Pontifícia Universidade Católica
do Rio de Janeiro

Mateus Cabral Torres

Redes Convolucionais aplicadas à Segmentação Semântica de Imagens Sísmicas

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio.

Orientador: Prof. Sérgio Colcher

Rio de Janeiro
Maio de 2021

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Mateus Cabral Torres

Redes Convolucionais aplicadas à Segmentação Semântica de Imagens Sísmicas

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo.

Prof. Sérgio Colcher
Orientador
Departamento de Informática – PUC-Rio

Prof. Edward Hermann Haeusler
Departamento de Informática – PUC-Rio

Prof. André Bulcão
Petróleo Brasileiro – Rio de Janeiro - Matriz

Rio de Janeiro, 17 de maio de 2021

Mateus Cabral Torres

Graduado em Física pela Universidade Federal Fluminense e pela Universidade de Coimbra em 2014.

Ficha Catalográfica

Torres, Mateus Cabral

Redes convolucionais aplicadas à segmentação semântica de imagens sísmicas / Mateus Cabral Torres ; orientador: Sérgio Colcher. – 2021.
76 f. : il. color. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2021.
Inclui bibliografia

1. Informática - Teses. 2. Transferência de aprendizado. 3. Redes neurais convolucionais. 4. Segmentação semântica. 5. Detecção de sal. 6. Processamento de imagens. I. Colcher, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Para a minha família, no sentido mais
extenso da palavra, pela base sólida
que me sustenta hoje e sempre.

Agradecimentos

Quero agradecer primeiramente à Pontifícia Universidade Católica do Rio de Janeiro, em especial aos professores e funcionários do Departamento de Informática, não apenas pela qualidade do ensino reconhecido internacionalmente, mas também pelo imenso acolhimento que despertou em mim um renovado interesse e paixão pela vida acadêmica. Em especial gostaria de citar os professores Ruy Milidiú e Sérgio Colcher pela orientação e pelas oportunidades oferecidas de se descobrir todo um novo campo de conhecimento colocando a mão na massa. Esse diferencial da PUC-Rio em aproximar a atividade acadêmica da prática é essencial e deve servir de exemplo para todas as instituições de ensino superior, em minha humilde opinião.

No campo mais pessoal, não há como não citar em primeiro lugar meus pais e meu irmão. Ainda que a distância atrapalhe os abraços e as visitas noturnas de surpresa, ela não foi páreo para o apoio transatlântico absolutamente incondicional às minhas decisões e sonhos. O respeito e confiança que Mauricélia e Wilson depositam em mim desde muito pequeno (ouviam nos auges de meus 14 anos minhas opiniões apaixonadas sobre onde morar e saúde financeira da família) e a infinita gentileza, lealdade e inato senso de justiça de Tomás me deram a confiança e o discernimento necessários para sonhar irrestritamente com um futuro melhor para mim e para todos que eu amo.

E há minha família mais extensa! Minhas tias Vânia, Maria e Terê e meus queridos primos, que cultivam o legado matriarca de Dona Maria e sua eterna fé no poder transformador da educação, mas sem perder de vista os prazeres proporcionados por boas companhias. Por isso, agradeço muito a todos.

E continuo falando de família quando cito meus amigos. São muitos, mas me acompanham desde muito cedo Victor, Portella, Sales, Biro, Thaleco, Thomaz, Matheus e Murilo como meu núcleo de absoluta segurança e afeto. As incontáveis histórias que gostamos de repetir à exaustão em qualquer oportunidade e a perspectiva de novas aventuras dos Lixos Maravilhosos me fazem ter certeza que escolhi bem meus companheiros de vida. Digo com a mais absoluta certeza que eu não teria conseguido completar essa etapa da minha vida sem vocês e espero que possam sempre contar comigo para retribuir esse apoio (mesmo com minhas eventuais sumidas).

