TGS Salt Identification Challenge

Naomi Takemoto
RA 184849
n184849@dac.unicamp.br

Thamiris Florindo Coelho
RA 187506
thamycoelho10@gmail.com

Thiago Dong Chen
RA 187560
thiagodchen@gmail.com

I. Resumo

Este relatório apresenta os resultados do uso de *machine learning* para aumentar a confiabilidade, velocidade e uniformidade do processo de identificação de camadas de sal. O objetivo do *TGS Salt Identification Challenge* foi realizar a segmentação semântica, isto é, uma classificação pixel da imagem sísmica de entrada, de modo a indicar a presença ou não de sal [2]. Para resolver o problema de segmentação de imagens sísmicas foi proposto a utilização da rede U-Net e outras técnicas para melhorar a performance da rede.

II. Introdução

A. Descrição do Problema

A ocorrência de petróleo em diversas áreas da subsuperfície terrestre em muitos casos está relacionada a depósitos de sal. Isso faz com que a identificação dessas estruturas seja de particular interesse para empresas do ramo de exploração de hidrocarbonetos. No entanto, encontrar tais áreas de forma precisa pode se tornar uma tarefa bastante difícil, pois a análise de dados sísmicos exige o trabalho de intérpretes especialistas em detecção de camadas de sal. Um problema que surge em decorrência disso é que as análises se tornam subjetivas e variam de profissional para profissional o que pode diminuir a confiabilidade do processo. [2]

B. Dados

Para treinamento e validação, foram disponibilizadas 4000 imagens sísmicas de 101x101px com 4000 *labels* correspondentes que consistem nas máscaras (segmentação), estas possuem mesma dimensão mencionada. Além disso, a cada um dos pares descritos anteriormente estavam associados uma medida de profundidade disponibilizada em um arquivo *csv*.

Na competição, a validação das submissões (feitas pela plataforma *Kaggle*) foram realizadas em 33% dos dados de teste, que não foram disponibilizados até o encerramento do concurso. Os dados de teste (com 9000 pares de imagens ao todo) só foram utilizados ao final da competição.

III. Metodologia

A. Métrica de Desempenho

A seção abaixo foi baseada na explicação encontrada na referência [4]. A competição utilizou como métrica de desempenho a precisão média do IoU (*intersection over union*), também conhecida como índice de Jaccard, para diferentes limiares. A fórmula para o IoU é apresentada a seguir:

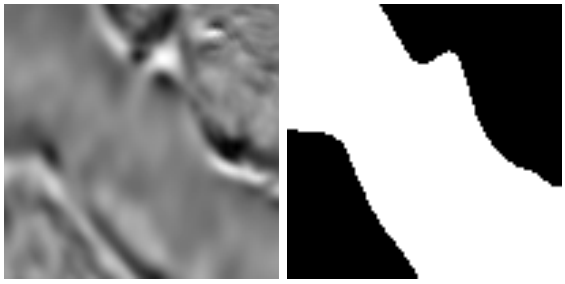

Figura 1. À esquerda uma imagem sísmica, à direita a segmentação correspondente (áreas em branco representam a presença de sal e áreas pretas a ausência)

$$IoU(A,B) = \frac{A \cap B}{A \cup B}. \tag{1}$$

Os limiares t utilizados variavam de 0.5 a 0.95 com um passo de 0.05. Ao utilizar um limiar de 0.5, a classificação sobre determinado objeto é considerada correta se o IoU entre imagem segmentada gerada o *label* for maior ou igual a 0.5. Para cada t, o valor de precisão era calculado de acordo com a fórmula:

$$\frac{TP(t)}{TP(t) + FP(t) + FN(t)} \tag{2}$$

Onde $TP(t)$, $FP(t)$ e $FN(t)$ são respectivamente o número de verdadeiros positivos, falsos positivos e falsos negativos resultantes da comparação entre a máscara gerada no processo e o *label*.

