looking at Figure 3.7. This Figure clearly shows the effect of a very drastic decrease in the number of labeled images. Note that when we use only 3.125% of labels, the algorithm fails in some sections (in this case, section XZ 200), incorrectly delimiting the turbiditic channel. Although this effect is relatively rare and does not drastically change the results of the quantitative metrics, we observed that when using 6.25% of labels (40 images), this problem does not appear for the same section.

Num samples	dice	IoU	Precision	Recall
100% (646)	0.01%	98%	99%	99%
50% (323)	0.04%	97%	97%	98%
25% (162)	0.04%	97%	97%	98%
12.5% (81)	0.05%	96%	96%	96%
6.25% (40)	0.05%	95%	95%	96%
3.125% (20)	0.05%	94%	94%	98%

Table 3.1: Training metrics.

Num samples	dice	IoU	Precision	Recall
100% (646)	0.03%	94.5%	96.5%	97.5%
50% (323)	0.033%	94%	97%	96.5%
25% (162)	0.04%	93%	97%	95%
12.5% (81)	0.037%	93%	97%	95%
6.25% (40)	0.03%	95%	97%	97%
3.125% (20)	0.05%	92%	94%	96%

Table 3.2: Validation metrics.

Num samples	dice	IoU	Precision	Recall
100% (646)	0.03%	94%	99%	94.5%
50% (323)	0.035%	93.5%	99%	94%
25% (162)	0.04%	92.5%	99%	93%
12.5% (81)	0.04%	92%	99%	92.5%
6.25% (40)	0.04%	93%	99%	93%
3.125% (20)	0.04%	95%	98%	96%

Table 3.3: Test metrics.

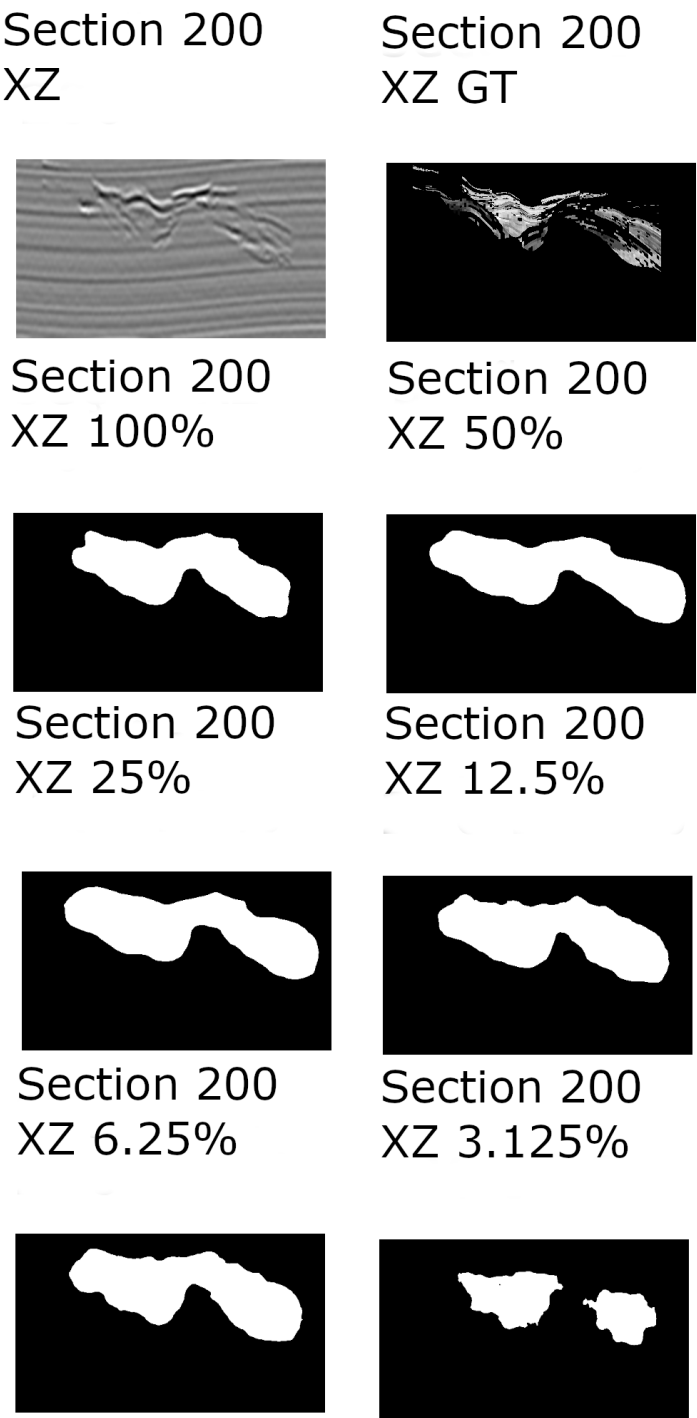
3.3 Marimbá field dataset

The previous experiments with the synthetic data showed that, training for identifying areas of interest conducted on synthetic data was reasonably practical when applied to actual data. However, this training is subject to the interpreter’s subjectivity since a geoscientist performs the labeling. As a way to mitigate this subjectivity, at this stage, a test was conducted using the synthetic data model of porosity that simulates turbidite channels, provided to the project synthetic data, as a label for the area of interest (Fig. 3.8). In other words, it eliminates the interpreter’s bias in segmenting areas of interest in this new phase.

The result of identifying areas of interest using the new approach, the porosity labels as masked labels, was similar to that of the previous approach with the manual labels. As we can see in Figure 3.9, the method was able to identify the channel feature located in the central portion of the seismic section (Fig.3.9A). Furthermore, although discontinuous, the new approach also identified contourite features.

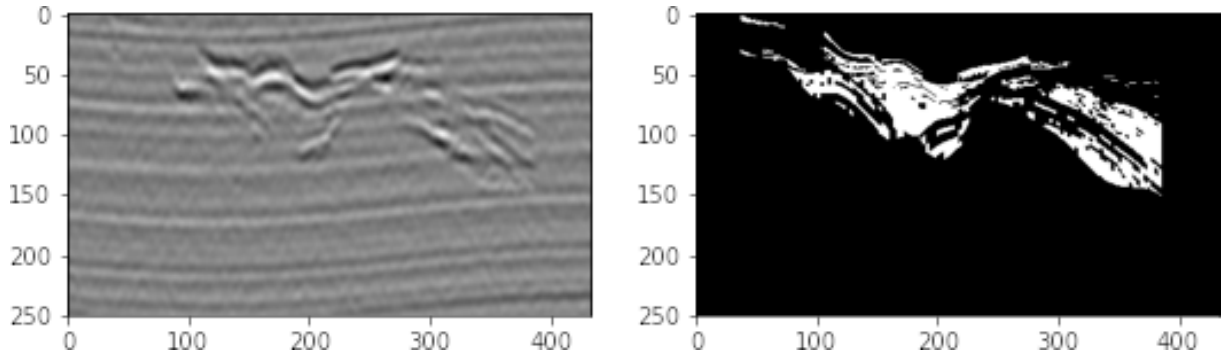
However, we noticed that the second attempt more efficiently identified other turbiditic systems not as evident in the Marimbá Field. In Figure 3.10A, we can observe the existence of a turbidite channel feature that was identified by the new approach (positive for the area of interest in V shape in the upper left corner of Figure3.10C) and was not as well delineated by the approach using the interpreter’s labeling (Fig. 3.10C).

Figure 3.7: Comparison of the segmentation of the 200 XZ section with different amounts of samples.



Source: The author

Figure 3.8: Seismic section and mask constructed from the synthetic porosity model. In the left Figure, white regions are positive indications for the area of interest (i.e., positive for the channel).



Source: The author

3.3.0.1 Expansion of the test area

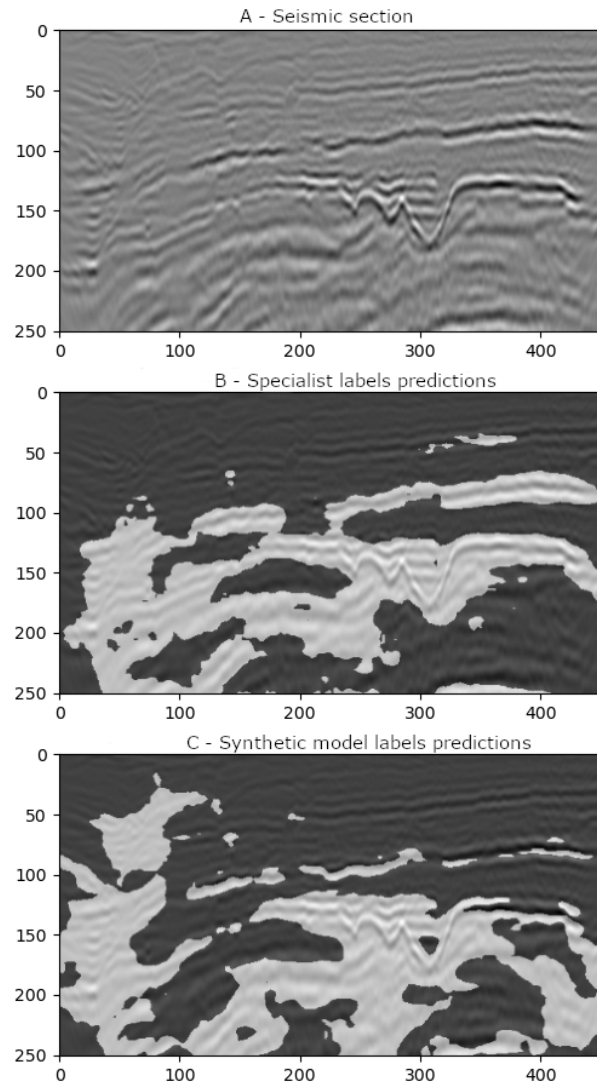
In addition to attempting to eliminate the interpreter's bias, at this stage of the project, we also decided to extend the segmentation tests of areas of interest to other regions of the Marimbá Field (i.e., actual data). In the previous stages, the tests were limited to a certain depth; however, the turbiditic systems of the Marimbá Field extend over a more extensive field area. With the aim of including the entire extension of the turbiditic systems of the Marimbá Field, we subdivided the seismic data into two parts. We executed the inference on each one of them.

As seen in Figure 3.11, the segmentation was quite efficient in other turbiditic systems of the Marimbá Field. Thus, the segmentation method we used can identify regions of interest with different morphologies, proving efficient when applied to a more extensive region or geographically distant regions within the same petroleum system context, in the case of the post-salt reservoirs.

3.4 Buzios Region dataset

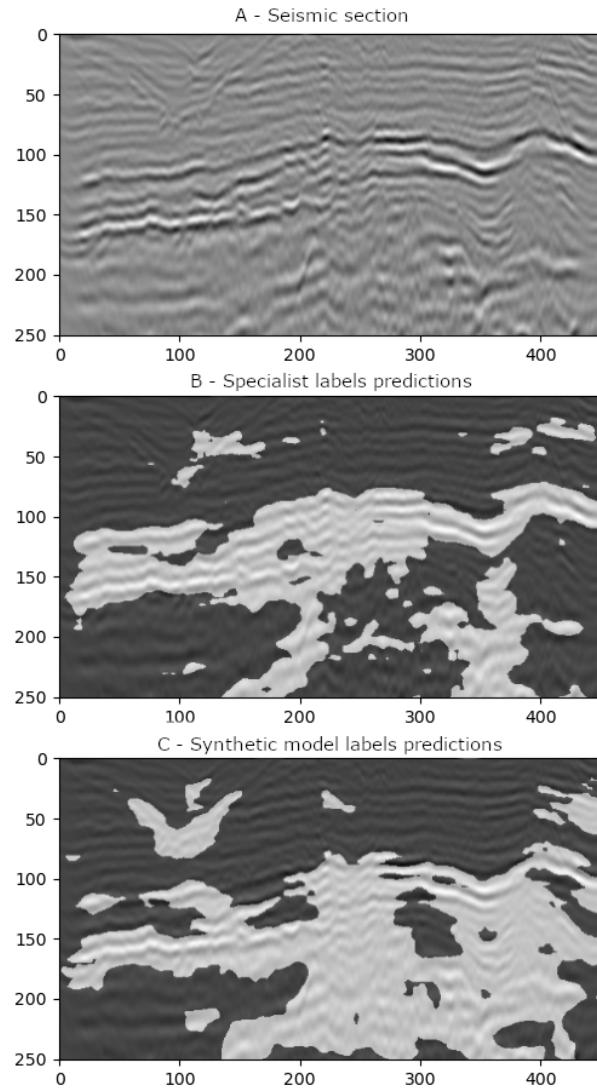
This data comes from a test area selected in the ANP data. The survey used in the cut was R0276_BS_500_FRANCO_FLORIM_PSDM.3D.PSDM.1-002. In this new dataset, we are not trying to identify turbiditic channels any longer; we are trying to separate the layers of salt, the post-salt, salt, pre-salt, and the basement. Our focus is

Figure 3.9: Comparison of results of seismic section 50 YZ (CL2052). A) Seismic section; B) Results of channel segmentation using labels with interpretative subjectivity; C) Results of channel segmentation using labels from the porosity model. Both approaches were able to identify multiple morphologies of the area of interest. Light portions indicate positive for the area of interest, and dark portions are negative.



Source: Fonte-Boa, Tobias.

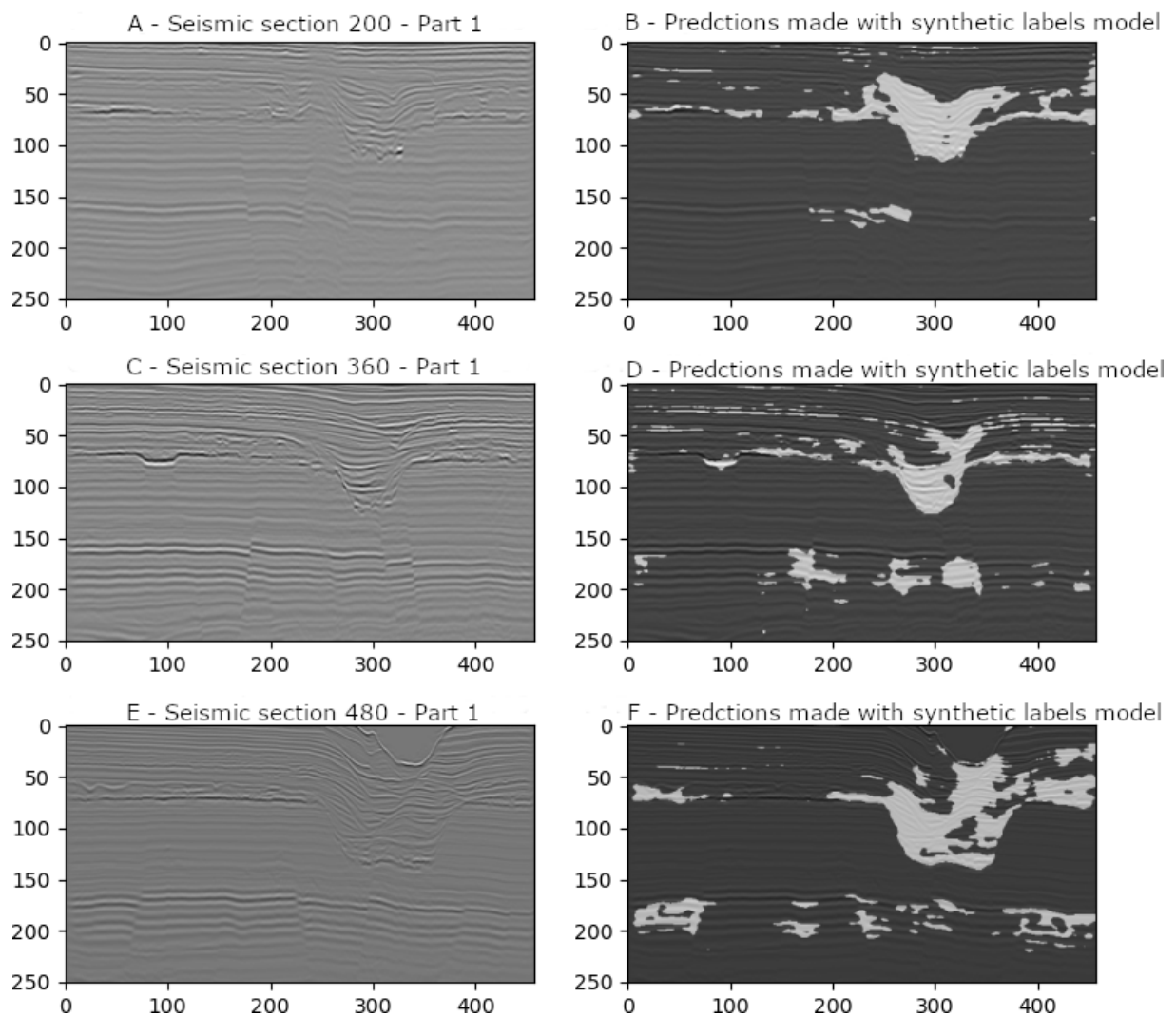
Figure 3.10: Comparison of results of seismic section 305 YZ (CL2562). A) Seismic section; B) Results of channel segmentation using labels with interpretative subjectivity; C) Results of channel segmentation using labels from the porosity model. Unlike the previous approach, the method using labels from the porosity model was able to delineate a channel feature in the upper left corner of the seismic section. Light portions indicate positive for the area of interest, and dark portions are negative.



Source: Fonte-Boa, Tobias.

specifically the pre-salt, and the segmentation is a great form to save resources. We can also have later uses for the post-salt and the salt areas; however, there is no use for the pre-salt basement area at the moment, as extracting oil from it is highly expensive[23].

Figure 3.11: Results of segmenting areas of interest in other regions of the Marimbá Field. The seismic section is on the left, and the segmentation result is presented on the right. The method could identify with great efficiency other turbiditic systems of the Marimbá Field. Light portions indicate positive for the area of interest, and dark portions are negative.



Source: Fonte-Boa, Tobias.

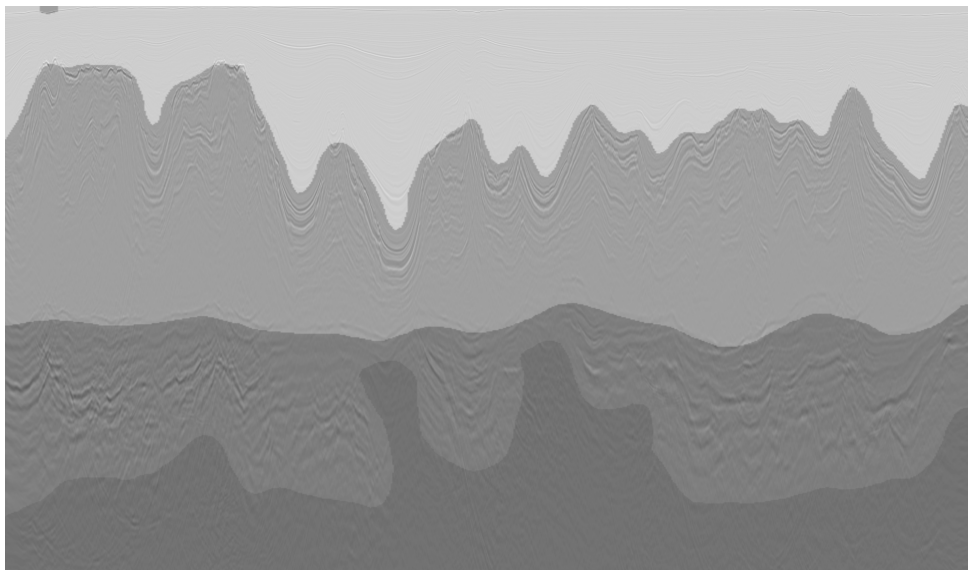
Initially, we opted to use a Transfer Learning (TL). TL is a technique that leverages pre-trained deep learning models to solve new tasks or domains [33]. It involves using the knowledge and representations learned from one task to improve the performance of another related task [44]. We used a TL from the model trained in the synthetic area to the pre-salt area to verify the viability of this particular solution and simultaneously check if we could identify turbiditic channels on those new images. The first experiment was to run the model train in the synthetic in the new images to see what it could segment.