Meus companheiros de laboratório tiveram também importância fundamental nesta jornada, não apenas nas trocas intelectuais marcadas nos muitos rabiscos do quadro da 527, mas também pelas amizades profundas forjadas nas viagens apertadas de carro e pizzas encomendadas à meia noite no caos das entregas. Quero agradecer Betine, João e Pedro por serem incríveis companheiros de trincheira. Ao Rafa (O Bruxo!), pelo prazer de dividir momentos incríveis, seja discutindo filosofia e literatura, seja explorando blocos de carnaval pelo Rio. E principalmente quero agradecer ao apoio absurdo que Luis me deu durante todo este processo, sempre demonstrando não apenas sua inteligência quase inacreditável, mas também uma paciência e gentileza com as quais aprendo em cada conversa nossa.

Quero citar também pessoas que representam instituições, ensinamentos e momentos importantes que me fizeram superar os desafios do mestrado. O professor Jorge Simões de Sá Martins e seu incansável incentivo ao aprendizado e à curiosidade científica que moldam boa parte da minha visão de mundo. O professor Sebastião Alves Dias pela eterna gentileza perfeitamente misturada com lições de física de partículas e papos sobre o Mengão. As parceiras de luta Ju Custódio e Vivy Dutra que representam a importância que o TETO Brasil tem na minha renovada vontade de participar na luta para construção de uma sociedade mais justa. O professor André Ramos, da Orquestra Voadora, que me ajudou no resgate da alegria simples e infinita da brava arte do carnaval de rua.

Por fim, mas definitivamente não menos importante, quero agradecer o apoio de minha namorada, companheira e amiga Carol. Nós dois sabemos o que foi passar por provações dessa magnitude no meio da maior pandemia do século. O que eu não sabia ser possível era a existência de tamanha empatia, inteligência e elegância, que ela me mostrou não só ao me apoiar, me consolar e me carregar pra frente em alguns momentos mas também ao me tirar da casca rígida de minhas certezas e me apresentar a um mundo mais complexo e belo. Toda a dificuldade enfrentada por nós ajudou a solidificar um pouco desse nosso pedacinho de vida que cultivamos juntos, o que já é o bastante pra me fazer agradecer até pelos momentos mais sombrios. Te amo.

Ao CNPq e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Código de Financiamento 001.

Resumo

Torres, Mateus Cabral; Colcher, Sérgio. **Redes Convolucionais aplicadas à Segmentação Semântica de Imagens Sísmicas.** Rio de Janeiro, 2021. 76p. Dissertação de Mestrado − Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A partir de melhorias incrementais em uma conhecida rede neural convolucional (U-Net), diferentes técnicas são avaliadas quanto às suas performances na tarefa de segmentação semântica em imagens sísmicas. Mais especificamente, procura-se a identificação e delineamento de estruturas salinas no subsolo, o que é de grande relevância na indústria de óleo e gás para a exploração de petróleo em camadas pré-sal, por exemplo. Além disso, os desafios apresentados no tratamento destas imagens sísmicas se assemelham em muito aos encontrados em tarefas de áreas médicas como identificação de tumores e segmentação de tecidos, o que torna o estudo da tarefa em questão ainda mais valioso.

Este trabalho pretende sugerir uma metodologia adequada de abordagem à tarefa e produzir redes neurais capazes de segmentar imagens sísmicas com bons resultados dentro das métricas utilizadas. Para alcançar estes objetivos, diferentes estruturas de redes, transferência de aprendizado e técnicas de aumentação de dados são testadas em dois *datasets* com diferentes níveis de complexidade.

Palavras-chave

Aprendizado profundo; redes neurais convolucionais; segmentação semântica; detecção de sal; processamento de imagens; aumento de dados; aprendizado supervisionado; U-Net; FCN; transferência de aprendizado.

Abstract

Torres, Mateus Cabral; Colcher, Sérgio (Advisor). **Convolutional Networks Applied to Semantic Segmentation of Seismic Images.** Rio de Janeiro, 2021. 76p. Dissertação de Mestrado − Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Through incremental improvements in a well-known convolutional neural network (U-Net), different techniques are evaluated regarding their performance on the task of semantic segmentation of seismic images. More specifically, the objective is the better identification and outline of subsurface salt structures, which is a task of great relevance for the oil and gas industry in the exploration of pre-salt layers, for example. Besides that application, the challenges imposed by the treatment of seismic images also resemble those found in medical fields like tumor detection and tissue segmentation, which makes the study of this task even more valuable.