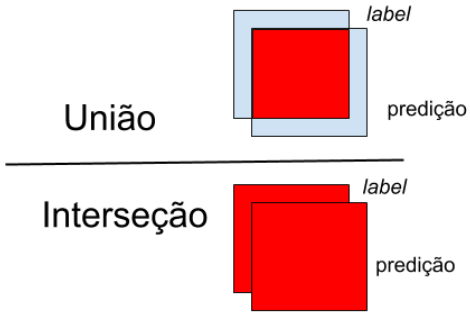

Figura 2. Representação da métrica de IoU

A precisão média de uma única imagem é calcula como a média tomada sobre o valor de precisão para cada valor de IoU de limiar t, de acordo com:

Tabela I
EXPLICAÇÃO SOBRE OS TERMOS DA FÓRMULA PARA O CÁLCULO DE PRECISÃO

label	**predição**	**estado**
sal	sal	TP
sal	não sal	FN
não sal	sal	FP

$$\frac{1}{|thresholds|}\sum_{t}\frac{TP(t)}{TP(t)+FP(t)+FN(t)}. \quad (3)$$

Finalmente, o *score* contabilizado pela métrica da competição é a média tomada sobre todas as precisões médias de cada imagem do *dataset*.

IV. EXPERIMENTOS

1) O primeiro experimento realizado foi o treinamento da rede a U-Net original que será útil como base de comparação com os outros modelos a serem propostos.
2) Em seguida, o segundo experimento utilizou a regularização *Dropout* como método de prevenção de *overfitting*, isto é, deixar que o modelo consiga generalizar conjuntos de amostras diferentes do conjunto de treinamento. Neste experimento, foram divididas em duas etapas. Na primeira etapa foram adicionadas *dropouts* nas camadas de convoluções. Na segunda etapa, foram adicionadas *dropouts* nas camadas de *pooling*.
3) Na terceira parte do experimento, realizamos a aumentação de dados, mas para isso, foi necessário tomar cuidado para não realizar aumentação de dados de maneira que não faça sentido para o problema. Visto isso, dobramos a quantidade de dados ao realizar o espelhamento horizontal da imagem, ver Figura 3.

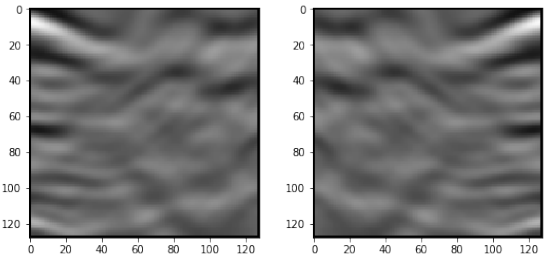

Figura 3. À esquerda uma imagem sísmica original, à direita a imagem espelhada horizontalmente.)

4) O quarto experimento utilizou a normalização de *batch*, pois segundo a literatura [5], a adição das camadas de normalização de *batch* ajuda a aumentar a velocidade da rede neural.
5) O quinto experimento utilizou a U-Net com transferência de aprendizado com os pesos pré-treinados de detecção de características nas imagens de satélite.

Após o quinto modelo gerado, a competição chegou ao fim e vários dos participantes divulgaram as estratégias que utilizaram. Estudamos algumas das abordagens adotadas por alguns dos competidores, em especial o uso de modelos pré-treinados que mesclam arquiteturas de redes de classificação com redes de segmentação, tendo como base o código disponibilizado em [12].

V. MOTIVAÇÕES PARA OS EXPERIMENTOS

Nesta seção apresentamos brevemente as motivações e embasamentos teóricos que motivaram a realização dos experimentos descritos acima.

A. Escolha da U-Net

A rede U-Net foi originalmente proposta para segmentação de imagens biomédicas e o seu uso se tornou popular para a segmentação em diversos problemas. A rede pode ser dividida em duas partes principais *Down* e *Up*. A primeira possui uma arquitetura semelhante a de uma rede convolucional de classificação e seu propósito é semelhante, atuar como extratora de *features*. Esse pedaço da U-Net possui camadas convolucionais seguidas de *max pooling* que realizam o a diminuições consecutivas nas nas dimensões de altura e larguras das imagens de entrada (*downsampling*) enquanto o número de canais aumenta. A segunda parte, *Up*, em contraste com a primeira, realiza o processo inverso, aumentando altura e largura e diminuindo o número de canais, para que a imagem segmentada tenha as dimensões desejadas.