The second experiment was to use newly labeled images from the ANP dataset and do a TL from the synthetic model on them. As for the images, we labeled 28 images from the XZ section. Therefore, with the labeled images, the same model of semantic segmentation of images is then used to train the identification of the post-salt, salt, pre-salt, and basement regions.

The following experiment was without TL, training a new model from scratch using the labels described above. The new pre-salt images were added, divided into 20 for training, 4 for validation, and 4 for tests. We used this number of images due to the difficulty of labeling. We used all those experiments to confirm the best method to segment new areas.

The results of the model metrics can be seen in Figures B.4, B.5, and B.6. The metric used to evaluate the quality of the model is *IoU*, which means *Intersection over Union*, which is a metric that evaluates the overlap of two detection boxes, or in this case, segmentation. Geologists were used to evaluating the model's quality due to the lack of a template for comparison. Some results can be seen in Figure 3.12.

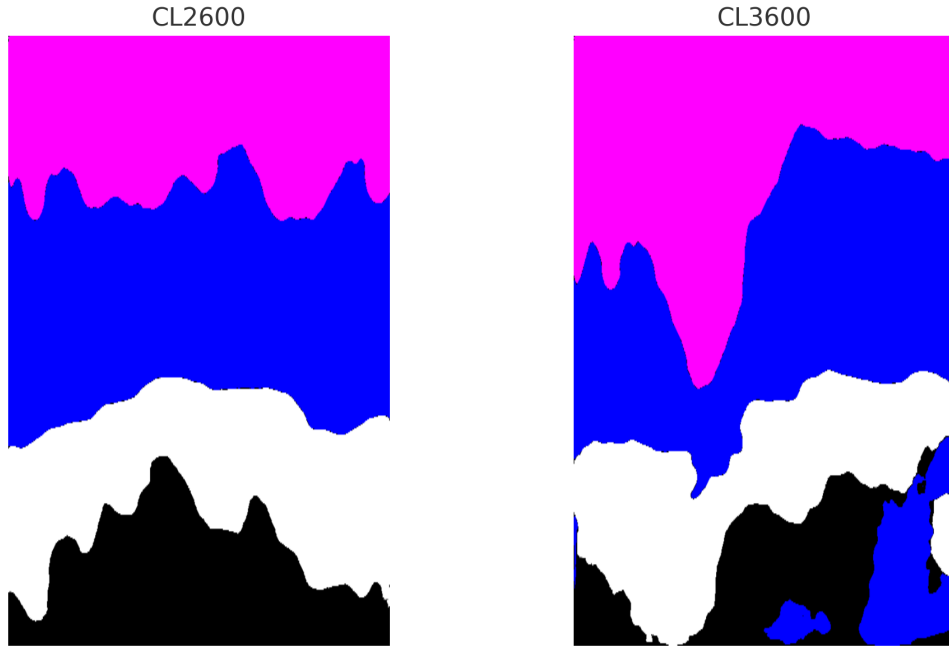
Figure 3.12: Overlapping of the region and segmentation of the section XZ 100



Source: The author

After this experiment, we received a new, smaller, and more controlled area for the

Figure 3.13: Comparison between two predictions in the Buzios region with the Crossline’s 2600 and 3600, one good and one bad, respectively. The predictions were made with the seismic data.



Source: The author

training and testing in the Buzios region. This time, specialists provided the labels for the training in a total of 20 training samples and 1 validation sample. The training follow the same pattern as before, with the TL, but this time, the TL came from the previous ANP training model. The TL made learning from the new data more manageable for the model. The segmentation worked within a specific range but could not perform properly for the entirety of the testing data, causing some lousy segmentation.

Figure 3.13 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL2600 is close to the training area, the prediction CL3600 is far. In the first one, it is visible that the four segmentation regions are separated in their own space, while in the second, some intrusion occurs, especially in the pre-salt region, which was intruded by salt and the basement. We can also see the salt in the basement region.

3.4.1 Semi-Supervised training approach

To circumvent this problem, we decide to use a semi-supervised approach. With the model able to segment and some good segmentation that it provide, we used those good segmentations on the training set and decided to move the specialist labels to the validation set. Although the model was capable of providing better segmentation, the problem with certain areas persisted. Then, we decided to extract seismic features from this data and evaluate whether we could improve the results in those poor-performance areas. The selected features were the instantaneous phase and the similarity. Since it was the same region and the same images but in a different form, it was possible for us to use the same labels as before, and so we did. According to the specialists, the features produced partially better results; while they were better in the pre-salt region, they were worse in the salt and post-salt regions. At this point, we opted for using an ensemble approach since we had a variety of models and results. We used a weighted majority vote technique to decide which part would be filled by each model, with the seismic data having a higher weight in the post-salt and salt regions, the feature data having a higher weight in the pre-salt and they all having the same weight in the basement. Due to some differences in the borderlines of each region, this particular method shows some things that could be improved, such as presenting empty areas.

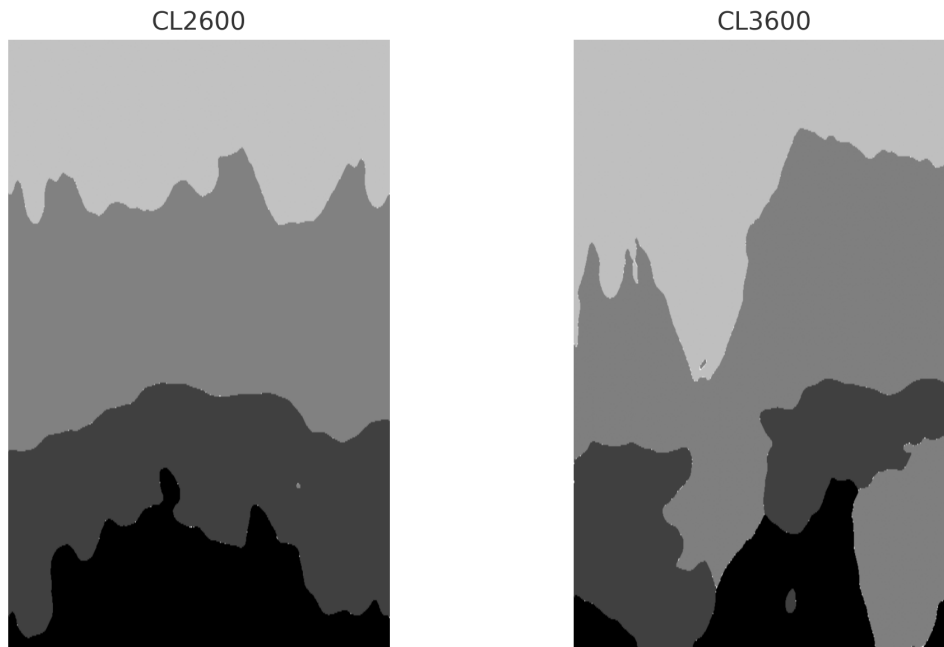
Figure 3.14 shows two side-by-side predictions. We used 300 sequential training samples for the Semi-Supervised model. Both were made outside the training and validation areas; while the CL2600 is close to the training area, the prediction CL3600 is far. In the first one, it is visible that the four segmentation regions are separated in their own space, while in the second, some intrusion occurs, especially in the pre-salt region, which was intruded by both salt and the basement. This Figure shows slight improvement.

Figure 3.15 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL2600 is close to the training area, the prediction CL3600 is far. In the first one, it is visible that the four segmentation regions are separated in their own space, while in the second, some intrusion occurs in all areas except for post-salt. According to the specialist, this feature segmented the pre-salt region better despite being worse elsewhere.

Figure 3.16 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL2600 is close to the training area, the prediction CL3600 is far. In the first one, it is visible that the four segmentation regions are separated in their own space, while in the second, some intrusion occurs, like the salt in the basement and the pre-salt in the basement. Despite not having some lousy errors as the intrusions, the segmentation lines above the pre-salt are poorly segmented.

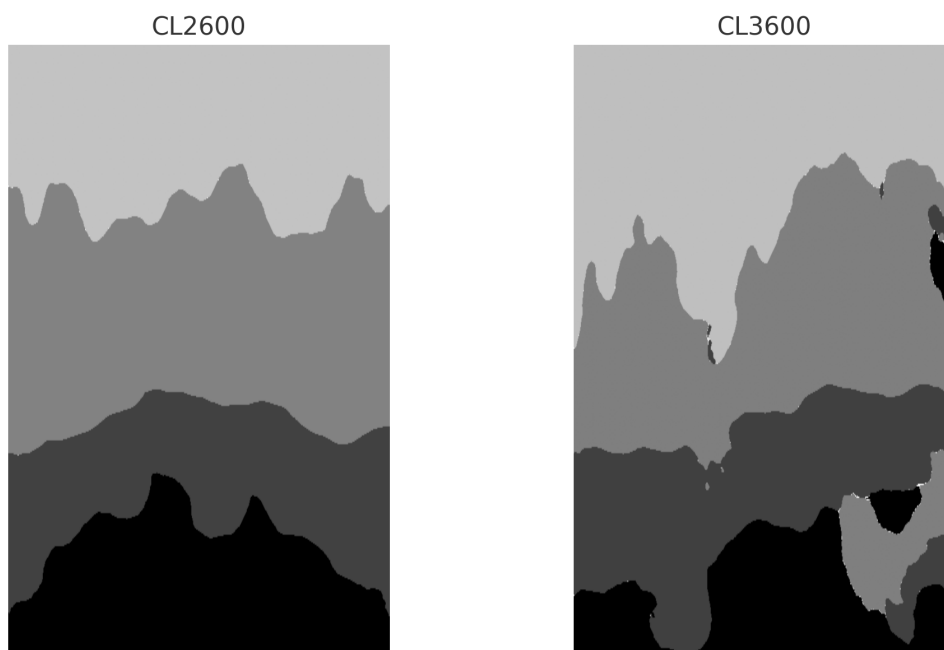
Figure 3.17 shows two side-by-side predictions. Both were made outside the train-

Figure 3.14: Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made with the seismic data.



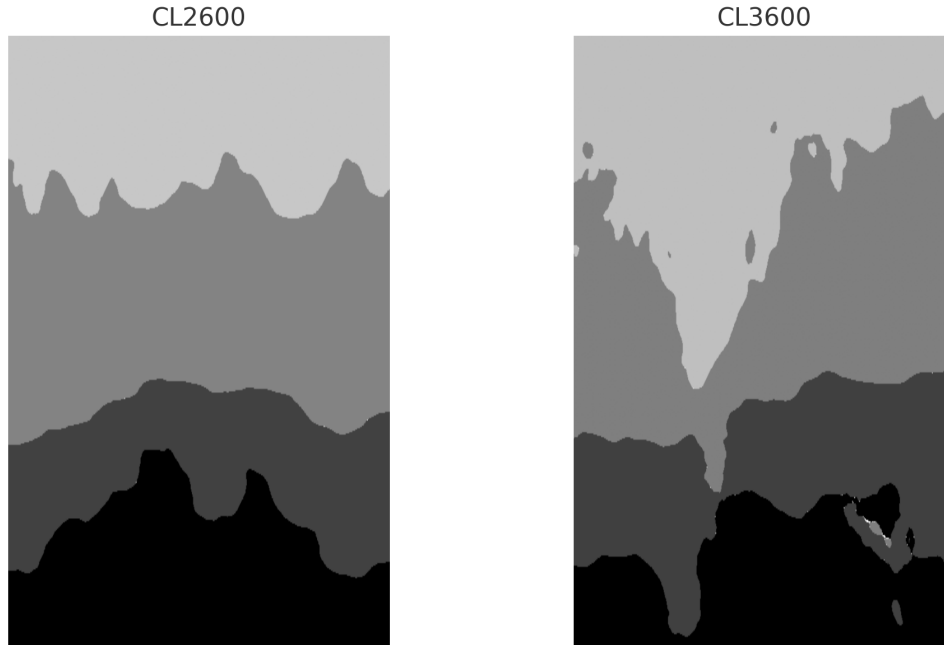
Source: The author

Figure 3.15: Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made with the instantaneous phase data.



Source: The author

Figure 3.16: Comparison between two predictions in the Buzios region with the Crossline’s 2600 and 3600, one good and one bad, respectively. The predictions were made with the similarity data.



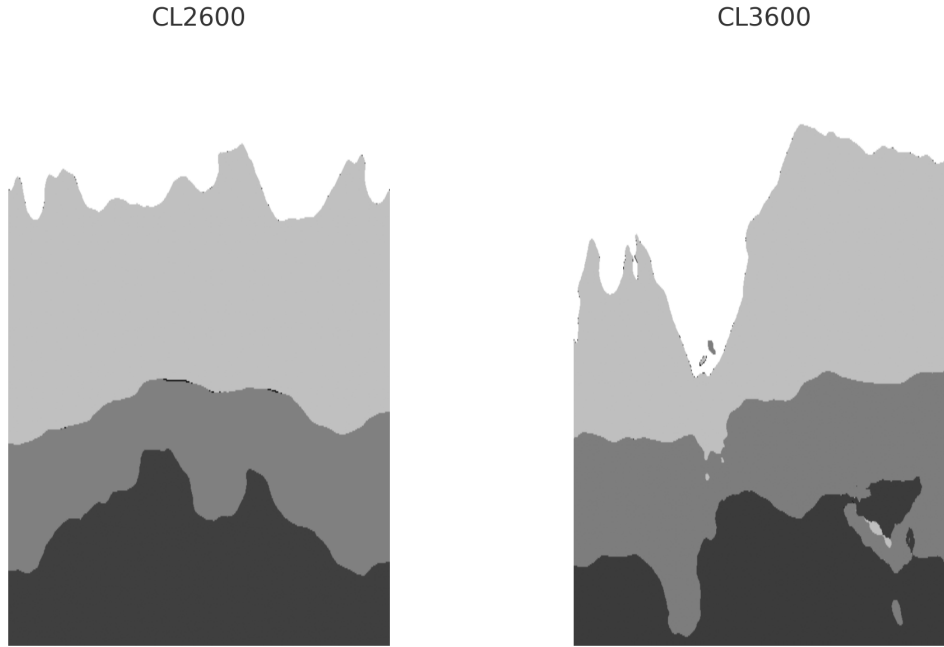
Source: The author

ing and validation areas; while the CL2600 is close to the training area, and in the region more to the middle right above the pre-salt is possible to see some disconnections between the regions provoked by the combination of the models. The prediction CL3600 is far from the training region. In the first one, it is visible that the four segmentation regions are separated in their own space, while in the second, some intrusion occurs, like the salt in the basement and the pre-salt in the basement. By combining the models, we assumed that the errors would be minimized, but instead, they were aggregated, even though not visible in this particular prediction.

3.4.2 Bayesian model

Due to the ensemble and its flaws, we were still determining whether the predictions were correct. So, we opted to introduce a Bayesian model. However, as those models can be costly, we chose the epistemic uncertainty method that could provide the results that we needed at a low computational cost. This method works by applying the Dropout. We train the model with the Dropout at a specific value. At the inference moment, we

Figure 3.17: Comparison between two predictions in the Buzios region with the Crossline’s 2600 and 3600, one good and one bad, respectively. The predictions were made by combining the model’s predictions.



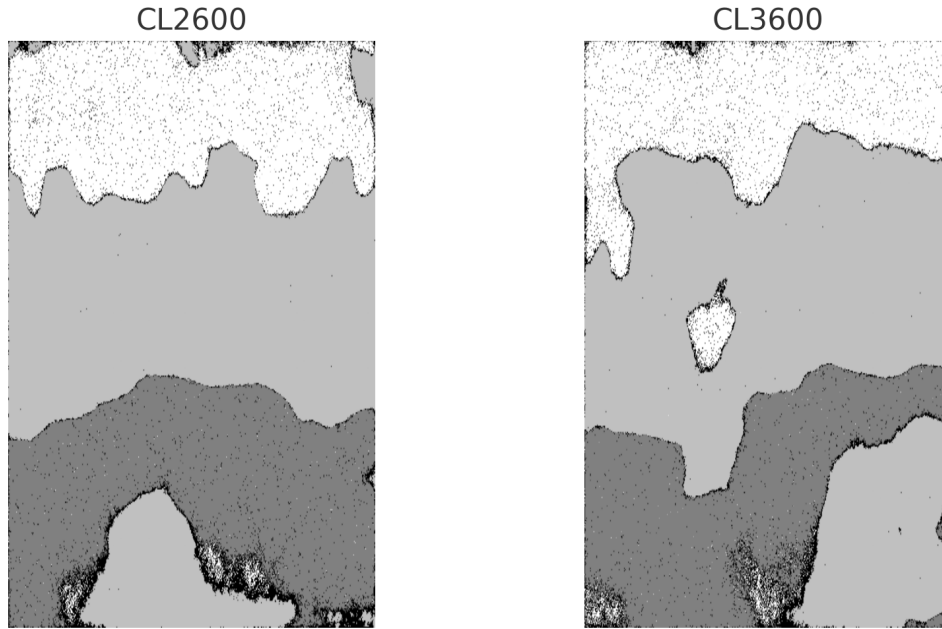
Source: The author

maintain the model in the training mode so that it can continue applying the Dropout. We execute a Monte Carlo sampling method with the Dropout, taking the average for the predictions and the uncertainty within this method. This Bayesian model showed that the model has a high degree of uncertainty in the borderline regions and also in the areas where the ensemble makes subpar predictions. We also used a weighted loss combining the CrossEntropy and Dice losses. Segmentations often use either loss, and after several tests, we concluded that the best values are $(0.1 * CrossEntropy + 0.9 * Dice)$. An extensive test determined the Dropout’s best value, which is 0.1.

Figure 3.18 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL2600 is close to the training area, the prediction CL3600 is far from the training region. This model was the initial draft of the Bayesian model using the same training methods as before. At this point, [13] methodology, we kept the Dropout value as 0.5, which hindered the model’s learning capabilities. This fact is best seen not just by the visible errors and missing basements but in the granularity created by the predictions, which never existed before.

Figure 3.19 shows two side-by-side confidence intervals (CI). Both were made outside the training and validation areas; while the CL2600 is close to the training area, the prediction CL3600 is far from the training region. In the CI, the darker the blue, the lower the value; the darker the red, the higher the value, while white is the middle

Figure 3.18: Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made using the Bayesian model's predictions for the segmentation masks.



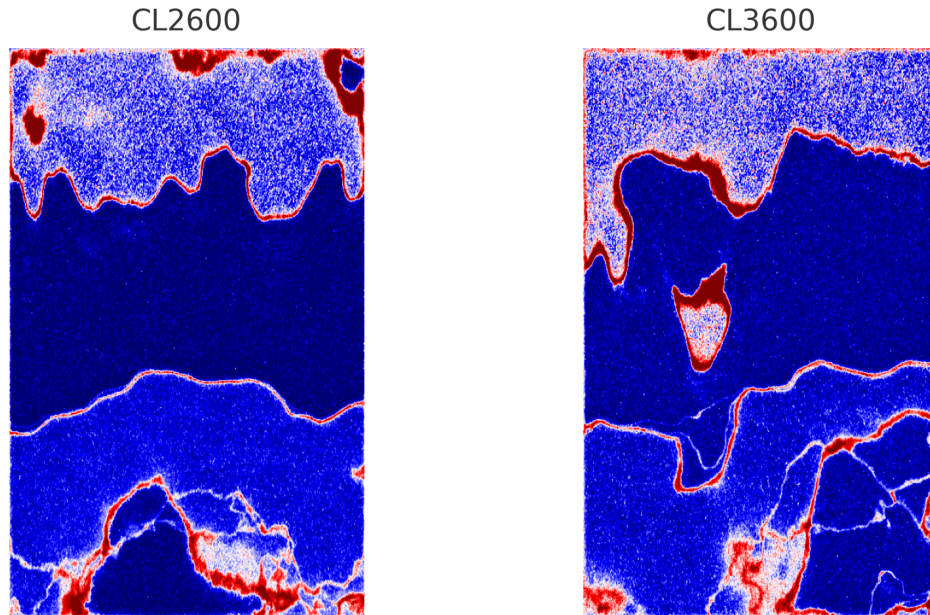
Source: The author

between them. This Figure shows little confidence in the results as the blue is not dark, and there is much red.