This work seeks to suggest a suitable methodology for the task and to yield neural networks that are capable of performing semantic segmentation of seismic images with good results regarding specific metrics. For that purpose, different network structures, transfer learning and data augmentation techniques are applied in two datasets with different levels of complexity.

Keywords

Deep learning; convolutional neural networks; semantic segmentation; salt detection; image processing; data augmentation; supervised learning; U-Net; FCN; transfer learning.

Sumário

Lista de figuras

Lista de tabelas

[...] Só nos conhecemos na medida
Em qu somos postos à prova.
Digo-lhes isso
Do meu coração, que desconheço.

Wisława Szymborska, Um minuto de silêncio por Ludwika Wawrzyńska

1. Introdução

Este trabalho aborda estratégias para a segmentação semântica de imagens sísmicas que permitam identificar áreas com presença de depósitos de sal abaixo do solo. O interesse por esta tarefa se justifica por ela exigir a análise e anotação manual por especialistas em um grande volume de dados, tomando assim uma grande quantidade de tempo.

A tarefa de segmentação semântica consiste em, a partir de uma imagem sísmica de entrada, gerar uma máscara que indique, pixel a pixel, a presença ou não de depósitos de sal. Além de longa, esta tarefa não é simples. Embora a segmentação de diferentes tipos de solo venha sendo bastante estudada e passe a usar cada vez mais as técnicas aqui abordadas, o caso do sal é especialmente difícil: de acordo com [1] "o tratamento de imagens sísmicas de evaporitos (ou depósitos salinos) é notoriamente difícil devido às suas complexas configurações" (em tradução livre).

A importância deste trabalho é evidenciada pela grande presença do tema "sal" em publicações, acadêmicas ou não, relacionadas à indústria petroquímica. Dentre as publicações na Society of Exploration Geophysicists (SEG) [2], por exemplo, a palavra-chave "salt" é encontrada em mais de onze mil artigos. Também pode-se lembrar do destaque na mídia dado à camada pré-sal de petróleo em poços no Brasil.

Neste trabalho, estuda-se a utilização de técnicas de aprendizado de máquina para endereçar este tipo de tarefa, mais especificamente usando redes neurais convolutivas (*convolutional neural networks* ou CNNs [35]) treinadas com métodos de aprendizado supervisionado, ou seja, a partir de dados cuja resposta já é sabida. Estes tipos de técnicas vêm ganhando destaque nos últimos anos como uma poderosa ferramenta para resolução de diversos tipos de problemas, com destaque para processamento de imagens, como é o caso.

Devido a esta popularidade, diversos tipos de redes e módulos adicionais são encontrados na literatura [5, 6, 7, 15, 17, 18, 19, 25, 26], sendo o objetivo desta dissertação abordar alguns dos mais promissores, bem como fazer ajustes, combinações e propostas para se alcançar o melhor resultado possível dentro dos *datasets* propostos.

Os dois *datasets* utilizados são oriundos do (1) desafio "TGS Salt Identification Challenge", proposto pelo conhecido site Kaggle [32], e de (2) um *dataset* fornecido pela Petrobras no âmbito do projeto com a PUC-Rio em que esta dissertação está inserida.

1.1. Objetivos

O objetivo do trabalho é a partir de uma rede simples (U-Net [4]), que incorpore as ideias básicas da classe de arquiteturas que se pretende utilizar, fazer graduais adições e observar os efeitos produzidos nos resultados. Isso não só permite um melhor entendimento dessas adições e da tarefa em si, como é o caminho para a proposição de uma rede eficiente para lidar com os *datasets* propostos.

Além disso, como a metodologia utilizada possui passos bem definidos, pode ser usada como um guia para os casos apresentados neste trabalho e também na experimentação em outros *datasets* e redes semelhantes.

A exploração de diferentes arquiteturas é complementada pela investigação e experimentação de diferentes técnicas de um tipo de pré-treinamento conhecido como *Transfer Learning* [36]. Nele, outros *datasets* são utilizados para iniciar os pesos da rede em uma posição mais vantajosa para seu aprendizado.

Por fim, pretende-se discutir aspectos relativos ao treinamento, como métricas de avaliação [37], função de perda [21], otimização [20] etc, bem como a elaboração do *dataset* em si, incluindo as discussões sobre técnicas de aumentação de dados [33].