A escolha de U-Net como *baseline* se baseou nas vantagens que essa arquitetura apresentou no tratamento de problemas de segmentação de imagens em imagens médicas (propósito original) dessa rede. Características que foram ressaltadas nos fóruns de discussão da competição, onde os participantes forneceram dicas para começar a abordar o problema. Os principais pontos positivos são:

1) Eficiência computacional em decorrência do número de parâmetros relativamente pequeno.
2) Treinável em *datasets* com poucos exemplos. Neste problema, foram disponibilizados apenas 4000 pares de imagens, que é um número reduzido levando se em conta o tamanho de *datasets* de classificação como por exemplo MNIST para classificação de dígitos (60000 para treino e 10000 para validação) ou a ImageNet que conta com mais de 1000000 de imagens para treinamento.

As implementações iniciais que utilizamos foi inspirada principalmente pelas dicas fornecidas pelo participante Jesper Dramsch em uma de suas postagens [11].

B. Aumentação de dados

Uma maior quantidade de dados, e principalmente uma maior variedade de dados pode aumentar a capacidade de generalização de uma rede neural, reduzindo problemas de *overfitting*. As técnicas mais comuns de aumentação em imagens consistem em translações, rotações, cortes, mudanças de contraste e cores, etc.

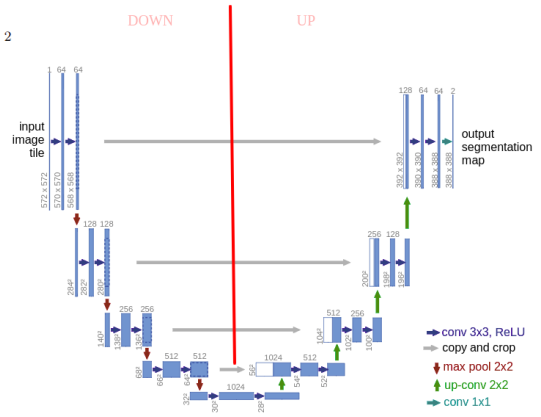

Figura 4. Arquitetura da rede U-Net. Disponível em [6]

C. Normalização de Batch

Este é um outro método para realizar a regularização de redes neurais convolucionais, que possui ainda a vantagem de aumentar a velocidade do treinamento, pois ameniza o efeito de desaparecimento dos gradientes, problema comum em redes mais profundas. Esta técnica tem sido usada para substituir o uso de *dropout* em redes convolucionais mais recentes [7] por apresentar resultados melhores, como pode ser visto na seção de resultados.

D. Transferência de Aprendizado

Uma outra estratégia para evitar *overfitting* é o uso de transferência de aprendizado. Segundo estudos realizados por Yosinski *et al.* [8], a inicialização de uma rede com *features* pré treinadas resulta em um aumento da capacidade de generalização, mesmo após retreino. No experimento 4 realizado, utilizou-se pesos obtidos no treinamento da U-Net em um problema de segmentação de imagens de satélite (disponibilizados por um dos participantes [9] de uma competição antiga do Kaggle) como parâmetros iniciais. Posteriormente, realizou-se o *fine tunning* de todas as camadas. A escolha de retreinar todas as camadas foi tomada levando-se em consideração que os dois desafios contavam com dados de características bastante distintas.

E. Early Stopping

Durante o treinamento inicial dos modelos, utilizamos a ferramenta de *early stopping* disponibilizada pelo *framework Keras* para avaliar cada uma das estratégias. A intenção foi parar o treinamento quando o erro no conjunto de validação parasse de reduzir, o que diminuiria o tempo de treino, permitindo o desenvolvimento de mais modelos.