Figure 3.20 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL3100 is close to the training area, the prediction CL3600 is far from the training region. Now we have done a gridsearch for the best parameters, changed the amount of training samples, and used the combined model's predictions for the results. Even though the improvements are, if any, marginal, now we can know at some level the certainty of the model's predictions.

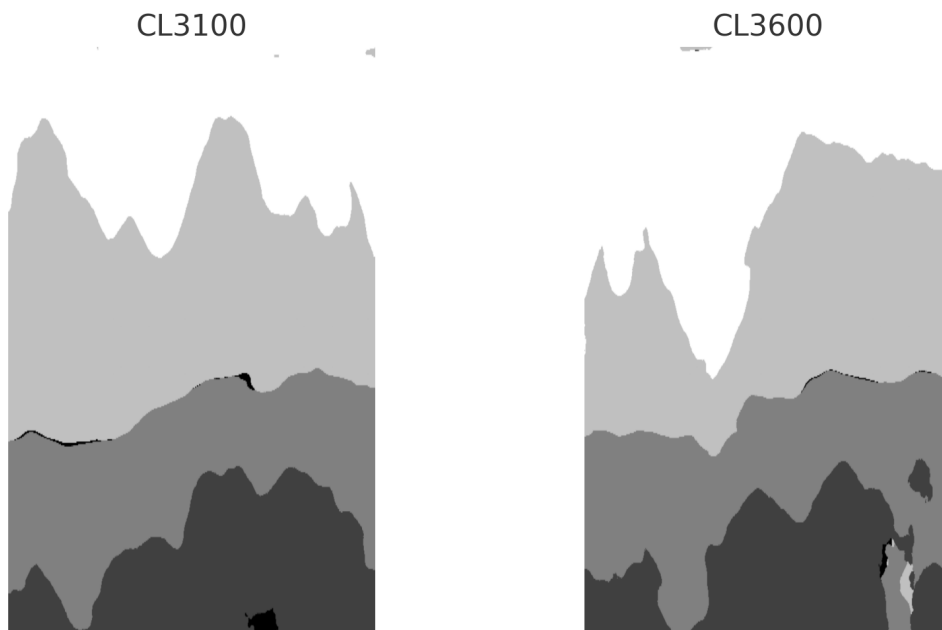
Figure 3.21 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL3100 is close to the training area, the prediction CL3600 is far from the training region. The seismic data model practically does the superior region (post-salt), so it can maintain the one borderline. However, from the salt region and beyond, as we have different models with different predictions, we have a different borderline for each and for each low-confidence prediction.

Figure 3.19: Comparison between two predictions in the Buzios region with the Crossline's 2600 and 3600, one good and one bad, respectively. The predictions were made using the Bayesian model's predictions for the confidence interval.



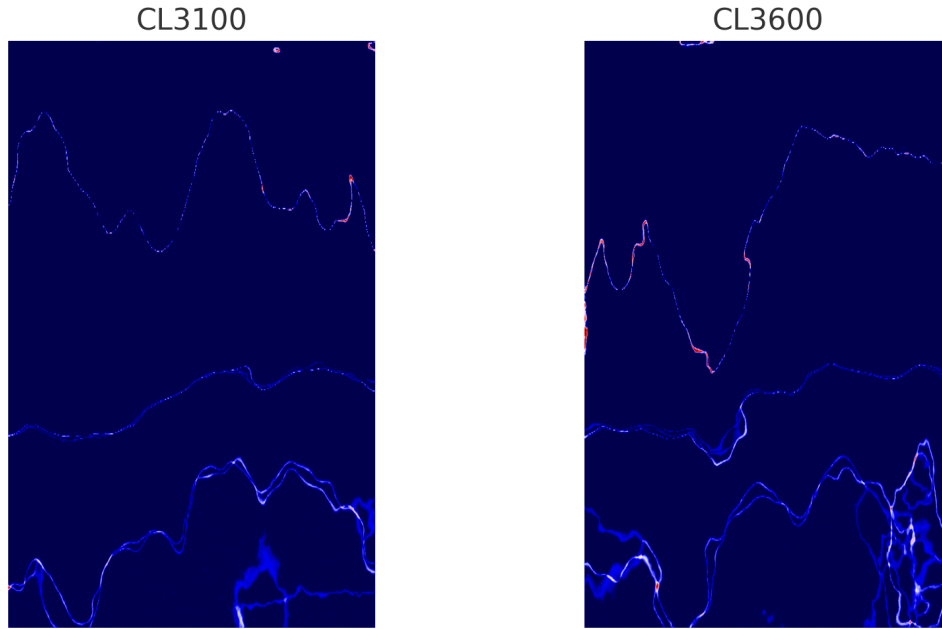
Source: The author

Figure 3.20: Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, one OK and one bad, respectively. The predictions were made using the parameters adjusted to the Bayesian model's predictions for the segmentation masks.



Source: The author

Figure 3.21: Comparison between two predictions in the Buzios region with the Crossline’s 3100 and 3600, one OK and one bad, respectively. The predictions were made using the parameters adjusted to the Bayesian model’s predictions for the confidence intervals.



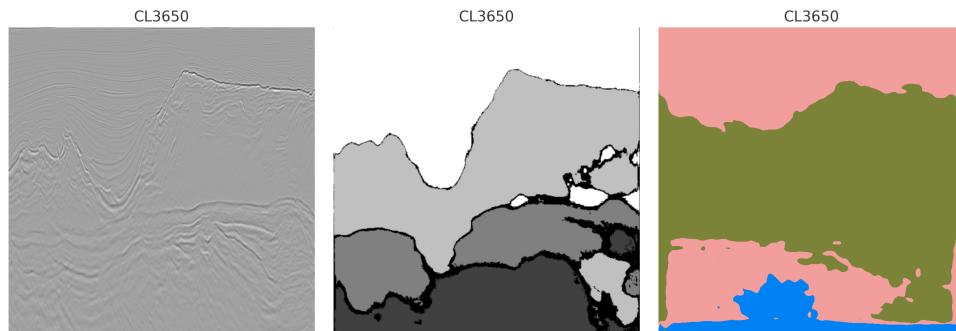
Source: The author

3.4.3 SFM

As shown in the chapter 2, one of the state-of-the-art models for geobodies segmentation is the SFM. We decided to benchmark ourselves against them. As they trained the Seismic Foundation Model with more extensive datasets, we believed that this model could perform better than the U-Net model, which was trained only with our tiny dataset and semi-supervised method. Even though it was cumbersome dealing with the SFM, we finetuned the model with our dataset with only the specialist’s labels. Then, we inferred the lousy samples, verifying whether the predictions were better or not. Also, for a sanity check, we decided to train our model with the dataset provided for the finetuning of the SFM.

Figure 3.22 shows three images side-by-side and in a different size due to the SFM only accepting 768x768 inputs. The prediction CL3650 is far from the training region. Due to the lack of resources for the training of the SFM, we made the finetuning using the same initial 20 samples dataset. This method would give us clues about the SFM capabilities and whether it could provide good predictions with fewer samples. The Figure shows first the seismic data as a reference for what both models were trying to predict. The second image is the prediction of the U-Net model when using only the 20 samples provided by

Figure 3.22: Comparison between two predictions in the Buzios region with the Crossline’s 3650. The first image is the seismic data, the second is the U-Net prediction, and the third is the SFM prediction.



Source: The author

the specialist. The third image shows the equivalent prediction for the same section when trained with the 20 samples provided by the specialist. It was bad, as only three classes are visible in the prediction and barely follow any pattern in the seismic data. Considering the resources that we would need for a better result, we decided to utilize this model no longer, not even for state-of-the-art comparisons.

3.4.4 Feature Pyramid Network

As the results with SFM were disappointing, and we been having troubles with the U-Net model in the region far from the training area, we decided to check other architectures present in the `segmentation_models_pytorch` library. We also opted to check different backbones and initial weights. We performed a gridsearch with the models:

- FPN,
- U-Net,
- U-Net++.

As the backbones, we used:

- `se_resnext50_32x4d`,
- `efficientnet-b7`,
- `se_resnet101`.

- resnext101_32x8d,
- densenet121,
- inceptionv4.

Moreover, for the initial weights were used:

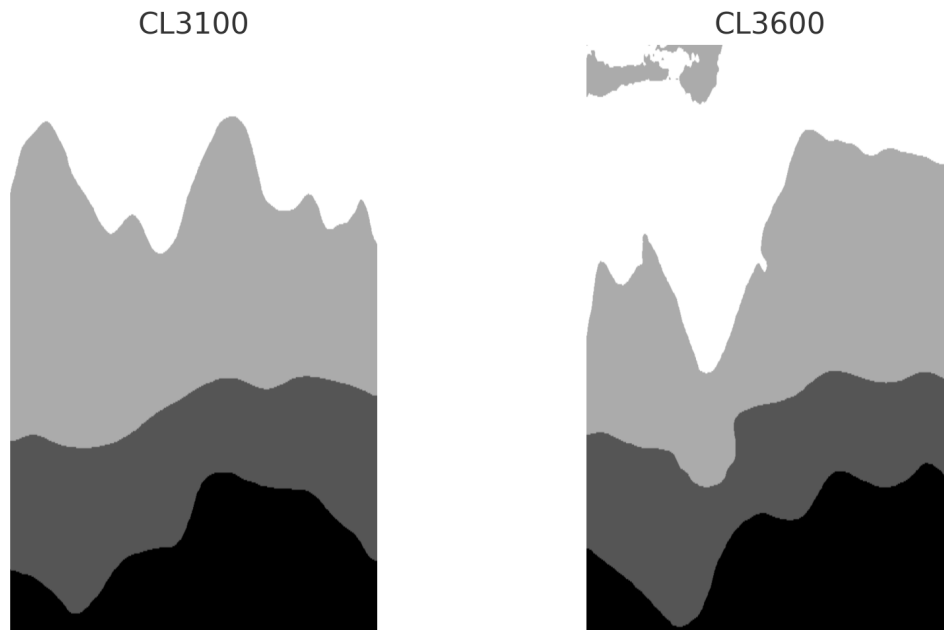
- imagenet,
- swsl (Semi Weak Supervised Learning),
- ssl (Self-Supervised Learning).

The best combination was the FPN with the resnext101_32x4d and ssl. This combination, along with the Semi-Supervised technique, provided the most accurate segmentations with precise uncertainty intervals, almost erasing the lousy predictions made by the U-Net model before. We obtained this result using the seismic data alone, without the seismic features or the ensemble model, showing the full extent of the FPN capabilities. Therefore, we decided to change our primary segmentation model to the FPN from this point onward.

Figure 3.23 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL3100 is close to the training area, the prediction CL3600 is far from the training region. We already had the best parameters, so we used only the seismic data and changed the base model to the FPN. As it is perceptible, there are no more intrusions in the salt, pre-salt, and basement regions. However, there are some intrusions in the post-salt done by the salt, this does not interfere with our objective of pre-salt segmentation as the post-salt region is irrelevant at this point. The FPN model also eliminates the need for other seismic data as it can solve the pre-salt issues independently.

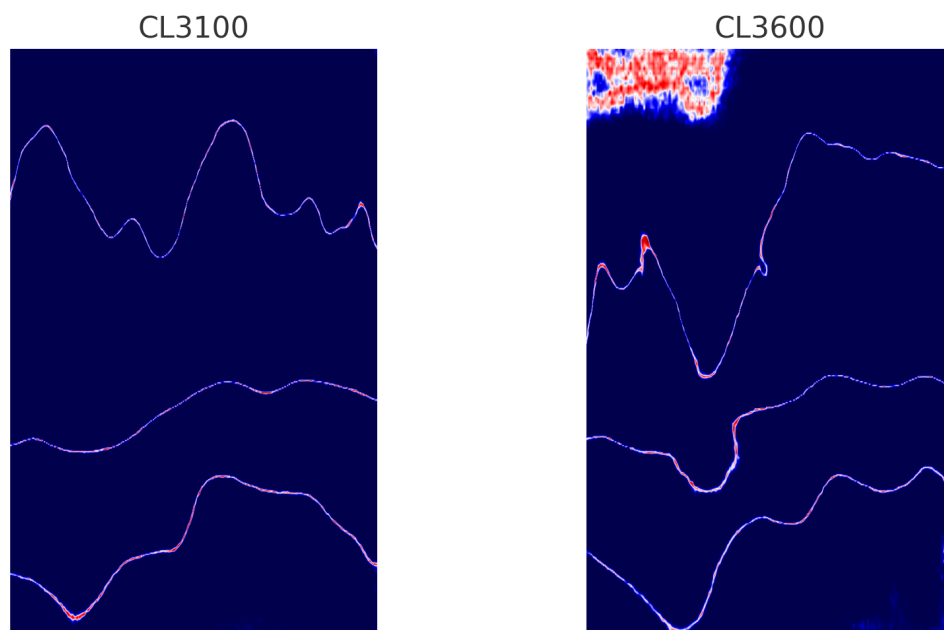
Figure 3.24 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL3100 is close to the training area, the prediction CL3600 is far from the training region. We already had the best parameters, so we used only the seismic data and changed the base model to the FPN. The CI shows that the model has little confidence in the wrong predictions made at the top of the post-salt and also shows that there might be something wrong at the right side of the basement, something that might pass unnoticed in the regular prediction.

Figure 3.23: Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, one good and one OK, respectively. The predictions were made using the parameters adjusted to the Bayesian model with the FPN and seismic data only for the predictions of segmentation masks.



Source: The author

Figure 3.24: Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, one good and one OK, respectively. The predictions were made using the parameters adjusted to the Bayesian model with the FPN and seismic data only for the predictions of confidence intervals.



Source: The author

3.4.5 Errors and Fixes

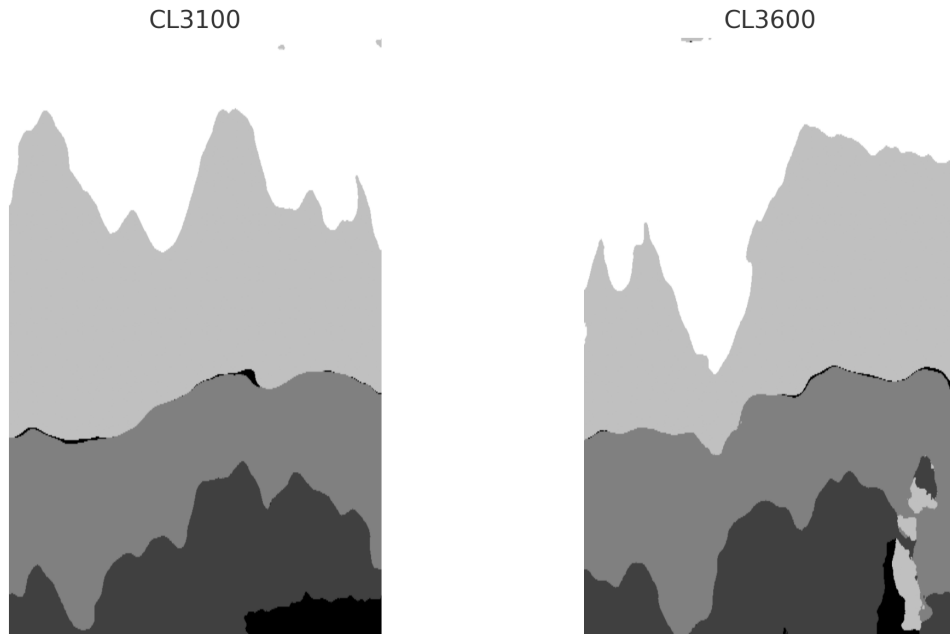
As we performed additional tests, we discovered some inconsistencies. While the first model with the seismic amplitude performed well, all the other features did not. We also tried using histogram equalization and standard deviation with the seismic amplitude but to no avail. The results were terrible, and the middle model produced inconsistent labels for the semi-supervised model. We also could not reproduce the best result that we achieved. After many trials, we discovered the root of the problem: the decoder. The initialization of the weights is random and does not respect the seed. To address the issue, we created a universal base model, trained for a few epochs with very little data, to allow the models to start from the same point and ensure they always converge. While that helps the middle model, the semi-supervised still had problems. The main problem was that the middle model only sometimes produced the best labels, and those imperfect labels propagated all the errors, making the model underperform. We decided that a way to fix this issue was to utilize an edge detection technique and a moving average. The average collects n labels and does an edge detection in every single one, then counts the number of edges, usually 3, and starts comparing with the following labels. If the next label has more edges than the average, the label is removed from the training and moves to the next one. This simple technique allowed us to obtain better results without any human interaction. At this point, we also decided that we no longer needed the Bayesian model because it hinders the performance and does not show any advantage regarding the regular model, as the uncertainty shown in the map is also visible in the segmentation mask.

Figure 3.25 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL3100 is close to the training area, the prediction CL3600 is far from the training region. We already had the best parameters, so we used only the instantaneous phase data and changed the base model to the FPN. The results are alarming, showing much confusion about the model's predictions. This also happened with all the other models using different data. This inability to achieve better results, and we had already produced one before, was the trigger for the search for the inconsistencies.

Figure 3.26 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL3100 is close to the training area, the prediction CL3600 is far from the training region. This Figure is an excellent example of what the model is utilizing for the training. Despite this, two sections are not present in the training; they reflect what the model saw in the training and reproduced on them. As we eliminated this kind of section in the training with edge detection, the model began to produce better results.

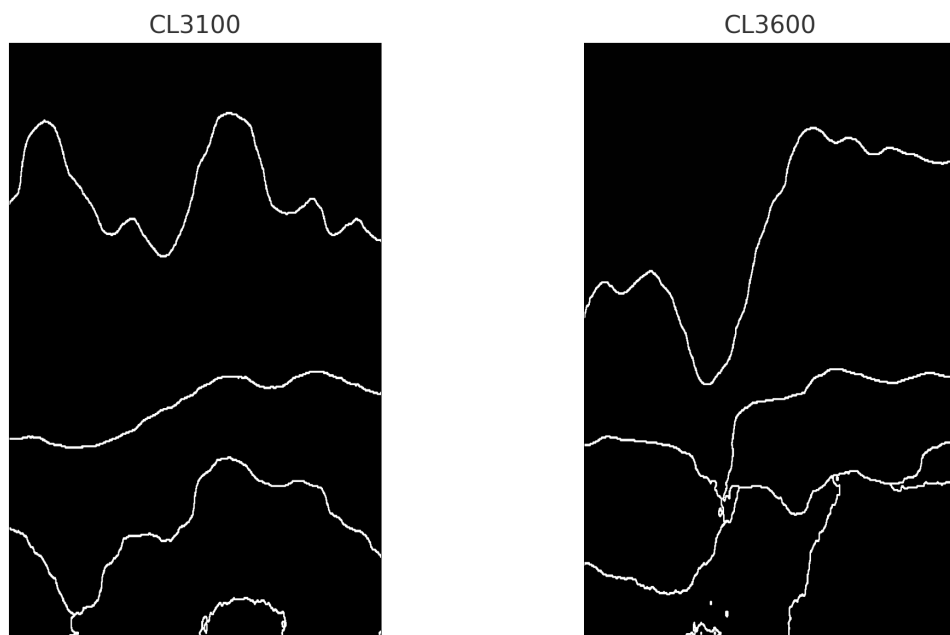
Figure 3.27 shows two side-by-side predictions. Both were made outside the train-

Figure 3.25: Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, one Ok and one bad, respectively. The predictions were made using the parameters adjusted to the Bayesian model with the FPN and instantaneous phase data only for the predictions.