1.2. Contribuições

Este trabalho apresenta um estudo cuidadoso para projetar um modelo de rede neural para cumprir a tarefa de segmentação semântica de sal. Desta forma, são feitas as seguintes contribuições: (1) Criação de redes neurais treinadas e ajustadas para realizarem segmentação semântica de depósitos de sal nos *datasets* estudados. A alta qualidade das redes construídas é atestada pelo alcance de 83.02% e 96.53% na métrica *Intersection over Union* (IoU) [37] nos grupos de

teste referentes aos *datasets* Petrobras e Kaggle, respectivamente; (2) Demonstração da importância do uso da técnica de *Transfer Learning* neste tipo de tarefa; (3) Proposição de diferentes estruturas que podem ser incorporadas a redes convolucionais para melhoria dos resultados em tarefas de segmentação semântica; (4) Melhoria de resultados a partir de diversos pós-processamentos realizados no dataset Petrobras.

Além de prover um estudo cuidadoso das técnicas utilizadas na tarefa de segmentação semântica e suas aplicações no contexto de imagens sísmicas, este trabalho tem como produto uma rede para aplicação em dados não disponíveis publicamente e um ambiente de produção construído para performar e analisar facilmente as centenas de experimentos realizados ao longo do projeto. Estes experimentos estão disponíveis para a Petrobras no contexto do projeto "Machine Learning na Geofísica" em parceria com o Departamento de Informática da PUC-Rio.

1.3. Organização

Uma vez introduzidos os objetivos, resta expor a estrutura em que esta dissertação se organiza. No capítulo 2, são discutidos os detalhes da tarefa de segmentação semântica de imagens sísmicas, passando brevemente pelo método de aquisição dos dados. Em seguida, no capítulo 3, são apresentadas as bases teóricas usadas na elaboração deste trabalho, o que inclui um breve histórico do uso das redes neurais em processamentos de imagens e trabalhos relacionados na área de imagens sísmicas. O capítulo 4 é dedicado à descrição da metodologia utilizada. Ele também inclui descrições dos *datasets* e os seus pré-processamentos, redes utilizadas, métodos de treinamento e pré-treinamento e avaliação. Os resultados do processo experimental são mostrados no capítulo 5. Finalmente, o capítulo 6 traz as conclusões, incluindo uma seção sobre possíveis trabalhos futuros. O capítulo 7 traz a bibliografia utilizada.

2. Identificação de sal em imagens

A segmentação semântica para a identificação de sal em imagens sísmicas traz em si diversas particularidades e desafios. Para que ela possa ser melhor compreendida, este capítulo traz informações mais detalhadas da tarefa, incluindo a forma de aquisição de dados e a descrição do que é a segmentação semântica, com aplicações a exemplos dentro e fora do universo das imagens sísmicas.

2.1. A aquisição de dados sísmicos

A página do desafio TGS no Kaggle [32] traz uma boa descrição do processo de aquisição dos dados, bem como as particularidades do caso da detecção de sal (em tradução livre):

> (...) O método requer uma fonte de energia sísmica controlada, como ar comprimido ou um vibrador sísmico, e sensores registram a reflexão [das ondas geradas por estas fontes] em interfaces de rochas na subsuperfície. Os dados são então processados para criar uma visão em 3D do interior da Terra. (...) A imagem sísmica mostra as fronteiras entre diferentes tipos de rochas. Na teoria, a intensidade da reflexão é diretamente proporcional à diferença das propriedades físicas em ambos os lados da interface. Enquanto as imagens sísmicas mostram fronteiras de rochas, elas não têm muito a dizer sobre as próprias rochas; algumas são fáceis de serem identificadas, enquanto outras são difíceis. (...) Um dos desafios da análise das imagens sísmicas é a identificação das partes da subsuperfície compostas de sal. O sal tem características que o torna ao mesmo tempo simples e difícil de identificar. A densidade do sal é geralmente de 2.14 g/cc, que é menor do que a maioria das rochas que o cerca. A velocidade sísmica do sal é de 4.5 km/s, que é normalmente mais rápida que as de rochas vizinhas. Esta diferença cria reflexões acentuadas na interface sal-sedimento. Normalmente o sal é uma rocha amorfa sem muitas estruturas internas. Isso significa que tipicamente não se tem muita refletividade dentro do sal, a não ser que haja sedimentos aprisionados nele. A velocidade excepcionalmente alta do sal pode criar problemas na criação da imagem sísmica. (TGS-NOPEC GEOPHYSICAL COMPANY ASA. Kaggle, 2019. Descrição de *dataset* utilizado na competição “TGS Salt Identification

Challenge”.Disponível em: https://www.kaggle.com/c/tgs-salt-identification-challenge. Acesso em: 14 de maio de 2021).