VI. Resultados

Utilizando os resultados do experimento 1) para comparações, que realiza o treinamento da rede U-Net original, sem nenhuma modificação e inicialização dos pesos aleatoriamente,

Tabela II
Acurácia das estratégias utilizadas na U-net

Modelo	***Acurácia Conjunto de Treinamento(%)***	***Acurácia Conjunto de validação(%)***
U-Net	81,8	81,9
Dropout nas camadas de convolução	61,3	61,7
Dropout nas camadas de *pooling*	69,5	69,6
Aumentação de dados	86,0	86,1
Normalização de *batch*	75,6	75,6
Transferência de aprendizado	87,8	87,9
ResNeXt50 *Backbone*	89,8	89,8
ResNet50 *Backbone*	87,6	87,7

(Ver Tabela II), obtivemos a acurácia de 81,8% no conjunto de treinamento e 82,9%. Aparentemente, o modelo generalizou bem, pois tanto as duas acurácias estão bem próximas.

Comparando os resultados dos experimentos com a técnica de regularização de *dropout* era esperado que a acurácia no conjunto de validação fosse maior que a acurácia no experimento utilizando a U-Net original. Entretanto, a acurácia do *dropout* nas camadas de convolução foi de 61,7%, (ver Tabela II), isto significa que houve uma queda de 24,7% de acurácia.

Visto este resultado, tentamos aplicar o *dropout* em outra parte da U-Net, nas camadas de *pooling*. Mesmo aplicando a regularização em outra parte da rede, a acurácia no conjunto de validação continuou caindo. Neste experimento, a acurácia obtida foi de 69,6%, isto é, uma queda de 15,6%.

Essa queda de acurácia na tentativa de utilizar o regularizador *dropout* é explicado por Harrison Jansma [7]. Ele explica que as redes convolucionais possuem poucos parâmetros, logo elas precisam menos de regularização. Ademais, devido a relação espaciais codificados nos mapas de *features*, as funções de ativações podem podem tornar altamente correlacionadas. Dessa forma, o *dropout* torna-se menos relevante. O Harrison Jansma também recomenda a utilização da normalização de *batch* que foi aplicado neste trabalho.

A aumentação de dados utilizando o espelhamento horizontal foi o nosso primeiro experimento que a acurácia subiu em comparação ao modelo inicial da U-Net, a U-Net original com aumentação de dados teve a acurácia de 86,1%, um aumento de 5,1%. Este resultado mostra a importância da quantidade de dados para deixar o modelo mais robusto e mais confiável.

Apesar da literatura [7] comentar a eficiência da normalização de *batch*, ao colocar as camadas de normalização *batch* entre as camadas de convolução, a acurácia é de 62,1%.

O experimento, com a U-Net usando transferência de aprendizado com os pesos pré-treinados de detecção de características nas imagens de satélite, obteve a acurácia de 87,9%. Isso mostra que o *transfer learning* é poderoso para treinar o modelo sem necessitar o inicializar o modelo randomicamente, isto permite que o modelo convirja mais rapidamente, considerando que os pesos iniciais consegue extrair *features*.

A partir do sucesso do *transfer learning* e ao final da competição, quando vários participantes compartilharam as táticas que utilizaram, pesquisamos sobre a técnica de *backbone*, que consiste em combinar redes convolucionais de classificação, como por exemplo VGG, ResNet, etc, com redes de segmentação. A ideia é aproveitar os pesos pré treinados em *datasets* como ImageNet, que possuem grande variabilidade de dados para melhorar a capacidade de extração de características. Ao usar ResNet50 como *backbone* em conjunto com a U-Net, que realiza a tarefa de segmentação de imagem, (ver Figura 6) o que acontece é que na parte de *downsampling*, (ver Figura 5), é realizada por blocos da ResNet já com pesos pré-treinados. O modelo pré-treinado com *backbone* ResNeXt50 e ResNet apresentam, respectivamente, a acurácia de 89,8% e 87,7%.

A figura 8 mostra que a partir do intervalo de 20 a 30 épocas, o erro no conjunto de validação começa a aumentar com o treinamento e de acordo com a figura. Assim, suspeitamos que poderíamos parar o treinamento utilizando a função de *early stopping* do Keras, que para de treinar o modelo após um número de épocas sem nenhum ganho no erro da validação. Entretanto, o *early stopping* faz com que a acurácia de todos os modelos cair, pois os modelos ainda não chegaram no ponto de mínimo convergência. Dessa forma, descartamos o uso do *early stopping* e utilizamos o decaimento do *learning rate* quando o o modelo para de melhorar, isso faz com que a rede convirja mais perto do mínimo local.