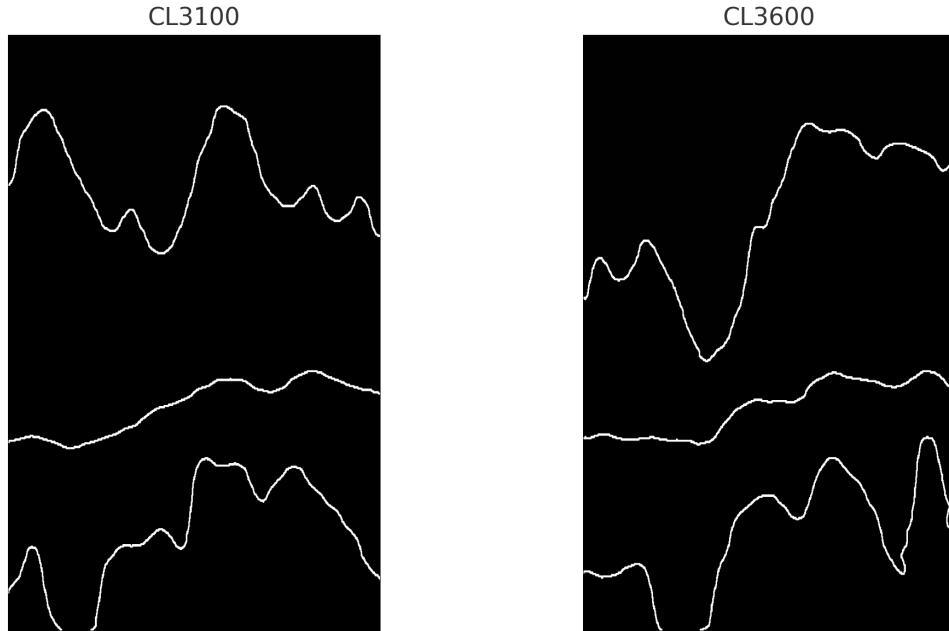
Source: The author

Figure 3.26: Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, both bad. The predictions were made using the parameters adjusted to the FPN model and instantaneous phase data only for the predictions.



Source: The author

Figure 3.27: Comparison between two predictions in the Buzios region with the Crossline's 3100 and 3600, both good. The predictions were made using the parameters adjusted to the FPN model and instantaneous phase data only for the predictions.

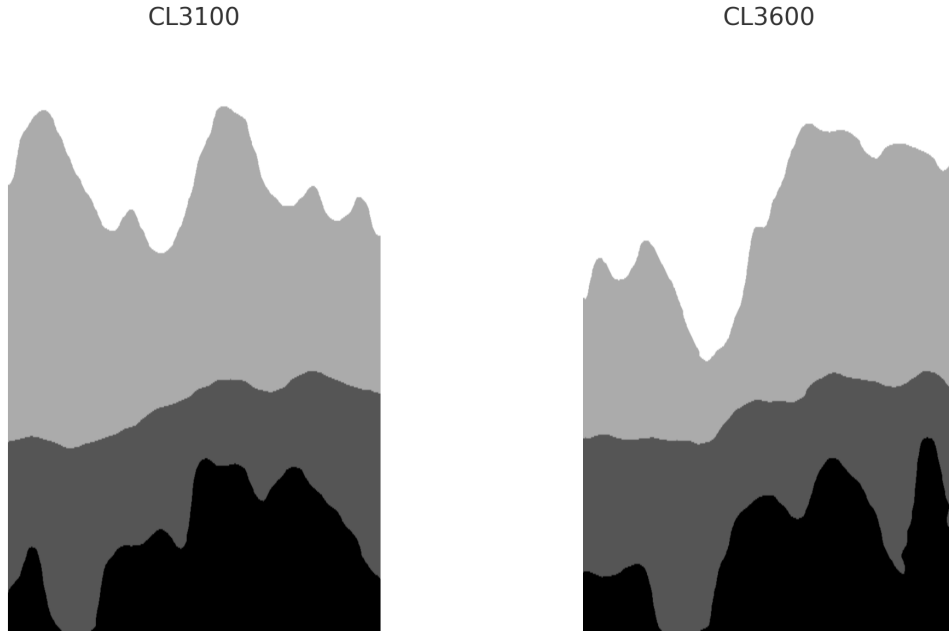


Source: The author

ing and validation areas; while the CL3100 is close to the training area, the prediction CL3600 is far from the training region. We already had the best parameters, so we used only the instantaneous phase data and changed the base model to the FPN. This Figure is an excellent example of what the model uses for the training after the edge detection. Now, the model usually sees the 3 edges as it is supposed to, making it easier for it to perform correctly as the errors are no longer introduced or propagated.

Figure 3.28 shows two side-by-side predictions. Both were made outside the training and validation areas; while the CL3100 is close to the training area, the prediction CL3600 is far from the training region. We already had the best parameters, so we used only the instantaneous phase data and changed the base model to the FPN. The results are far superior to anything we had before using the instantaneous phase. This was achieved thanks to edge detection by eliminating the imperfect labels and training with only the good ones, even if it means reducing the total of training samples, showing that quality is better than quantity.

Figure 3.28: Comparison between two predictions in the Buzios region with the Crossline’s 3100 and 3600, both good. The predictions were made using the parameters adjusted to the FPN model and instantaneous phase data only for the predictions.



Source: The author

3.5 Discussions

The original U-Net model demonstrated strong performance in biomedical image segmentation tasks, particularly with limited labeled data. In this work, the U-Net architecture achieved up to 94% IoU on the synthetic dataset using all labeled slices and remained stable with only 12.5% of labels, achieving 92% IoU in testing. When applied to real seismic data from the Marimbá and Búzios regions, qualitative results remained consistent within the training area, but with some inconsistency far away from it. While the datasets and metrics differ, the observed label efficiency and segmentation stability confirm U-Net’s effectiveness in data-scarce domains. The results are aligned with, but not superior to, the original U-Net benchmarks. This model was later discarded as we found that the FPN model performed better with the same dataset.

The SFM introduced a transformer-based architecture with self-supervised pre-training on over 2.2 million seismic images, achieving superior generalization and accuracy (mIoU) across seismic facies classification tasks. In this research, our pipeline was qualitatively compared to the SFM. Although our method achieved visually comparable results on selected sections, no quantitative evidence indicated superior performance. Direct numerical comparison is not possible due to dataset differences. Our method pro-

duced comparable qualitative results, but no evidence of better performance was found. Due to the complexity of the training we could not fully explore this model, only conduct tests with small data samples.

Bayesian SegNet uses Monte Carlo dropout to quantify pixel-wise uncertainty, improving performance by 2–3% in mean IoU on small datasets. In this dissertation, Monte Carlo dropout was employed to diagnose uncertainty in seismic segmentation, identifying low-confidence regions but not incorporated into final predictions. We have not directly used the Bayesian SegNet model, as we opted for using a similar approach with the U-Net model. Therefore we have no real comparison between the methods.

FPN improves multi-scale feature representation for object detection. In our experiments, FPN integrated into seismic segmentation pipelines (particularly in the Búzios region) yielded more robust predictions compared to U-Net, especially when combined with edge detections techniques for automatic labels disposal. FPN outperformed the baseline U-Net within our dataset. While not directly comparable to the COCO results from the original paper, our findings are consistent with the literature regarding improved multi-scale representation. As this model was the better performer, it was chosen as the main model. This model also take advantage of the edge detection technique, providing only good samples for the semi-supervised regime.

Epistemic Uncertainty estimation via Dropout shows that dropout approximates Bayesian inference, capturing model uncertainty efficiently. In this research, Monte Carlo dropout was implemented accordingly to visualize epistemic uncertainty across seismic sections. However, no quantitative improvement in predictive metrics was observed. Our application aligned conceptually with the original formulation, though without measurable accuracy enhancement. Unfortunately due to the simplicity of this technique, the high computational costs and the low gains, we opted for not going forward with this technique.

ResNet architectures enable deeper models without vanishing gradients, achieving state-of-the-art accuracy on ImageNet. In the porosity estimation experiments, the autoencoder with a ResNet-18 encoder performed worse than U-Net-based encoders, producing less spatially coherent porosity maps. In our context of spatial regression with seismic data, the U-Net encoder outperformed ResNet-18. The tasks are not directly comparable, but U-Net proved more suitable for 2D porosity mapping.

These results validate the methodological choices of this dissertation: the adoption of U-Net/FPN architectures, Bayesian uncertainty diagnostics, and proved effective for data-scarce, geologically complex environments, aligning with the principles established in the foundational works reviewed, even if all the techniques were not selected as the final solution.

Chapter 4

Porosity Estimation

4.1 Bias correction model

[24] provided the geostatistical regression model that possesses a particular bias from the petroelastic inversion data used to train, which causes it to underestimate the porosity value. The general idea is to use a model to learn the data distribution and correct the bias, improving the geostatistical model quality overall.

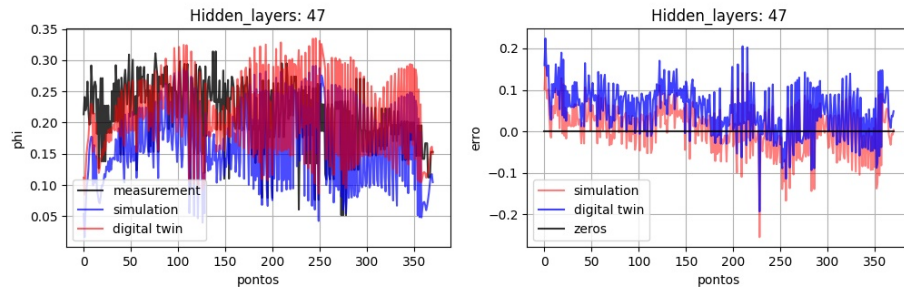
4.1.1 Multi-layer Perceptron

The first idea was to use an Multi-layer Perceptron (MLP) model, the simplest model possible, to validate the idea without doing an overkill. The MLP model uses the geostatistical model's prediction as an input, the absolute of the ground truth, and the geostatistical model subtraction as the output, using the mean squared error (MSE) loss. Then, the MLP model makes the sum of the prediction made by the geostatistical model and the prediction made by the MLP to try to correct the bias. We made some exhaustive tests to determine the ideal value of hidden layers for the MLP model. The value of the best fit was 47 hidden layers. This model did not remove the bias and could only add the found constant to every value. Therefore, we decided to try a convolutional model that could "see" the neighborhood and use that to predict a value.

The results are in the figures below. Figure 4.1 shows at the left the difference in the porosity values from the ground truth (black), the geostatistical model (blue), and our MLP (red). The right shows the error difference between the geostatistical model against the ground truth (blue) and the MLP model (red). The figure 4.2, show the porosity values for the predicted section 287 YZ, in the same pattern as the figure 4.1, and 4.5 are their respective error heatmaps. Those particular sections were used as they were far

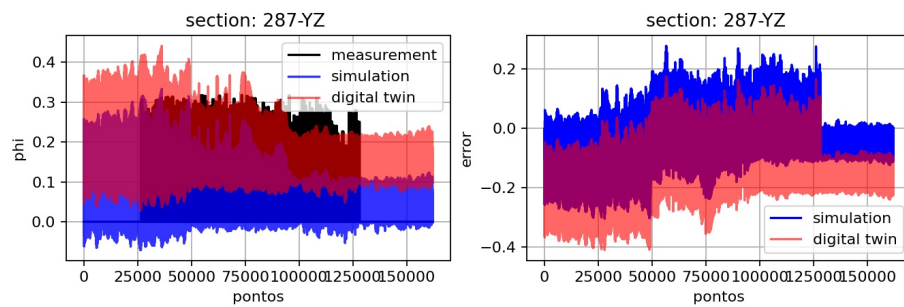
from the well, and it was necessary to know how the geostatistical model would fare in sections far away from the training data. For reference, the figure 4.3 is the ground truth and the 4.4 is the geostatistical model the we are trying to remove the bias.

Figure 4.1: Predictions of the MLP model with 47 hidden layers to the well data.



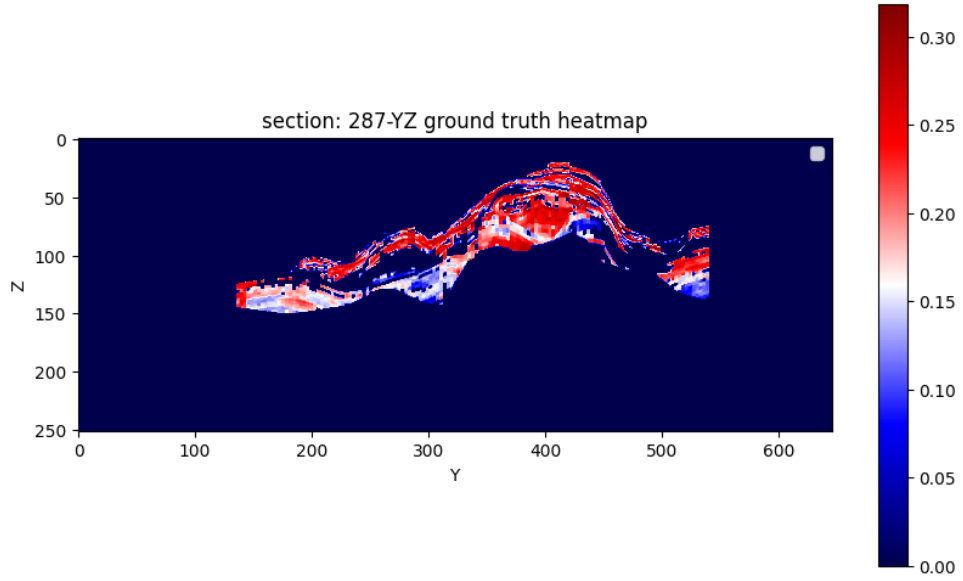
Source: The author

Figure 4.2: Predictions of the MLP model with 47 hidden layers for section 287 YZ.



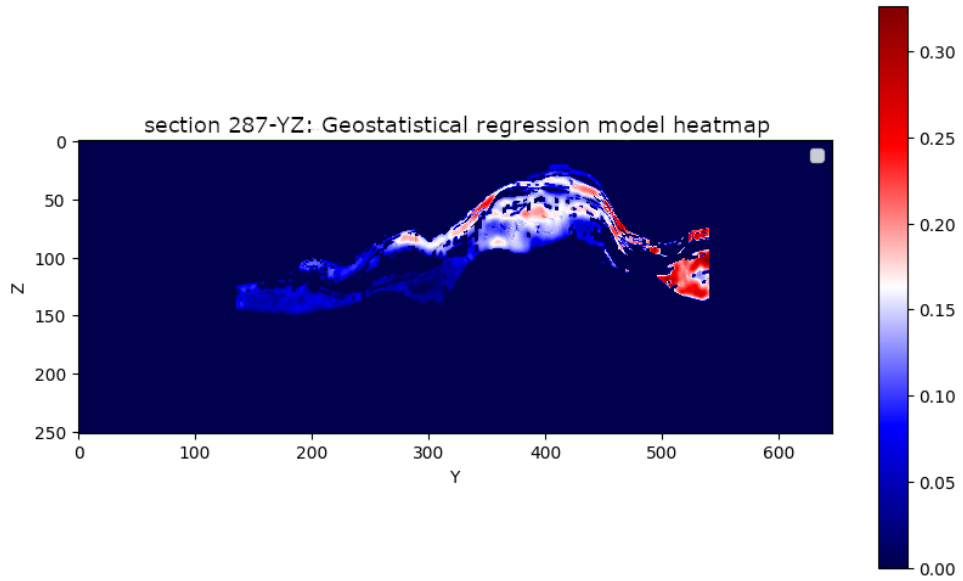
Source: The author

Figure 4.3: Porosity heatmap for section 287 YZ.



Source: The author

Figure 4.4: Geostatistical model heatmap for section 287 YZ.

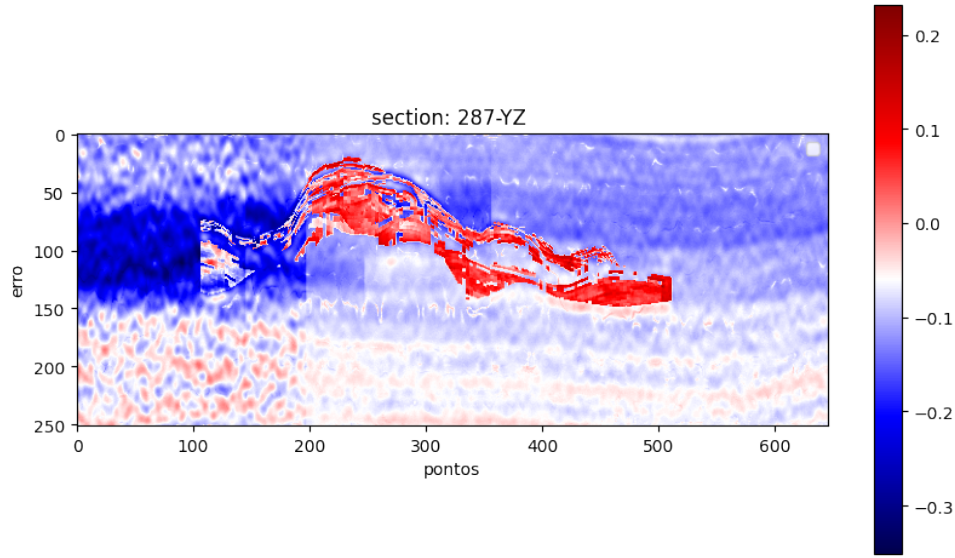


Source: The author

4.1.2 Autoencoder model

We used the Convolutional Autoencoder model with the ResNet-18 backbone and the MSE loss function. The model utilizes a modified ResNet-18 network as its encoder

Figure 4.5: Error heatmap for section 287 YZ.