O processo descrito acima é ilustrado pela Figura 1, que mostra como ondas sônicas provenientes de uma pistola de ar (*air gun*) são refletidas nas camadas da subsuperfície e captadas pelos hidrofones para gerar o sinal do qual se originam as imagens sísmicas.

Importante notar na explicação do processo a relativa dificuldade de se obter informações sobre as texturas internas das rochas, bem como a ênfase dada na importância das interfaces rochosas na execução da tarefa. Essas características são importantes na diferenciação da identificação do sal de outras segmentações de imagens na próxima seção.

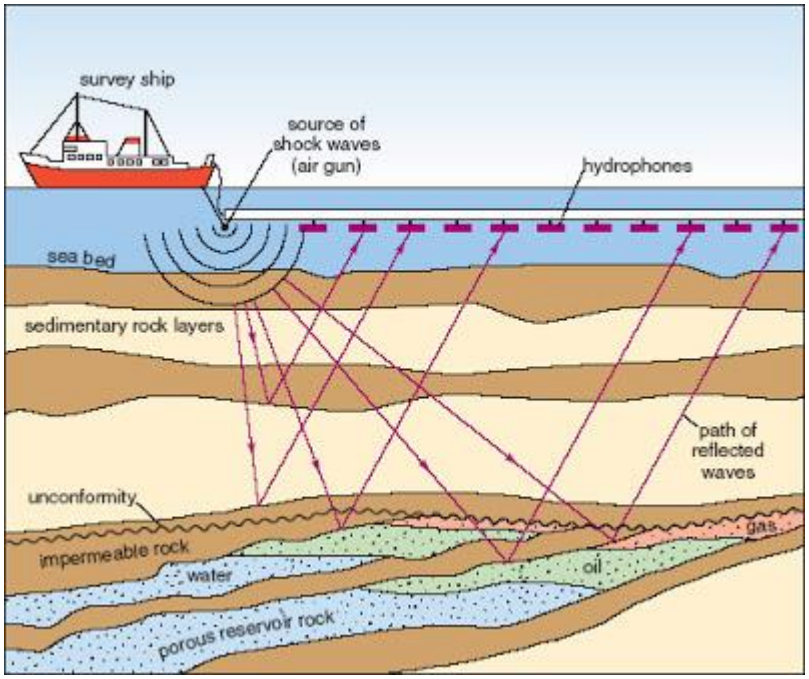


Figura 1 - Ilustração do processo de aquisição de dados em alto-mar.
Fonte: <http://geologylearn.blogspot.com/2015/06/marine-and-land-seismic-aquisition.html> Acessado em: 2 de abril de 2021.

Vale ressaltar que este é o processo através do qual o *dataset* Kaggle é obtido. Já o *dataset* Petrobras é composto por imagens sintéticas geradas por softwares que simulam estes efeitos.

2.2. Segmentação semântica e localização do sal

A segmentação semântica pode ser vista como uma generalização da tarefa clássica de classificação de imagens [38]. Mas em vez de dar um rótulo para cada imagem em sua totalidade, o objetivo na tarefa de segmentação semântica [8] é produzir uma máscara que traga a previsão correta para a classificação de cada pixel da imagem de entrada de acordo com classes previamente determinadas. A Figura 2 mostra um exemplo de uma segmentação em diversas classes.

Figura 2 - Exemplo de segmentação semântica
Fonte: <https://www.researchgate.net/figure/Example-of-2D-semantic-segmentation-Top-input-image-Bottom-prediction_fig3_326875064> Acessado em: 2 de abril de 2021.