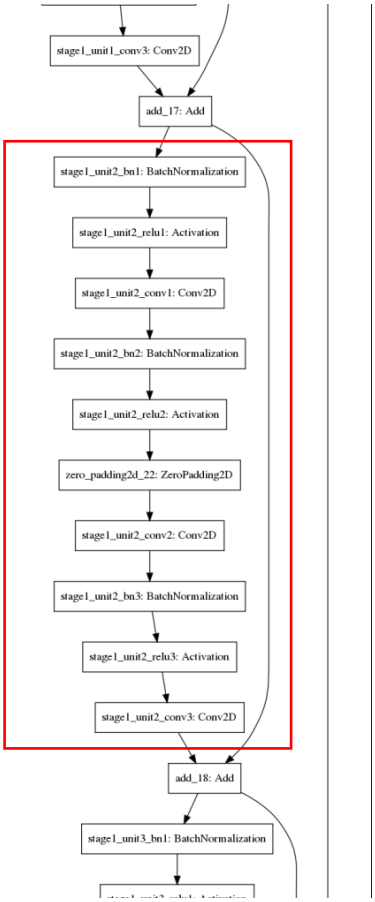

Figura 5. Bloco residual da ResNet50 utilizado na U-Net para a extração de *features*.

Tabela III
ACURÁCIA NO CONJUNTO DE VALIDAÇÃO DO *Kaggle*

Modelo	***Acurácia (%)***
U-Net	61,3
Dropout na camada de convolução	55,9
Dropout na camada de *pooling*	62,9
Transferência de aprendizado	56,6
Normalização do *Batch*	62,1
ResNeXt50 *Backbone*	6,8
ResNet50 *Backbone*	73,3

As Figuras 6 e 5 mostram os blocos do modelo com o *backbone* da ResNet-50. Estes blocos possuem normalização de *batch*, porém em locais diferentes do proposto no quarto experimento. Isso pode ser um fator que causou a diminuição da acurácia, (Ver Tabela II).

VII. Escolha do Melhor modelo

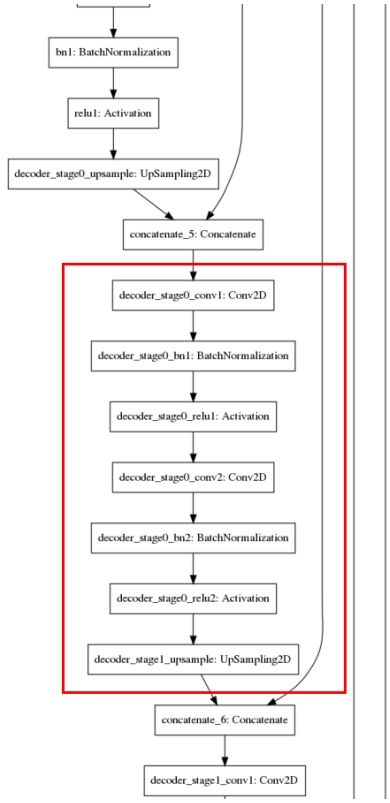

Figura 6. Bloco da etapa *Bloco upsampling da rede ResNet50 backbone*.

A nossa posição no último dia do desafio foi 2819 de 3234 usando o modelo U-Net, porém continuamos estudando e o melhor modelo foi obtido após o encerramento da competição. A escolha foi baseada na acurácia do conjunto de validação do *Kaggle*, ver Tabela III. Este modelo foi o ResNet50 *backbone*. Se a competição ainda estivesse subiríamos em 564 posições.