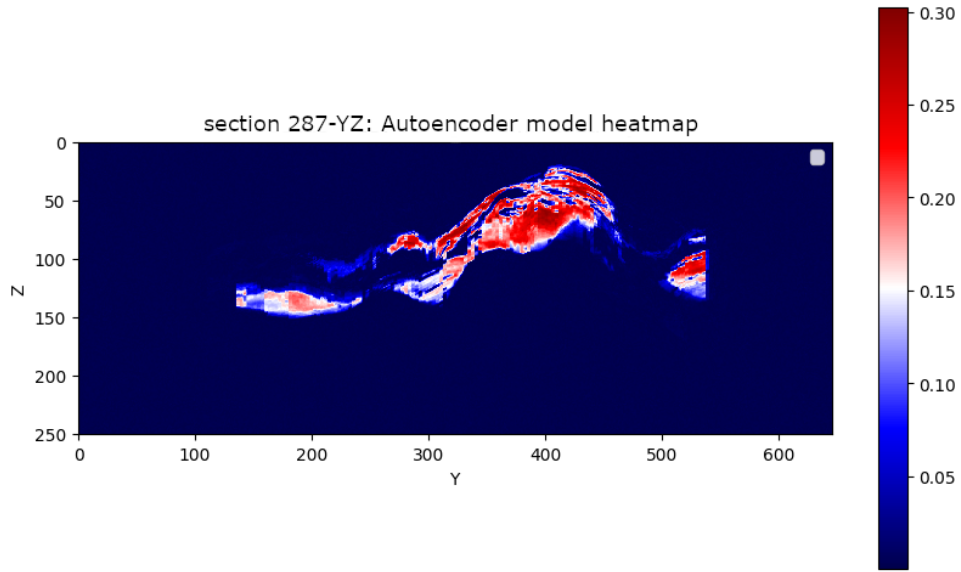
Source: The author

to handle complex image reconstruction tasks. The architecture known for compressing and then reconstructing image data, effectively learning a compact representation of the input data. The encoder, labeled as ResNet18, modifies the original ResNet-18 model by changing the first convolutional layer to accommodate an input with 3 channels and removes the final average pooling and fully connected layers. This allows it to act as a feature extractor that outputs a high-dimensional feature map. The decoder, on the other hand, is responsible for reconstructing the original input data from the compressed representation. We used the seismic data and the geostatistical model predictions as input, and the loss was also the geostatistical model predictions. The expected output was that the porosity predictions improved. We decided on that approach, considering we will not have the ground truth for the whole area in other training datasets, just the well's data. Although it worked better than the MLP approach, more was needed to reduce the bias. Hence, we decided to change the loss function.

The results can be seen in the figures 4.6, 4.7, 4.4 and 4.8. In the autoencoder error heatmap, the closer the value is to zero, the better since we are comparing it by subtracting it from the ground truth. The same is used for the geostatistical model error heatmap.

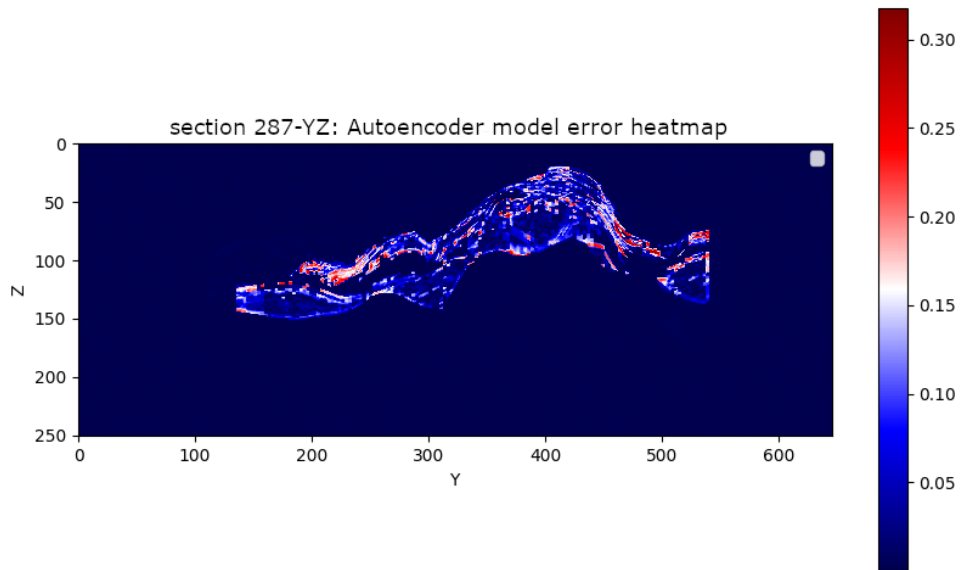
With those results, we established the Autoencoder model as the main model. Therefore we begin to refine the model.

Figure 4.6: Heatmap for section 287 YZ.



Source: The author

Figure 4.7: Error heatmap for section 287 YZ.

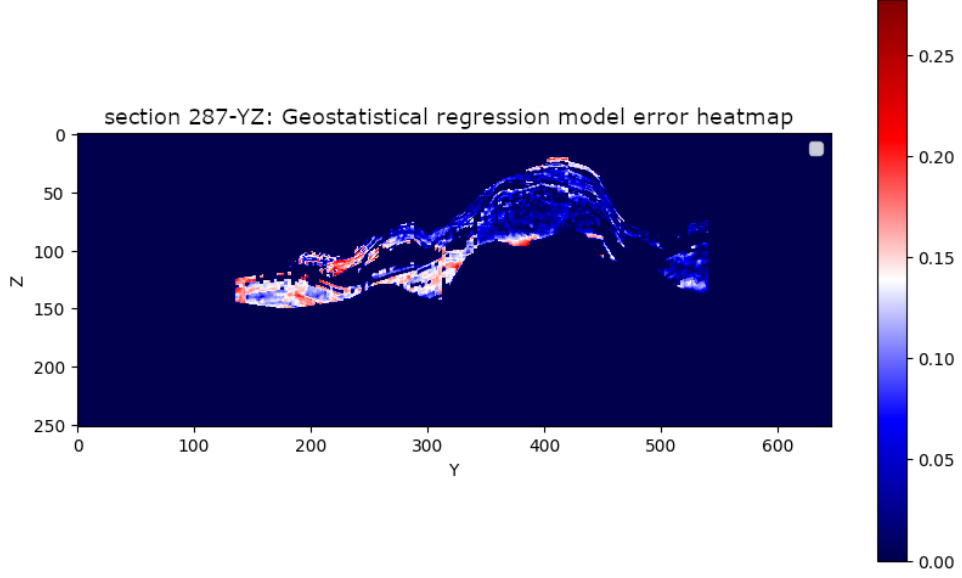


Source: The author

4.1.3 Weighted Loss

In order to further refine the AE model, we decided to change the loss function once we could not increase our data or do data augmentation. We opted for a weighted

Figure 4.8: Geostatistical model Error heatmap for section 287 YZ.



Source: The author

loss function to emphasize the error inside the turbidic channel. The function is seen in the Equation 4.1 and in the code A.2 as:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \frac{(y_i - \hat{y}_i)^2}{|y_i - \hat{y}_i| + \epsilon} \quad (4.1)$$

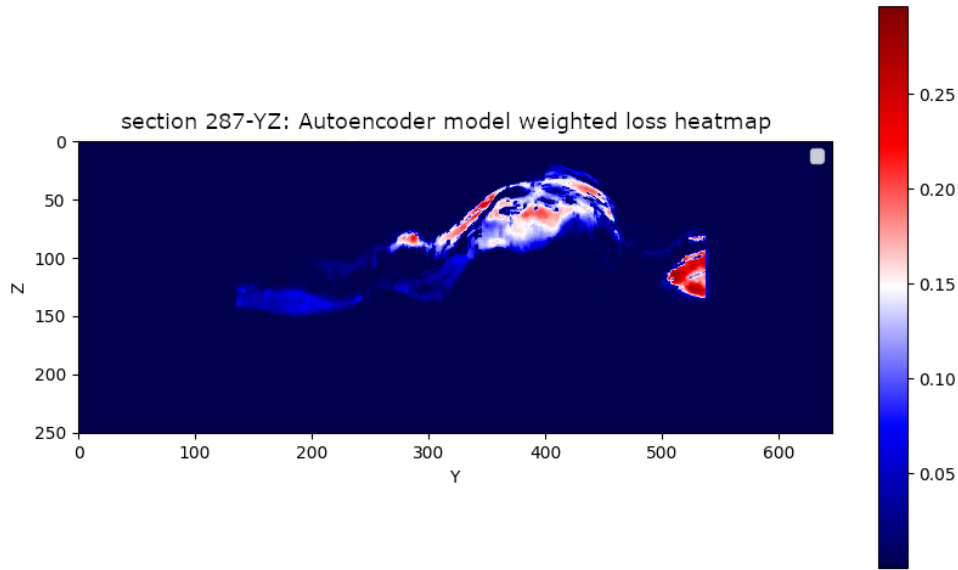
Where:

- y_i is the ground-truth value (target) of the i -th sample,
- $\hat{y}_i$ is the predicted value of the i -th sample,
- $e_i = y_i - \hat{y}_i$ is the prediction error,
- the denominator $|y_i - \hat{y}_i| + \epsilon$ defines the weight $w_i = \frac{1}{|e_i| + \epsilon}$,
- ϵ is a small positive constant added for numerical stability,
- N is the total number of samples,
- the numerator $(y_i - \hat{y}_i)^2 = e_i^2$ penalizes large errors quadratically.

The `WeightedLoss` function computes a weighted quadratic loss between predictions and actual values. Unlike conventional MSE, this loss gives larger weights to more accurate predictions (more minor errors) and smaller weights to less accurate predictions (more significant errors).

We obtain the results that can be seen in the figure 4.9. It does not show visible improvement regarding the original prediction in 4.4.

Figure 4.9: AutoEncoder heatmap for section 287 YZ with the WeightedLoss.



Source: The author

4.2 Adjusting the data

4.2.1 Linear regression

After several attempts to remove the bias ended up being fruitless, we decided to tackle the source of the bias, the petroelastic inversion data. Three seismic variables come from the inversion: VP, VS, and Rho. Those variables are related to the rock's porosity and benefit porosity predictions. Those variables are present in the wells with their ground truths. That fact prompted us to our next experiment, removing the bias from the inversion. Our initial experiment was, separately, using a linear regression model¹ to learn the residuals from the inversion data in the wells point-to-point and then sum the value for each point of the inversion data. Then, we would test that with a feed-forward MLP as seen in A.3 to predict the porosity. As we had the biased prediction values, comparing with the adjusted data was simple.

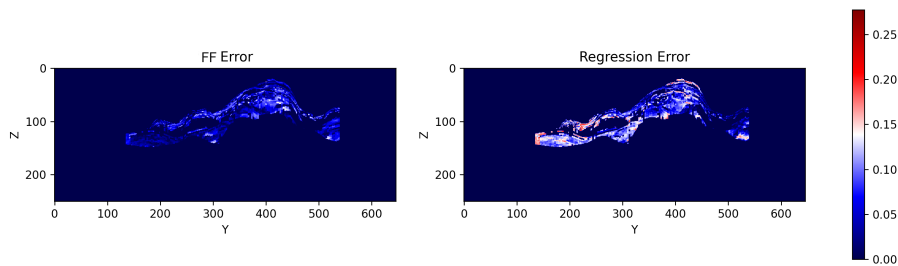
¹https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

4.2.2 MLP

As the linear regression model worked, we decided that we should test a non-linear model to check if we could improve the bias reduction. Our experiment used a simple MLP² model with the same data as the linear regression model used. An example of the parameters can be seen in A.4. The train was also point-to-point. After we adjusted the variables' values, we put them through the same feed-forward network to see the improvement in the porosity prediction.

Figures 4.10 and 4.11 show the results of the model A.3, both with the inversion data for reference and the adjusted inversion data. Figure 4.3 is the ground truth of the porosity of section 287 YZ, figure 4.10 compares the error in porosity prediction of the feed-forward model and the geostatistical model, after subtraction with the ground truth of the section, using the inversion data. Figure 4.11 has the exact representation but with the adjusted inversion data. Visually, it is not easy to notice the difference between the gains of a model trained with adjusted data and one trained with unadjusted data. However, the gains are evident numerically. The numerical results are in the table 4.1, where *GEO* stands for geostatistical model, and *FF* stands for Feed Forward model, and it is possible to see that this adjustment in the data already starts to make a difference in the values of porosity prediction. We achieved those values using only the positive values in the section once that the zeros would interfere with the actual evaluation by pulling the MAE value down.

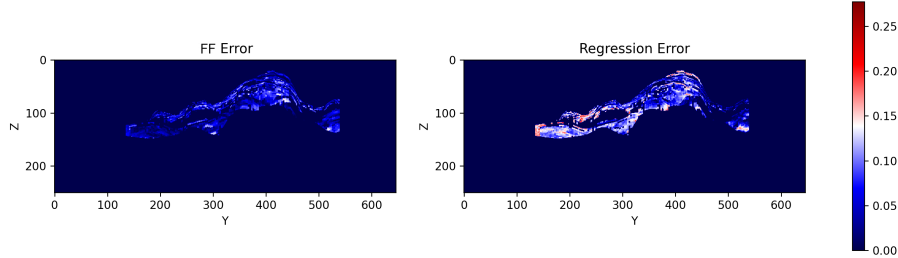
Figure 4.10: Comparison of the error relative to the ground truth of the Feed Forward model with the geostatistical model in section 287 YZ using the inversion variables



Source: The author

²https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

Figure 4.11: Comparison of the error relative to the ground truth of the Feed Forward model with the geostatistical model in section 287 YZ using the adjusted inversion variables



Source: The author

Table 4.1: Comparison of results for section 287 YZ for the geostatistical model and the Feed Forward model.

Section	MAE GEO	MAE FF
287 original	0.1093	0.0517
287 adjusted	0.1093	0.0508

Source: The author

4.3 Image-to-Image Translation

The previous situation presented a particular problem that prevented us from pursuing it. The correction only occurred where there was either the well or the ground truth, as the model needed two inputs, and the problem was that we could not use the model elsewhere as we would not have the ground truth. Otherwise, there was no need for the model itself. Therefore, we decided to try another approach, the IIT. IIT refers to transforming an input image from one domain into an output image in another while retaining the original image's essential content and structure of the original image [12]. This task can involve altering the style, enhancing certain features, or transforming the scene's context. For instance, this technique can transform a daytime photograph into a nighttime scene or convert a sketch into a photorealistic image. The transformation is driven by deep learning models, particularly GANs, which learn to capture and convert the essential characteristics from one domain to another, enabling a wide range of applications from art creation to medical image analysis.

We initially tried the model we used as baseline [37], but as it proved that it was not up to the task, we preferred to design our model. We based our idea of the IIT model on the same author, but we chose not to use GANs as we did not need to create the same

image in a different context or a different image in the same context; what we needed was just the ability to adapt one image to another context without losing the structure of either the base image or the target one.

Our model uses convolutional autoencoders, and our initial model uses the ResNet-18 as the backbone for the encoder and a simple decoder. The model is structured into three main parts: an encoder, a feature mapping layer, and a decoder. The encoder uses a modified ResNet architecture, typically ResNet-18, where the first convolutional layer is adjusted to accept a specified number of input channels, and the fully connected layers at the end are discarded. This allows the model to process input data and reduce its dimensionality while retaining critical spatial hierarchies. The feature mapping layer is a superficial convolutional layer that further maps the encoded features to a new space, adjusting the number of channels from 512 to 256 and preparing the encoded data for decoding. The decoder, implemented as a series of transposed convolutional layers with batch normalization and ReLU activations, gradually reconstructs the data to its original dimension. To manage overfitting and ensure model generalization, dropout is applied after some ReLU activations. The decoder also includes a final adjustment step using bilinear interpolation to ensure the output dimensions match specific requirements (646x251), making the model adaptable to different output sizes. This architecture is suitable for tasks where the understanding and reconstruction of input data across different domains are crucial, leveraging ResNet’s robust feature extraction capabilities and the precise reconstruction potential of transposed convolutions. This model can be seen in the [A.5](#) and produced results by transforming the seismic data into porosity data using the ground truth of porosity. However, the ResNet-18 model did not capture most details from the ground truth, so we decided to improve the model.

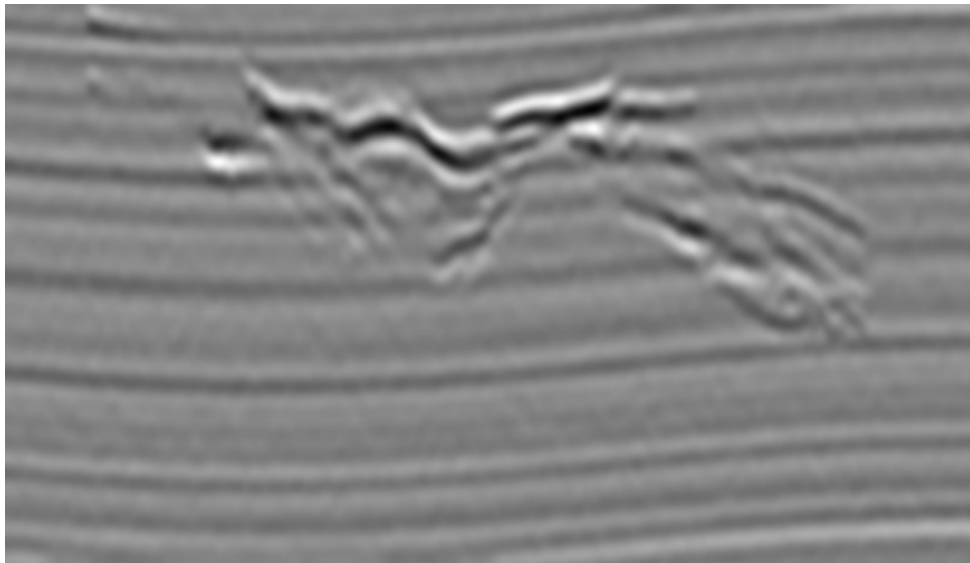
Our first attempt at enhancement involved transitioning the backbone from the ResNet-18 to the U-Net, renowned for its detail-capturing capabilities in segmentation. The heart of the model, the U-NetEncoder module, begins by progressively reducing the spatial dimensions of the input through a series of convolutional and pooling layers, increasing the channel depth as it delves deeper. This is accomplished using the DoubleConv class, which applies two consecutive convolutional operations, each followed by batch normalization and ReLU activation, capturing complex features at various levels of abstraction. Following the encoder, a FeatureMappingLayer adapts the feature maps from the encoder to a suitable form for the decoder by adjusting the number of feature channels. The CustomDecoderU-Net then reconstructs the output back to the original input dimensions through a sequence of transposed convolutions, essentially the inverse of convolutions, enlarging the feature maps while applying dropout for regularization. The output size is finely adjusted using bilinear interpolation to match specific height and width dimensions. This architecture is beneficial for tasks that require detailed feature preservation and reconstruction from compressed representations, such as image

segmentation, enhancement, or cross-modal translations where accurate spatial and feature resolution are crucial. This model is in A.6 and utilizes the same decoder, and that change alone was enough for the model capture a lot more details of the ground truth, but unfortunately the model now undervalue the real porosity values.

As a way of tackling that, we opted for the improvement of the decoder. The encoding and the feature mapping are the same as in the previous model A.6, with the difference being the decoder. The decoding process is handled by CustomDecoder-Improved, which uses a series of upsampling and convolutional layers with LeakyReLU activations to progressively reconstruct the input data to its original size, emphasizing precise detail restoration. Finally, the output is resized to specific dimensions using bi-linear interpolation, ensuring the output shape meets application-specific requirements. This architecture is beneficial in applications requiring precise and efficient reconstruction of high-dimensional data from compressed representations, such as in medical imaging or cross-modal data translation tasks. This change made the model overvalue the actual porosity values available in A.6. Now that we had a model that undervalues and one that overvalues, we opted for an ensemble between them. This ensemble does a simple average between the two predictions, which manages to capture the details and the correct values.

The figure 4.12 is the reference image for the inference of the model [37], and its result can be seen in the figure 4.13. The model could not transform the image 4.12 into 4.14, which is the ground truth for the seismic data. This model is insufficient without the ability to transform one data into the other. Therefore, another one was designed.

Figure 4.12: Reference image for the POV seismic section 200XZ



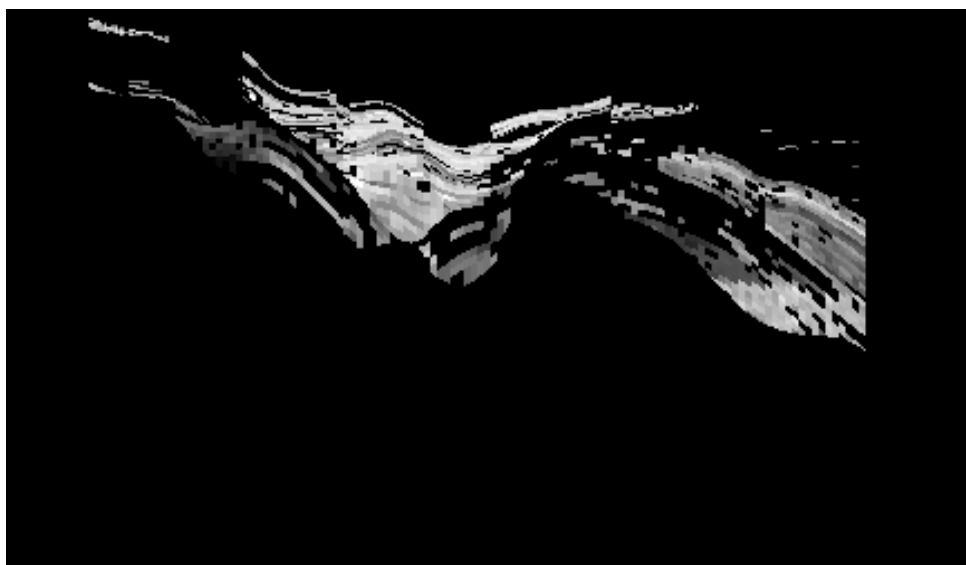
Source: The author

Figure 4.13: Result image for the seismic section 200XZ of the model created by [37].



Source: The author

Figure 4.14: Ground truth for the POV seismic section 200XZ

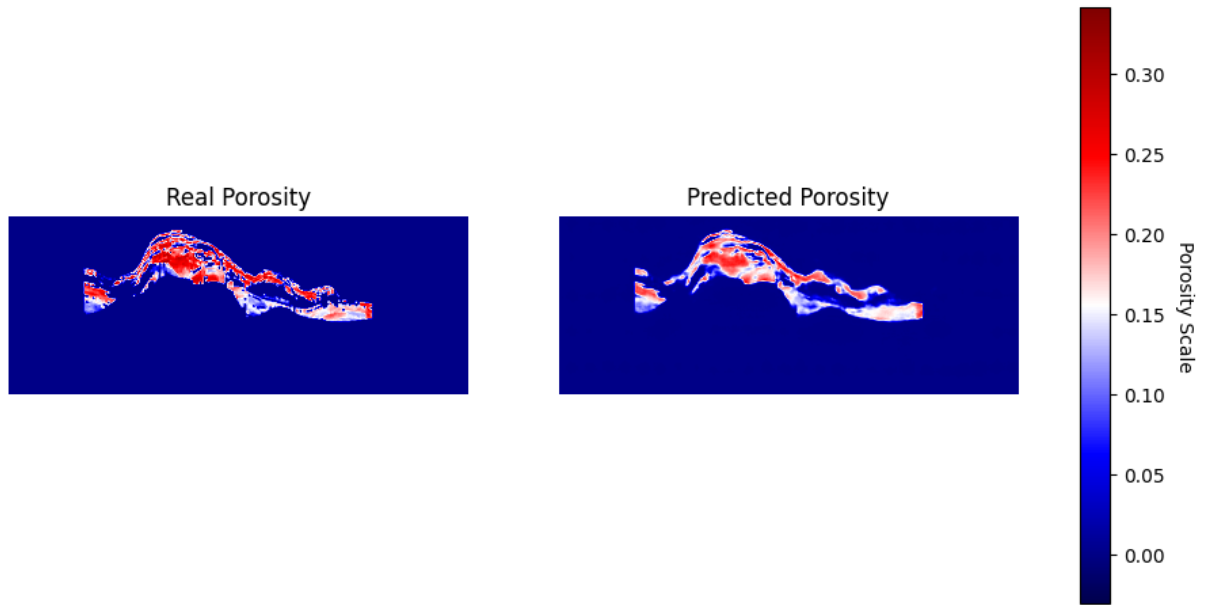


Source: The author

4.3.1 ResNet model

Figure 4.15 shows the result of the IIT model with the ResNet-18 encoder. It shows significant improvements regarding the [37] model. However, it is visible as it lacks many details that can make the difference, and also is not so precise in the values, as is shown in the color between the two images in the same scale.

Figure 4.15: Comparison between ground truth and the autoencoder prediction using the ResNet-18 encoder

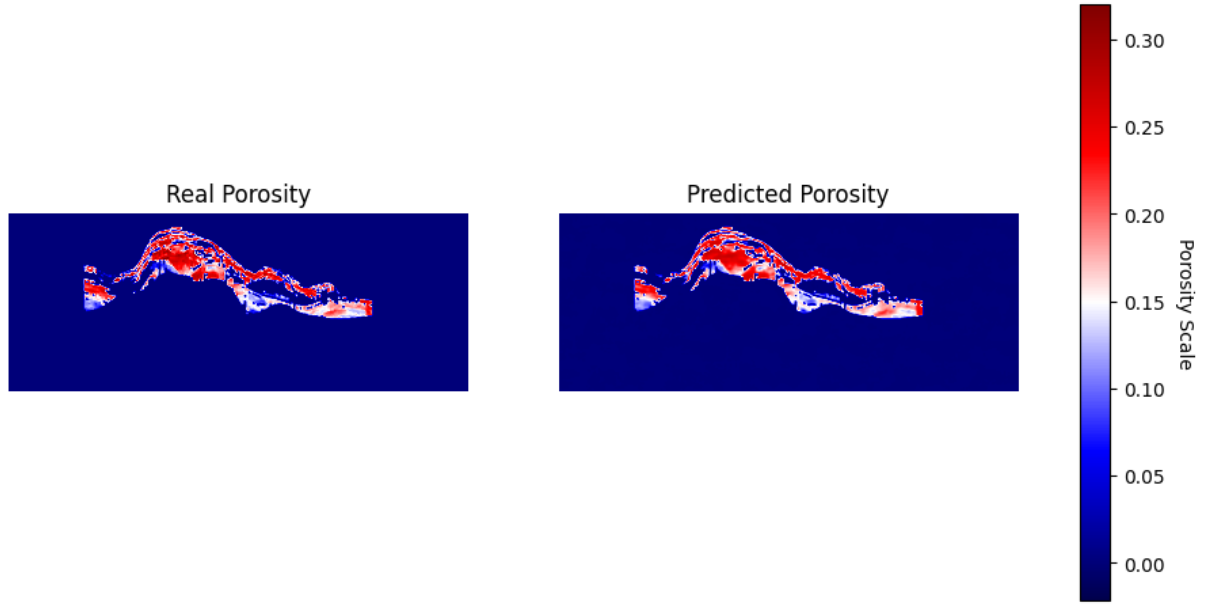


Source: The author

4.3.2 U-Net model

Figure 4.16 shows the result of the IIT model with the U-Net encoder. It shows significant improvements regarding the ResNet-18 model. However, even though it is not that perceptible when both images are in the same scale, this model undervalues the porosity values.

Figure 4.16: Comparison between ground truth and the autoencoder prediction using the U-Net encoder



Source: The author

4.3.3 U-Net improved model

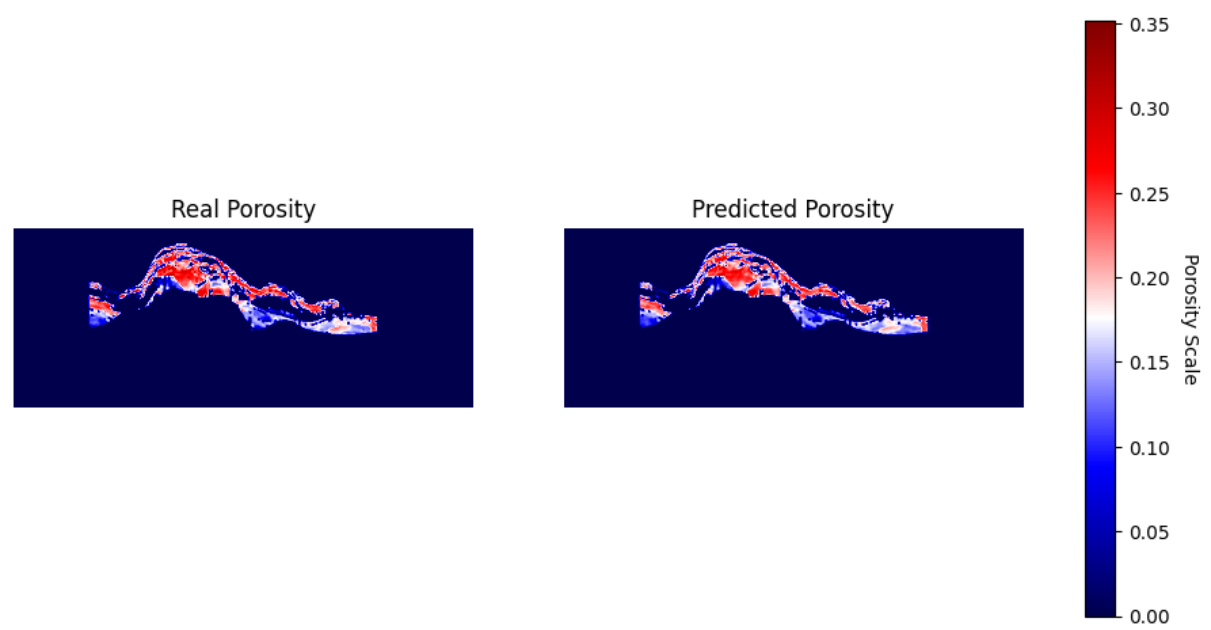
Figure 4.17 shows the result of the IIT model with the U-Net improved decoder. It does not show significant improvements regarding the U-Net model. This model overvalues the porosity values.

4.3.4 U-Net ensemble model

Figure 4.18 shows the result of the IIT model with the U-Net improved decoder. It shows very little improvements regarding the U-Net improved model. This model seems to correctly evaluate the porosity values. We also did a 5-fold with the three models under the same conditions to verify their capabilities numerically. The table 4.2 shows the results for the Mean MAE of the 5 folds. There is a small improvement from the U-Net model to the U-Net improved decoder, but a infimous difference from the U-Net improved decoder to the U-Net ensemble.

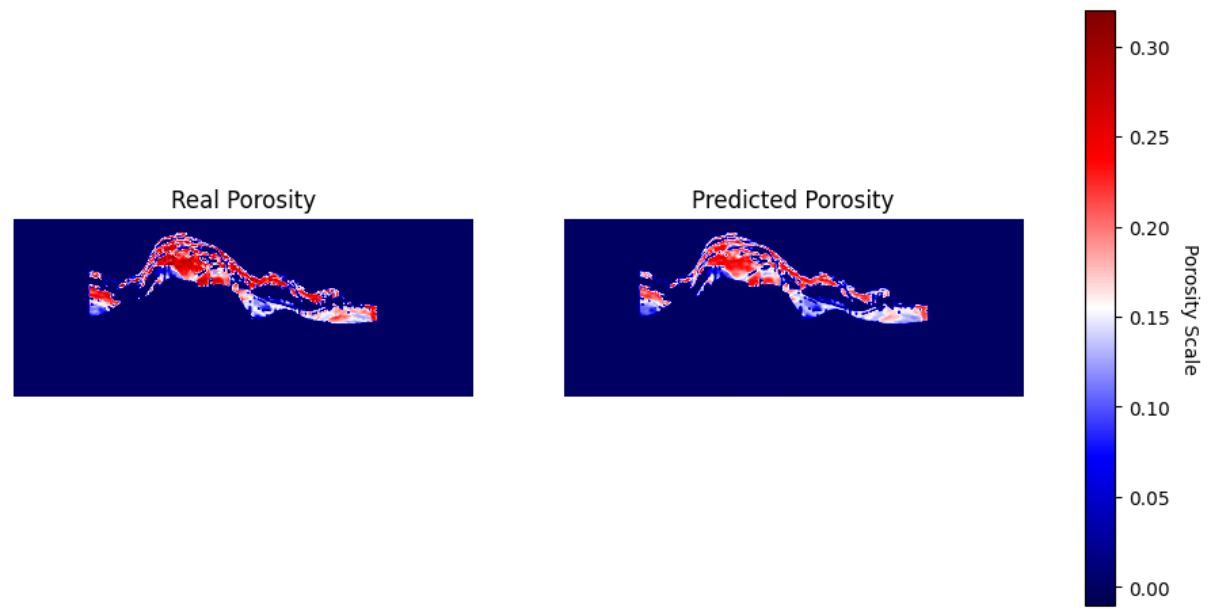
As no more datasets with ground truths were available, we could not train or test the model and could no longer perform with it. The model generalization will be done in

Figure 4.17: Comparison between ground truth and the autoencoder prediction using the U-Net improved decoder



Source: The author

Figure 4.18: Comparison between ground truth and the autoencoder prediction using the U-Net ensemble



Source: The author

a future work due to time and resources limitations currently.

Model	Mean MAE
U-Net	0.0487
U-Net improved	0.0448
U-Net ensemble	0.0440

Table 4.2: Comparison of results for K-fold Mean MAE error across all folds.

4.4 Discussions

[37] proposed cross-domain autoencoders capable of preserving content while translating between image domains, achieving high-quality results in natural image datasets. In our work, the IIT approach with U-Net architecture achieved a mean absolute error (MAE) of 0.0487, which improved to 0.0448 with an enhanced decoder and 0.0440 with an ensemble model. While absolute metrics are not directly comparable, the improvements within our own baselines demonstrate that the proposed IIT and ensemble strategies outperformed the simpler models tested in this work.

[26] presents a comprehensive systematic literature review, summarizing state-of-the-art deep learning methods for seismic image segmentation, especially for seismic facies delineation. It emphasizes comparative synthesis, identifying gaps such as the lack of standardized metrics and the scarcity of labeled datasets. This work intersects on the topic of deep learning for seismic interpretation but diverge in their technical execution: [26] catalogs deep learning architectures used in seismic segmentation, such as U-Net, DeepLabV3+, ResNet, and DAnet-FCN. It evaluates trends in supervised, semi-supervised, and self-supervised paradigms and identifies open challenges regarding reproducibility and benchmarking. This work develops and compares models including U-Net, FPN, and Autoencoders. It explores uncertainty estimation, transfer learning, and ensemble methods to enhance model robustness. This work also integrates porosity prediction a task absent from [26] work using IIT and regression-based models. This work presents detailed quantitative metrics for training, validation, and testing, while [26] paper is limited to qualitative and comparative reporting of literature findings. Even though [26] does a extensive sismofacies classification review, it does not analyze the pre-salt region where the majority of this work has been done. This fact shows that there are still a lot of room for improvement and research when we are talking about more complex regions for seismostratigraphy, and this is another advantage for this work, as we explore those complex regions and try to find solutions for it's challenges.

[14] also presents a systematic literature review, but focusing on the effectiveness of those ML state-of-the-art applications rather than in their utilities for the task in hand. This focus has been selected as a way of showing the path trail-blazed by those techniques so far in the seismic exploration field, a ways to improve it. This work intersects with

this paper on the seismostratigraphy task, as [14] shows how it has been done in the literature, using 2D/3D seismic, well-logs and derivative seismic attributes; with techniques such as CNN (U-Net), Self-organizing Maps (SOM), K-Means and Generative Adversarial Networks (GANS); training with both synthetic and real data; these techniques have been yielding good results as they can predict the seismofacies and petrophysics properties with good accuracy, but not forgetting the limitations that those techniques might come with such as the lack of labels or the low generalization between areas. As we were developing this work, we came across the same results and limitations, as we could achieve good metrics during training but bad predictions in areas far away from it. We could also suffer from the lack of labels for the training itself, and this was a big issue to the point where we had to develop a new strategy to circumvent this particular limitation with the semi-supervised method with the edge detection. This low generalization was also observable in the SFM, that could not be directly apply with our dataset, and even when trained with our dataset, still yield bad segmentations. But like [26], [14] does not cover the pre-salt region, which, once again, shows the relevance of this work as we cover it, showing where we can take advantage of ML techniques and also where ML will fall short for the time being.

These results also validate the methodological choices for the IIT-based porosity mapping under the same conditions of data-scarce and geologically complex environments.

Chapter 5

Conclusion

This dissertation presented the development and evaluation of machine learning methodologies applied to reservoir characterization, focusing on seismography segmentation and porosity estimation. The study was motivated by the persistent challenges associated with the scarcity of labeled data and the geological complexity of subsurface environments, particularly in pre-salt regions. Despite advances in seismic acquisition and processing, the interpretation of subsurface structures remains constrained by limited ground-truth information and interpreter subjectivity.

Regarding the seismostratigraphy task, convolutional architectures based on U-Net and FPN were evaluated under supervised, semi-supervised, and transfer learning schemes. Tests showed efficient training with as little as 12.5% of labeled slices with synthetic data. Additional tests also demonstrated that using masks derived from synthetic porosity models effectively reduced interpreter bias. The experiments conducted on the Marimbá Field and the Búzios Region datasets showed that the representativeness and quality of labeled samples exerted greater influence on accuracy than the absolute quantity of labeled data. The integration of edge detection techniques proved essential for filtering inconsistent labels, improving input data quality and model generalization.

In the porosity estimation task, the study approached the problem as a spatial regression challenge, implementing models ranging from simple MLPs to autoencoders with ResNet and U-Net backbones, as well as improved decoders and IIT strategies. The analyses revealed that convolutional models with optimized decoders and ensemble techniques yielded improved spatial coherence and predictive accuracy. Nevertheless, these gains remained moderate, emphasizing the inherent difficulty of estimating complex petrophysical properties from two-dimensional seismic inputs.

The experiments also underscored the importance of auxiliary strategies such as bias correction, data augmentation, and transfer learning to mitigate the scarcity of well-log-data still one of the main bottlenecks for model generalization, even though they all failed. Porosity predictions exhibited systematic variations in geologically heterogeneous zones, indicating the need for more robust and multimodal approaches.

Overall, the results confirmed that Deep Learning are promising tools for seismic interpretation and reservoir property prediction, offering enhanced efficiency and consis-

tency in geophysical analysis. However, the findings also highlighted that the representativeness of training data and the integration of diverse information sources remain crucial for improving model reliability and scalability.

Future work should consider the integration of additional seismic attributes, alternative data modalities (such as inversion volumes or geological models), and more advanced deep learning frameworks, including attention-based models and visual transformer architectures. Additionally, expanding the availability of labeled datasets, potentially through collaborative industry efforts or the development of synthetic training datasets, will be crucial for enhancing model performance and generalizability.