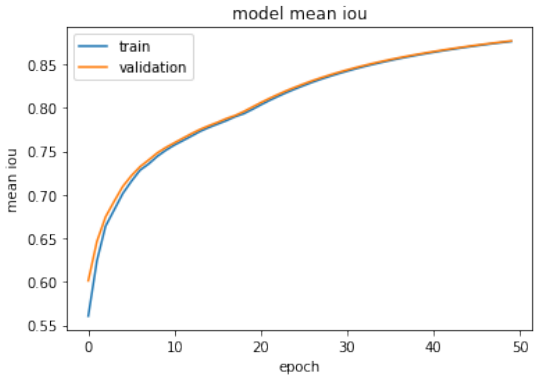

Figura 7. Acurácia do melhor modelo em função do número de épocas

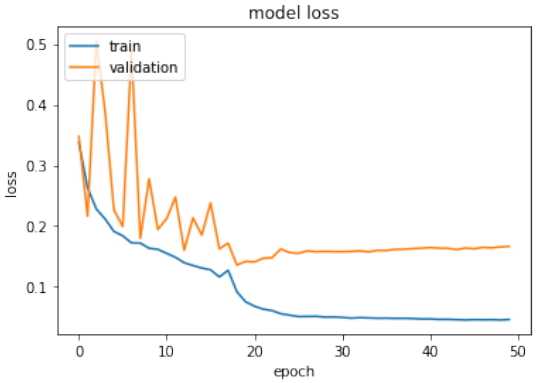

Figura 8. Custo do melhor modelo em função do número de épocas

VIII. Conclusões

Ao final do projeto, foi possível testar várias técnicas para melhorar a performance dos modelos. Devido ao fato de de a última estratégia utilizada (*backbone*) ter gerado o maior aumento de desempenho, pudemos perceber a importância de técnicas de *transfer learning*, que permitiu a extração de *features* relevantes, que seriam posteriormente utilizadas para a reconstrução da imagem na parte "*Up*"da rede U-Net. Em trabalhos futuros, um artifício não explorado neste projeto, mas que pode gerar ganhos de performance é o *ensemble*. Com esse método poderíamos combinar diferentes os diferentes modelos gerados para melhorar o desempenho na classificação. Um outro ponto que poderia trazer ganhos de performance é considerar os dados de profundidade, que não foram usados em nenhuma de nossas abordagens, tampouco apareceram de forma relevante em soluções e discussões de outros participantes que pesquisamos.

Referências

[1] Christopher M. Bishop. "Pattern Recognition and Machine Learning". Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] TGS Salt Identification Challenge. Disponível em : <https://www.kaggle.com/c/tgs-salt-identification-challenge>

[3] Wang J. & Perez L. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning.

[4] Kaggle's Evaluation. Disponível em: <https://www.kaggle.com/c/tgs-salt-identification-challenge#evaluation>. Acesso em: 5 de dezembro de 2018.

[5] Batch normalization in Neural Networks. <https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c>. Acesso em: Acesso em: 5 de dezembro de 2018.

[6] Disponível em : https://medium.com/@keremturgutlu/semantic-segmentation-u-net-part-1-d8d6f6005066. Acesso em: 6 de dezembro de 2018.

[7] Don't Use Dropout in Convolutional Networks. https://towardsdatascience.com/dont-use-dropout-in-convolutional-networks-81486c823c16. Acesso em: 6 de dezembro de 2018.

[8] Yosinski, J.; Clune, J.; Bengio, Y.;Lipson, H. How transferable are features in deep neural networks?. Disponível em <https://arxiv.org/pdf/1411.1792.pdf>. Acesso em: 6 de dezembro de 2018.

[9] Disponível em: <https://github.com/ZFTurbo/ZF_UNET_224_Pretrained_Mode>. Acesso em: 6 de dezembro de 2018.

[10] Dstl Satellite Imagery Feature Detection. Disponível em: <https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection>. Acesso em: 6 de dezembro de 2018.

[11] Dramsch, J. Intro to seismic, salt, and how to geophysics. Disponível em: <https://www.kaggle.com/jesperdramsch/intro-to-seismic-salt-and-how-to-geophysics>

[12] Segmentation models Zoo. Disponível em: <https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/61968#380909>. Aceso em: 6 de dezembro de 2018.