References

- [1] M. Abdideh. Estimation of permeability using artificial neural networks and regression analysis in an iran oil field. *International Journal of the Physical Sciences*, 7, 2012.
- [2] James Abe and Damilola Adekanye. Advancing reservoir characterization: A comparative analysis of xg boost and ann for accurate porosity prediction. *Journal of Data Analytics*, 3(1):47–60, Dec. 2024.
- [3] Fatai Anifowose, Jane Labadin, and Abdulazeez Abdulraheem. Predicting petroleum reservoir properties from downhole sensor data using an ensemble model of neural networks. In *Proceedings of Workshop on Machine Learning for Sensory Data Analysis*, MLSDA '13, page 27–34, New York, NY, USA, 2013. Association for Computing Machinery.
- [4] Sami Barchid, José Mennesson, and Chaabane Djeraba. Review on indoor RGB-D semantic segmentation with deep convolutional neural networks. *CoRR*, abs/2105.11925, 2021.
- [5] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12472–12482, 2020.
- [6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.
- [7] Jiaxing Du. Road crack detection based on improved U-Net. In Linlin Shen and Guoqiang Zhong, editors, *Third International Conference on Computer Vision and Pattern Analysis (ICCPA 2023)*, volume 12754, page 127541J. International Society for Optics and Photonics, SPIE, 2023.
- [8] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine*

- Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.
- [9] R. Gholami, A. Shahraki, and M. Paghaleh. prediction of hydrocarbon reservoirs permeability using support vector machine. *Mathematical Problems in Engineering*, 2012:1–18, 2012.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [11] T. Inan and U. Kacar. Ear semantic segmentation in natural images with tversky loss function supported deeplabv3+ convolutional neural network. *Balkan Journal of Electrical and Computer Engineering*, 10:337–346, 2022.
- [12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.
- [13] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding, 2016.
- [14] Farbod Khosro Anjom, Francesco Vaccarino, and Laura Valentina Socco. Machine learning for seismic exploration: Where are we and how far are we from the holy grail? *Geophysics*, 89(1):WA157–WA178, January 2024.
- [15] Q. Li and S. Dehler. Seismic data quality control and interpolation using principal component analysis. *International Journal of Geosciences*, 10:950–966, 2019.
- [16] X. Li, H. Chen, X. Qi, Q. Dou, C. Fu, and P. Heng. H-denseunet: hybrid densely connected unet for liver and tumor segmentation from ct volumes. *Ieee Transactions on Medical Imaging*, 37:2663–2674, 2018.
- [17] S. Lim, Y. Kim, Y. Park, D. Kim, K. Kim, and D. Lee. Automated pancreas segmentation and volumetry using deep neural network on computed tomography. *Scientific Reports*, 12, 2022.
- [18] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
- [19] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

- [20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.
- [21] M. Maleki, M. Cirne, D. Schiozer, A. Davolio, and A. Rocha. A machine-learning framework to estimate saturation changes from 4d seismic data using reservoir models. *Geophysical Prospecting*, 70:1388–1409, 2022.
- [22] T. Martin, R. Meyer, and Z. Jobe. Centimeter-scale lithology and facies prediction in cored wells using machine learning. *Frontiers in Earth Science*, 9, 2021.
- [23] T. F. Mein, A. L. V. Gimenes, E. M. Dias, M. L. R. P. D. Scoton, and M. E. M. Udaeta. Environmental vulnerability in pre-salt oil and gas operations. *Energies*, 14:732, 2021.
- [24] Lucas Azevedo Birro Michelin. Fast mixture spatial regression: A mixture in the geographical and feature space applied to predict oil in the post-salt. Dissertação (mestrado em estatística), Universidade Federal de Minas Gerais, Belo Horizonte, 2023. Versão final.
- [25] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: a survey. *Ieee Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.
- [26] Bruno A.A. Monteiro, Gabriel L. Canguçu, Leonardo M.S. Jorge, Rafael H. Vareto, Bryan S. Oliveira, Thales H. Silva, Luiz Alberto Lima, Alexei M.C. Machado, William Robson Schwartz, and Pedro O.S. Vaz de Melo. Literature review on deep learning for the segmentation of seismic images. *Earth-Science Reviews*, 258:104955, 2024.
- [27] T. Moussa, S. Elkatatny, M. Mahmoud, and A. Abdulraheem. Development of new permeability formulation from well log data using artificial intelligence approaches. *Journal of Energy Resources Technology*, 140, 2018.
- [28] Cassiane Maria Ferreira Nunes. *Impacto da Utilização de ‘Pseudo-Wells’ na Interpretação Sísmica: Enfoque 4D*. Tese de doutorado (geologia), Universidade Federal do Rio de Janeiro, jun 2008.
- [29] G. Roberts, S. Haile, R. Sainju, B. Hutchinson, and Y. Zhu. Deep learning for semantic segmentation of defects in advanced stem images of steels. *Scientific Reports*, 9, 2019.
- [30] G. Roberts, R. Sainju, B. Hutchinson, M. Toloczko, and Y. Zhu. Defectnet – a deep convolutional neural network for semantic segmentation of crystallographic defects in advanced microscopy images. *Microscopy and Microanalysis*, 25:164–165, 2019.

- [31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.
- [32] Hanlin Sheng, Xinming Wu, Xu Si, Jintao Li, Sibozhang, and Xudong Duan. Seismic foundation model (sfm): a new generation deep learning model in geophysics, 2023.
- [33] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 270–279, 2018.
- [34] Haoru Tan, Sitong Wu, and Jimin Pi. Semantic diffusion network for semantic segmentation, 2023.
- [35] S. Tran, C. Cheng, T. Nguyen, M. Le, and D. Liu. Tmd-unet: triple-unet with multi-scale input features and dense skip connection for medical image segmentation. *Healthcare*, 9:54, 2021.
- [36] J. Wang, Y. Luo, Z. Yang, X. Zhao, and Z. Niu. Research on prediction of movable fluid percentage in unconventional reservoir based on deep learning. *Applied Sciences*, 11:3589, 2021.
- [37] Jaechang Yoo, Heesong Eom, and Yong Suk Choi. Image-to-image translation using a cross-domain auto-encoder and decoder. *Applied Sciences*, 9(22), 2019.
- [38] D. Zhang, Z. Yang, S. Jiang, Z. Zhou, M. Meng, and W. Wang. Automatic segmentation and applicator reconstruction for ct-based brachytherapy of cervical cancer using 3d convolutional neural networks. *Journal of Applied Clinical Medical Physics*, 21:158–169, 2020.
- [39] Li Zhang, Yibei Chen, Yiyong Lin, Bingxi Dong, and Xinglong Ran. Research on semantic segmentation algorithm of multispectral remote sensing image based on deep learning. In Deqiang Cheng and Omer Deperlioglu, editors, *2nd International Conference on Signal Image Processing and Communication (ICSIPC 2022)*, volume 12246, page 1224615. International Society for Optics and Photonics, SPIE, 2022.
- [40] L. Zhao, S. Wu, and S. Xiao. Predicting the surveillance data in a low-permeability carbonate reservoir with the machine-learning tree boosting method and the time-segmented feature extraction. *Energies*, 13:6307, 2020.
- [41] Z. Zhou, M. R. Siddiquee, N. Tajbakhsh, and J. Liang. Unet++: redesigning skip connections to exploit multiscale features in image segmentation. *IEEE Transactions on Medical Imaging*, 39:1856–1867, 2020.

-
- [42] G. Zhu, X. Luo, T. Yang, L. Cai, J. Yeo, Y. Ge, and J. Yang. Deep learning-based recognition and segmentation of intracranial aneurysms under small sample size. *Frontiers in Physiology*, 13, 2022.
 - [43] X. Zhuo, F. Fraundorfer, F. Kurz, and P. Reinartz. optimization of openstreetmap building footprints based on semantic information of oblique uav images. *Remote Sensing*, 10:624, 2018.
 - [44] P. Łuszczek, W. M. Sid-Lakhdar, and J. Dongarra. Combining multitask and transfer learning with deep gaussian processes for autotuning-based performance engineering. *The International Journal of High Performance Computing Applications*, 37:229–244, 2023.

Appendix A

Apêndice

A.1 Convolutional Autoencoder

```

1
2 def get_encoder_output_size(encoder, input_shape):
3     dummy_input = torch.randn(1, *input_shape)
4     dummy_output = encoder(dummy_input)
5     return dummy_output.numel()
6
7 class ResNet18(nn.Module):
8     def __init__(self, in_channels):
9         super(ResNet18, self).__init__()
10        resnet_model = resnet.resnet18(pretrained=True)
11        resnet_model.conv1 = nn.Conv2d(in_channels, 64,
kernel_size=7, stride=2, padding=3, bias=False)
12        self.resnet_layers = list(resnet_model.children())
[: -2]
13        self.model = nn.Sequential(*self.resnet_layers)
14
15    def forward(self, x):
16        return self.model(x)
17
18 class ConvAutoencoderResnet18(nn.Module):
19    def __init__(self, input_shape):
20        super(ConvAutoencoderResnet18, self).__init__()
21
22
23        self.encoder = ResNet18(in_channels=3)
24

```

```

25
26         self.encoder_output_size = get_encoder_output_size(
self.encoder, input_shape)
27
28         self.fc1 = nn.Linear(self.encoder_output_size, 512)
29
30
31         self.decoder = nn.Sequential(
32             nn.ConvTranspose2d(512, 256, 2, stride=2),
33             nn.ReLU(inplace=True),
34             nn.ConvTranspose2d(256, 128, 2, stride=2),
35             nn.ReLU(inplace=True),
36             nn.ConvTranspose2d(128, 64, 2, stride=2),
37             nn.ReLU(inplace=True),
38             nn.ConvTranspose2d(64, 3, 2, stride=2),
39         )
40
41
42         self.fc2 = nn.Linear(3 * 16 * 16, 162146)
43
44     def forward(self, x):
45         x = self.encoder(x)
46         x = x.view(x.size(0), -1)
47         x = self.fc1(x)
48
49         x = x.view(x.size(0), 512, 1, 1)
50         x = self.decoder(x)
51         x = x.view(x.size(0), -1)
52         x = self.fc2(x)
53
54     return x

```

A.2 Weighted Loss

```

1 def WeightedLoss(pred, target):

```

```
2     epsilon=1e-6
3     assert pred.size() == target.size()
4     error = target - pred
5     weights = 1 / (torch.abs(error) + epsilon)
6     loss = weights * (error ** 2)
7     return loss.mean()
```

A.3 MLP Regression Model

```
1
2 class MLPRegression(nn.Module):
3     def __init__(self, input_dim, hidden_dim, output_dim):
4         super(MLPRegression, self).__init__()
5         self.fc1 = nn.Linear(input_dim, hidden_dim)
6         self.fc2 = nn.Linear(hidden_dim, output_dim)
7         self.activation = nn.ReLU()
8
9     def forward(self, x):
10        x = self.activation(self.fc1(x))
11        x = self.fc2(x)
12        return x
```

A.4 MLP parameters

```
1     mlp = MLPRegressor(hidden_layer_sizes=(128,64), max_iter
=10000, random_state=21)
```

A.5 Image-to-Image Translation Resnet

```

1
2 class FeatureMappingLayer(nn.Module):
3     def __init__(self, input_dim, output_dim):
4         super(FeatureMappingLayer, self).__init__()
5         self.conv = nn.Conv2d(input_dim, output_dim,
6                                 kernel_size=3, stride=1, padding=1)
7
8     def forward(self, x):
9         return self.conv(x)
10
11 class ResNetEncoder(nn.Module):
12     def __init__(self, input_channels=1, pretrained=True):
13         super(ResNetEncoder, self).__init__()
14
15         resnet = resnet18(pretrained=pretrained)
16
17         self.resnet = nn.Sequential(
18             nn.Conv2d(input_channels, 64, kernel_size=7,
19                       stride=2, padding=3, bias=False),
20             *list(resnet.children())[1:-2]
21         )
22
23     def forward(self, x):
24         x = self.resnet(x)
25         return x
26
27 class CustomDecoder(nn.Module):
28     def __init__(self):
29         super(CustomDecoder, self).__init__()
30
31         self.decoder = nn.Sequential(
32             nn.ConvTranspose2d(256, 256, kernel_size=4,
33                               stride=2, padding=1),
34             nn.BatchNorm2d(256),
35             nn.ReLU(True),

```

```

33         nn.Dropout(0.4),
34         nn.ConvTranspose2d(256, 128, kernel_size=4,
stride=2, padding=1),
35         nn.BatchNorm2d(128),
36         nn.ReLU(True),
37         nn.Dropout(0.4),
38         nn.ConvTranspose2d(128, 64, kernel_size=4, stride
=2, padding=1),
39         nn.BatchNorm2d(64),
40         nn.ReLU(True),
41         nn.Dropout(0.4),
42         nn.ConvTranspose2d(64, 32, kernel_size=4, stride
=2, padding=1),
43         nn.ReLU(True),
44         nn.ConvTranspose2d(32, 16, kernel_size=4, stride
=2, padding=1),
45         nn.ReLU(True),
46         nn.ConvTranspose2d(16, 1, kernel_size=4, stride
=2, padding=1),
47         nn.Identity(),
48     )
49
50     def forward(self, x):
51         return self.decoder(x)
52
53 class CustomDecoderAdjusted(nn.Module):
54     def __init__(self, output_height=646, output_width=251):
55         super(CustomDecoderAdjusted, self).__init__()
56         self.output_height = output_height
57         self.output_width = output_width
58         self.decoder = CustomDecoder()
59
60     def forward(self, x):
61         x = self.decoder(x)
62
63         x = F.interpolate(x, size=(self.output_height, self.
output_width), mode='bilinear', align_corners=False)
64         return x
65

```

```

66 class CustomCrossDomainAEAdjusted(nn.Module):
67     def __init__(self):
68         super(CustomCrossDomainAEAdjusted, self).__init__()
69         self.encoder = ResNetEncoder(input_channels=1,
pretrained=True)
70         self.feature_mapping = FeatureMappingLayer(512, 256)
71         self.decoder = CustomDecoderAdjusted(output_height
=646, output_width=251)
72
73     def forward(self, x):
74         encoded = self.encoder(x)
75         mapped = self.feature_mapping(encoded)
76         decoded = self.decoder(mapped)
77         return decoded

```

A.6 Image-to-Image Translation Unet

```

1
2 class DoubleConv(nn.Module):
3
4     def __init__(self, in_channels, out_channels,
mid_channels=None):
5         super().__init__()
6         if not mid_channels:
7             mid_channels = out_channels
8             self.double_conv = nn.Sequential(
9                 nn.Conv2d(in_channels, mid_channels, kernel_size
=3, padding=1),
10                 nn.BatchNorm2d(mid_channels),
11                 nn.ReLU(inplace=True),
12                 nn.Conv2d(mid_channels, out_channels, kernel_size
=3, padding=1),
13                 nn.BatchNorm2d(out_channels),
14                 nn.ReLU(inplace=True)
15             )

```

```

16
17     def forward(self, x):
18         return self.double_conv(x)
19
20 class UnetEncoder(nn.Module):
21     def __init__(self, input_channels=1, num_filters=64):
22         super(UnetEncoder, self).__init__()
23         self.input_channels = input_channels
24         self.num_filters = num_filters
25
26         self.inc = DoubleConv(input_channels, num_filters)
27         self.down1 = nn.Sequential(
28             nn.MaxPool2d(2),
29             DoubleConv(num_filters, num_filters * 2)
30         )
31         self.down2 = nn.Sequential(
32             nn.MaxPool2d(2),
33             DoubleConv(num_filters * 2, num_filters * 4)
34         )
35         self.down3 = nn.Sequential(
36             nn.MaxPool2d(2),
37             DoubleConv(num_filters * 4, num_filters * 8)
38         )
39
40
41     def forward(self, x):
42         x1 = self.inc(x)
43         x2 = self.down1(x1)
44         x3 = self.down2(x2)
45         x4 = self.down3(x3)
46         return x4
47
48 class FeatureMappingLayer(nn.Module):
49     def __init__(self, input_dim, output_dim):
50         super(FeatureMappingLayer, self).__init__()
51         self.conv = nn.Conv2d(input_dim, output_dim,
52                                kernel_size=3, stride=1, padding=1)
53
54     def forward(self, x):

```

```

54         return self.conv(x)
55
56 class CustomDecoderUnet(nn.Module):
57     def __init__(self):
58         super(CustomDecoderUnet, self).__init__()
59         self.decoder = nn.Sequential(
60             nn.ConvTranspose2d(256, 256, kernel_size=4,
61 stride=2, padding=1),
62             nn.BatchNorm2d(256),
63             nn.ReLU(True),
64             nn.Dropout(0.4),
65             nn.ConvTranspose2d(256, 128, kernel_size=4,
66 stride=2, padding=1),
67             nn.BatchNorm2d(128),
68             nn.ReLU(True),
69             nn.Dropout(0.4),
70             nn.ConvTranspose2d(128, 64, kernel_size=4, stride
71 =2, padding=1),
72             nn.BatchNorm2d(64),
73             nn.ReLU(True),
74             nn.Dropout(0.4),
75             nn.ConvTranspose2d(64, 32, kernel_size=4, stride
76 =2, padding=1),
77             nn.ReLU(True),
78             nn.ConvTranspose2d(32, 16, kernel_size=4, stride
79 =2, padding=1),
80             nn.ReLU(True),
81             nn.ConvTranspose2d(16, 1, kernel_size=4, stride
82 =2, padding=1),
83             nn.Identity(),
84         )
85
86     def forward(self, x):
87         return self.decoder(x)
88
89 class CustomDecoderAdjustedUnet(nn.Module):
90     def __init__(self, output_height=646, output_width=251):
91         super(CustomDecoderAdjustedUnet, self).__init__()
92         self.output_height = output_height

```

```

87         self.output_width = output_width
88         self.decoder = CustomDecoderUnet()
89
90     def forward(self, x):
91         x = self.decoder(x)
92         x = F.interpolate(x, size=(self.output_height, self.
93         output_width), mode='bilinear', align_corners=False)
94         return x
95
96 class CustomCrossDomainAEAdjustedUnet(nn.Module):
97     def __init__(self):
98         super(CustomCrossDomainAEAdjustedUnet, self).__init__()
99
100         self.encoder = UnetEncoder(input_channels=1,
101         num_filters=64)
102         self.feature_mapping = FeatureMappingLayer(512, 256)
103         self.decoder = CustomDecoderAdjustedUnet(
104         output_height=646, output_width=251)
105
106     def forward(self, x):
107         encoded = self.encoder(x)
108         mapped = self.feature_mapping(encoded)
109         decoded = self.decoder(mapped)
110         return decoded

```

A.7 Image-to-Image Translation Unet improved decoder

```

1
2 class DoubleConvImproved(nn.Module):
3
4     def __init__(self, in_channels, out_channels,
5     mid_channels=None):
6         super().__init__()
7         if not mid_channels:

```

```

7         mid_channels = out_channels
8         self.double_conv = nn.Sequential(
9             nn.Conv2d(in_channels, mid_channels, kernel_size
10            =3, padding=1),
11             nn.BatchNorm2d(mid_channels),
12             nn.ReLU(inplace=True),
13             nn.Conv2d(mid_channels, out_channels, kernel_size
14            =3, padding=1),
15             nn.BatchNorm2d(out_channels),
16             nn.ReLU(inplace=True)
17         )
18
19     def forward(self, x):
20         return self.double_conv(x)
21
22 class UnetEncoderImproved(nn.Module):
23     def __init__(self, input_channels=1, num_filters=64):
24         super(UnetEncoderImproved, self).__init__()
25         self.input_channels = input_channels
26         self.num_filters = num_filters
27
28         self.inc = DoubleConvImproved(input_channels,
29            num_filters)
30         self.down1 = nn.Sequential(
31             nn.MaxPool2d(2),
32             DoubleConvImproved(num_filters, num_filters * 2)
33         )
34         self.down2 = nn.Sequential(
35             nn.MaxPool2d(2),
36             DoubleConvImproved(num_filters * 2, num_filters *
37            4)
38         )
39         self.down3 = nn.Sequential(
40             nn.MaxPool2d(2),
41             DoubleConvImproved(num_filters * 4, num_filters *

```

```

41     def forward(self, x):
42         x1 = self.inc(x)
43         x2 = self.down1(x1)
44         x3 = self.down2(x2)
45         x4 = self.down3(x3)
46         return x4
47
48 class FeatureMappingLayer(nn.Module):
49     def __init__(self, input_dim, output_dim):
50         super(FeatureMappingLayer, self).__init__()
51         self.conv = nn.Conv2d(input_dim, output_dim,
52                                kernel_size=3, stride=1, padding=1)
53
54     def forward(self, x):
55         return self.conv(x)
56
57 class CustomDecoderImproved(nn.Module):
58     def __init__(self):
59         super(CustomDecoderImproved, self).__init__()
60         self.up1 = nn.Sequential(
61             nn.Upsample(scale_factor=2, mode='nearest'),
62             nn.Conv2d(256, 128, 3, padding=1),
63             nn.BatchNorm2d(128),
64             nn.LeakyReLU(0.1, inplace=True)
65         )
66         self.up2 = nn.Sequential(
67             nn.Upsample(scale_factor=2, mode='nearest'),
68             nn.Conv2d(128, 64, 3, padding=1),
69             nn.BatchNorm2d(64),
70             nn.LeakyReLU(0.1, inplace=True)
71         )
72         self.up3 = nn.Sequential(
73             nn.Upsample(scale_factor=2, mode='nearest'),
74             nn.Conv2d(64, 32, 3, padding=1),
75             nn.BatchNorm2d(32),
76             nn.LeakyReLU(0.1, inplace=True)
77         )
78         self.up4 = nn.Sequential(
79             nn.Upsample(scale_factor=2, mode='nearest'),

```

```

79         nn.Conv2d(32, 16, 3, padding=1),
80         nn.BatchNorm2d(16),
81         nn.LeakyReLU(0.1, inplace=True)
82     )
83     self.final = nn.Sequential(
84         nn.Conv2d(16, 1, 3, padding=1),
85         nn.Identity()
86     )
87
88     def forward(self, x):
89         x = self.up1(x)
90         x = self.up2(x)
91         x = self.up3(x)
92         x = self.up4(x)
93         x = self.final(x)
94         return x
95
96 class CustomDecoderAdjustedImproved(nn.Module):
97     def __init__(self, output_height=646, output_width=251):
98         super(CustomDecoderAdjustedImproved, self).__init__()
99         self.output_height = output_height
100        self.output_width = output_width
101        self.decoder = CustomDecoderImproved()
102
103    def forward(self, x):
104        x = self.decoder(x)
105        x = F.interpolate(x, size=(self.output_height, self.
106        output_width), mode='bilinear', align_corners=False)
107        return x
108
109 class CustomCrossDomainAEAdjustedUnet_improved(nn.Module):
110     def __init__(self):
111         super(CustomCrossDomainAEAdjustedUnet_improved, self)
112         .__init__()
113         self.encoder = UnetEncoderImproved(input_channels=1,
114         num_filters=64)
115         self.feature_mapping = FeatureMappingLayer(512, 256)
116         self.decoder = CustomDecoderAdjustedImproved(

```

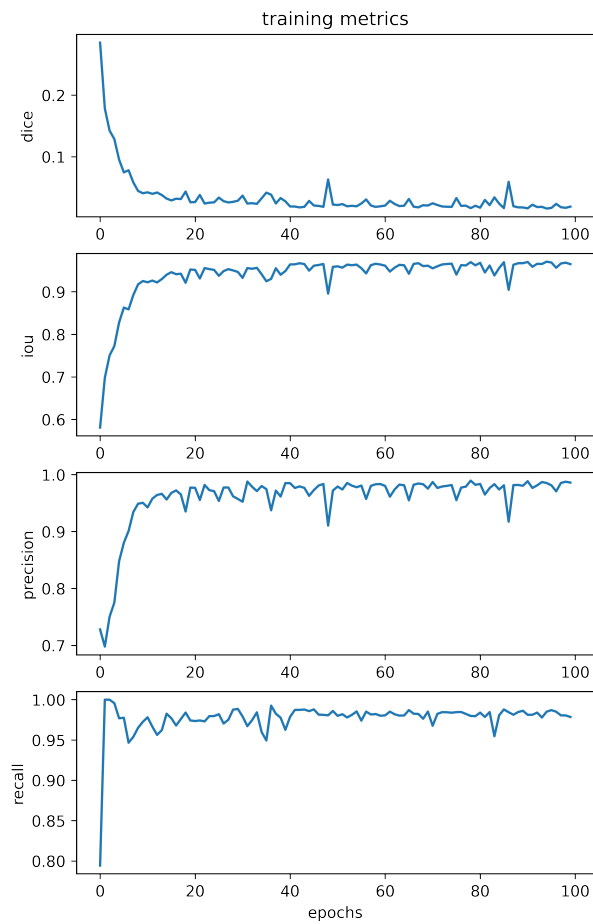
```
output_height=646, output_width=251)
115
116     def forward(self, x):
117         encoded = self.encoder(x)
118         mapped = self.feature_mapping(encoded)
119         decoded = self.decoder(mapped)
120         return decoded
```

Appendix B

Apêndice

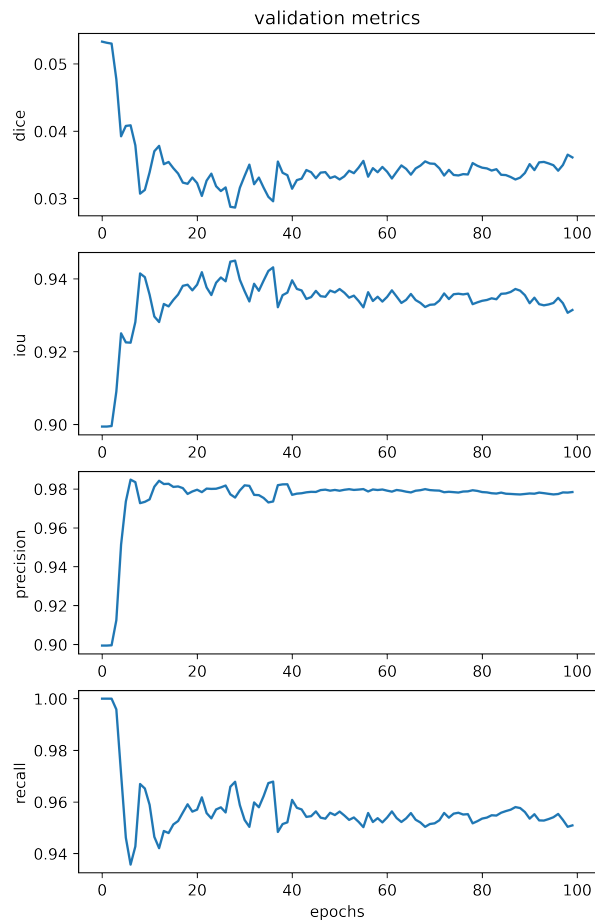
B.1 Minimal amount of samples

Figure B.1: Training metrics with 12.5% of training images.



Source: The author

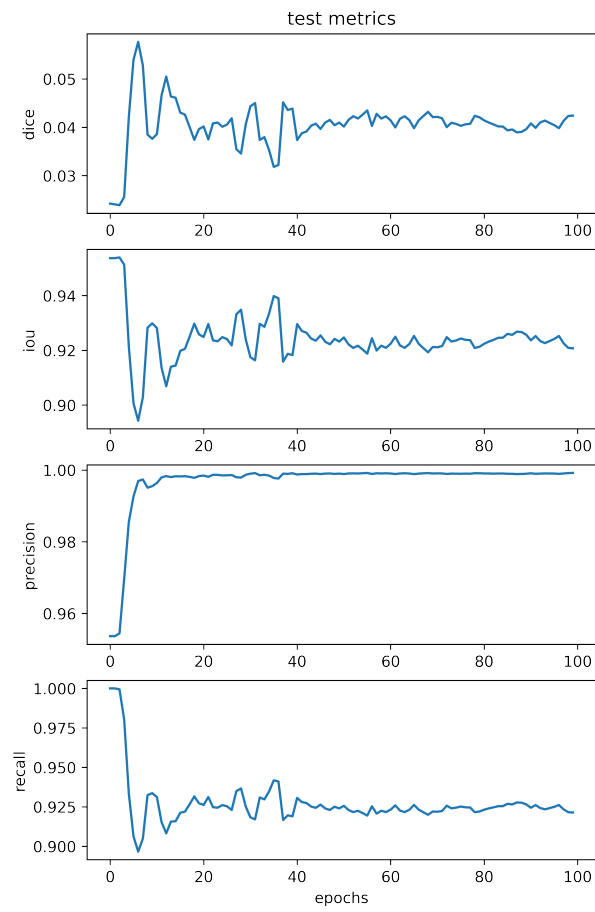
Figure B.2: Validation metrics with 12.5% of training images



Source: The author

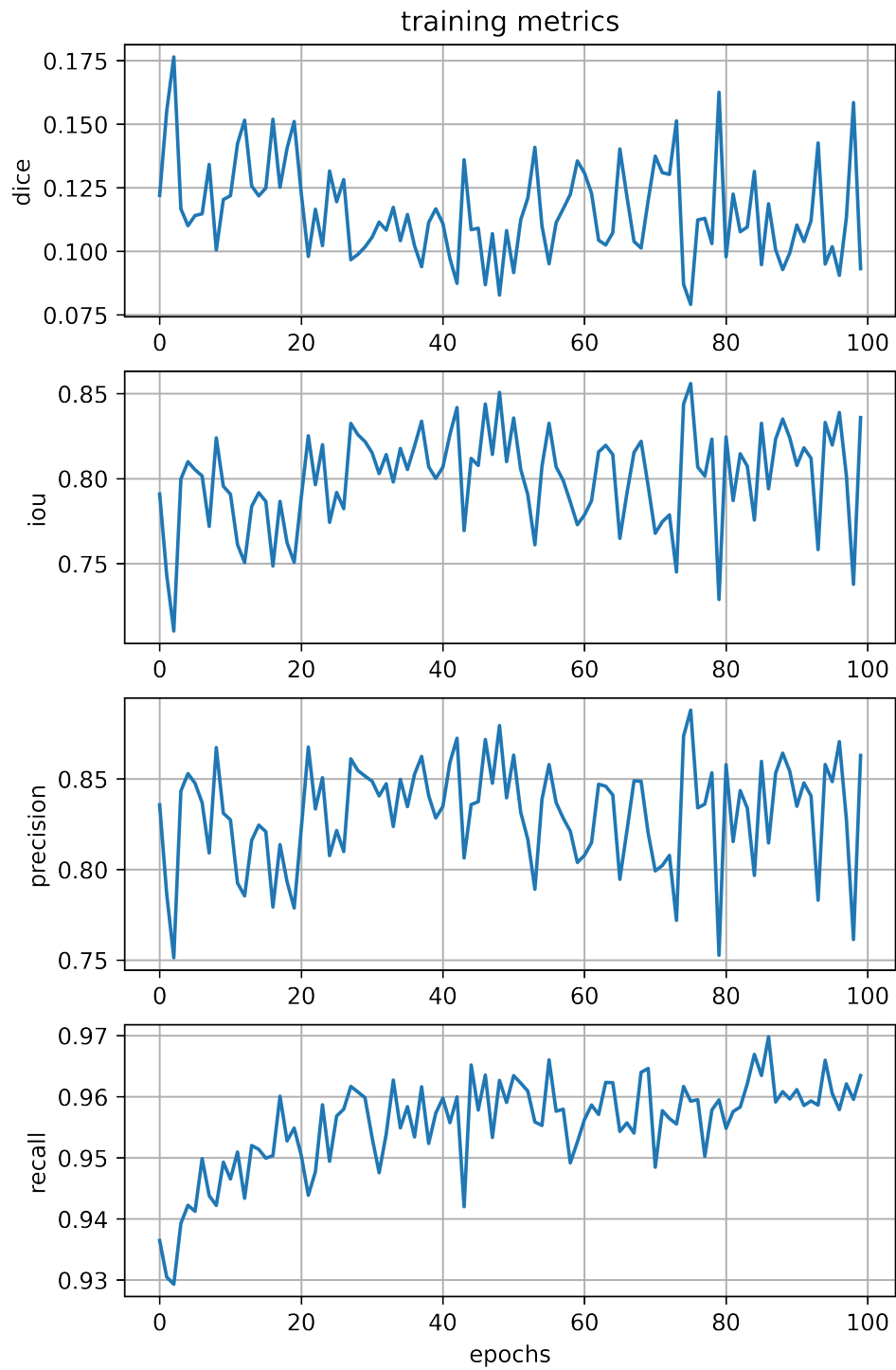
B.2 ANP Dataset

Figure B.3: Test metrics with 12.5% of training images



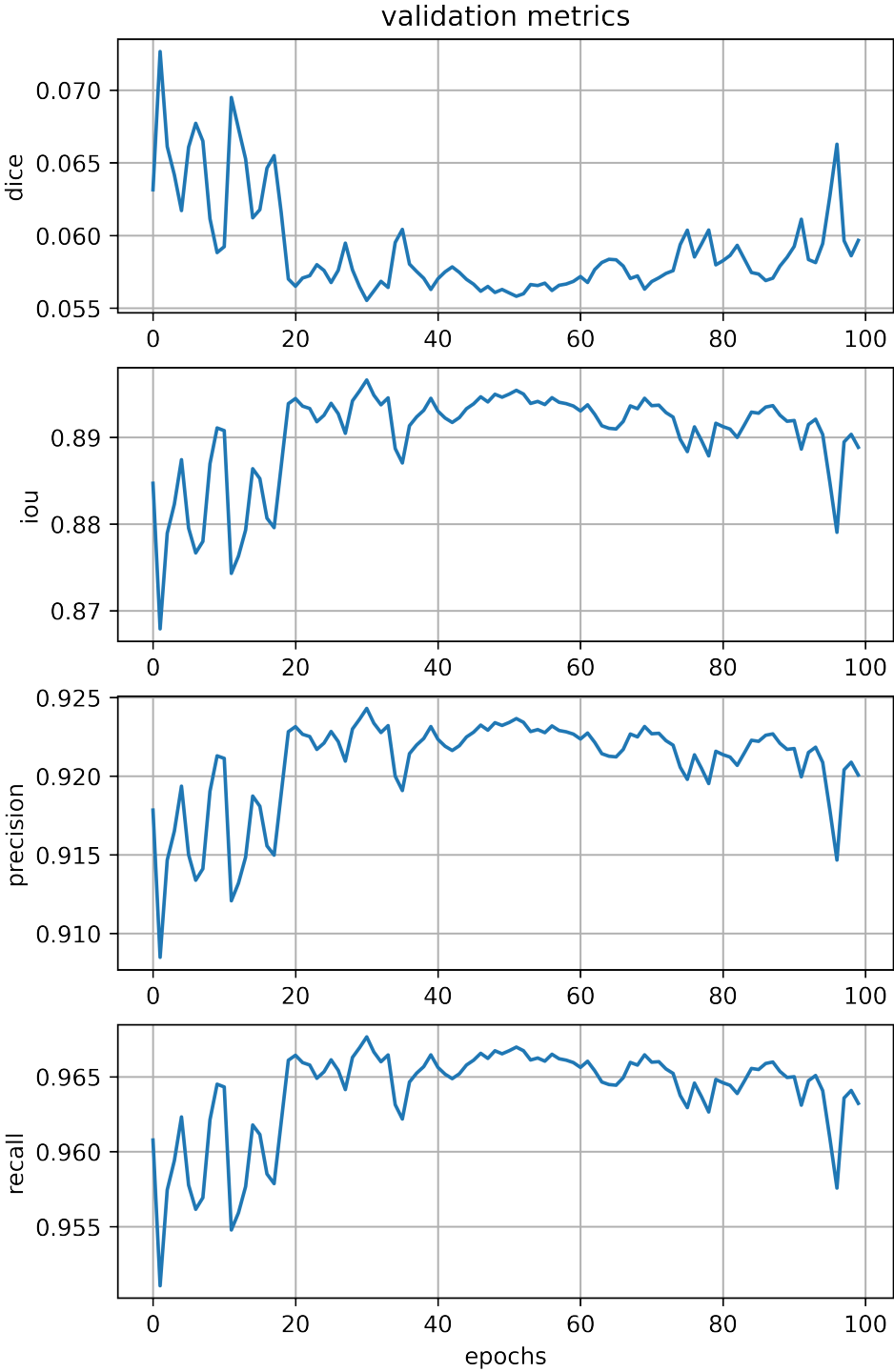
Source: The author

Figure B.4: Training metrics for the XZ section



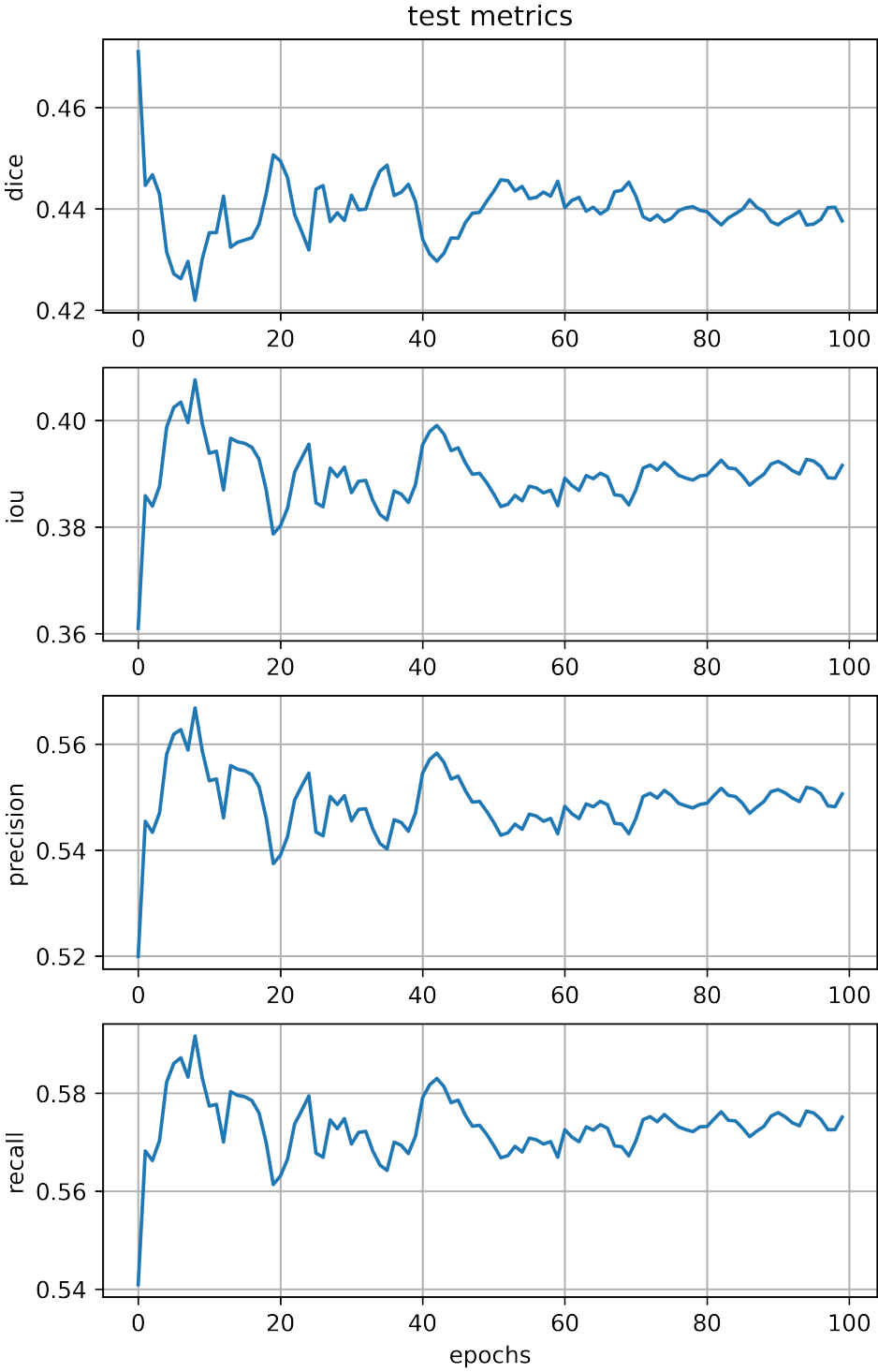
Source: The author

Figure B.5: Validation metrics for the XZ section



Source: The author

Figure B.6: Test metrics for the XZ section



Source: The author