A Figura 2 também ilustra bem uma das maiores motivações para o estudo desta tarefa: a aplicação a carros automatizados, que exigem uma precisão maior do que a oferecida com outros métodos de detecção conhecidos, como as *bounding-boxes* [40], além de uma eficiência alta o suficiente para ser realizada em tempo real [39].

A dificuldade de se fazer esta identificação pixel a pixel para um volume grande de imagens é um dos grandes fatores de motivação para o caso de uso em imagens sísmicas deste trabalho. A Figura 3 contém um exemplo de um recorte de uma imagem sísmica que, mesmo com apenas duas classes (sal e não-sal), mostra a dificuldade e a consequente necessidade de um especialista na elaboração da classificação.

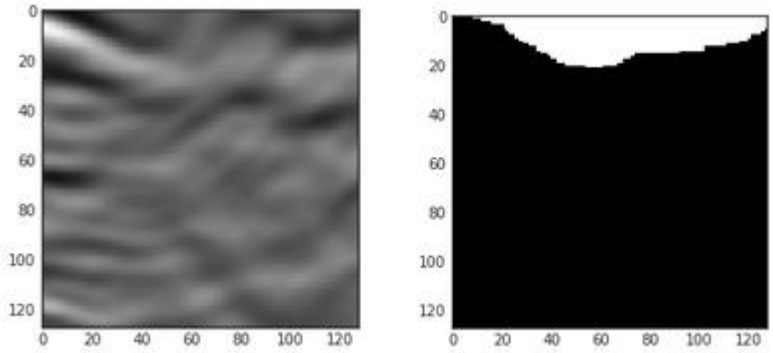


Figura 3: Segmentação semântica na tarefa de identificação de sal: *input* e *output*. O sal é representado pela cor branca na máscara à direita

Faz-se necessária uma diferenciação importante entre a tarefa realizada na Figura 3 em comparação com a Figura 2 com relação ao tamanho e ao contexto da imagem. Enquanto na primeira se tem um cenário completo, a segunda apresenta imagens com apenas 128x128 pixels, mostrando uma pequena área dentro do contexto total, ou um *patch*, como será chamado aqui. O aprendizado por redes neurais a partir desses *patches* é possível pela diferença fundamental entre os objetos que se procura identificar, com a imagem sísmica tendo uma ideia mais de *texturas e fronteiras*, enquanto objetos como carros e ruas requerem mais contexto para uma identificação precisa.

A divisão em *patches* é feita para que a rede seja treinada sem que se tenha *overfit*[1] [54] em relação à estrutura dos poucos reservatórios específicos a que se terá acesso nos treinos.

[1] Quando um modelo se ajusta com grande precisão aos dados já conhecidos mas falha em prever novos resultados

3. Conceitos Teóricos

Neste capítulo é feito um breve apanhado histórico do uso de CNNs na tarefa de segmentação semântica e, mais especificamente, em imagens sísmicas. Uma pequena introdução aos conceitos mais importantes também é realizada, deixando os detalhes mais específicos de cada rede para o próximo capítulo.

3.1. CNNs e classificação de imagens

A proposta de se atacar o problema em questão com Redes Neurais Convolucionais (CNNs), é justificada pelo grande sucesso que este tipo de rede obtém em tarefas de reconhecimento de imagens [41], o que fez com que esta abordagem se expandisse para outros campos [42, 43].

A classificação de imagens consiste em tomar uma imagem como *input* e classificá-la dentro de certas categorias pré-determinadas. Os desafios que um algoritmo de computação gráfica enfrenta incluem diversos tipos de variação que imagens da mesma categoria podem ter, como ponto de vista, escala, iluminação, etc. Para isso, o algoritmo tem que ser capaz de fazer uma classificação que seja, entre outras coisas, *invariante a translações, escalas e rotações, ou seja, à localização e posicionamento na figura* [25]. Isto será importante na análise do problema de segmentação semântica.

Métodos mais antigos de resolver este problema no paradigma de aprendizado de máquina incluem os classificadores lineares [44], por exemplo. Estes classificadores tentam, a partir da minimização de uma função de perda, encontrar matrizes que, aplicadas à imagem de *input*, resultem numa pontuação para cada categoria, como exemplificado na Figura 4. Repetições de estruturas como esta junto com funções de ativações dariam origem às Redes Neurais, que obtiveram grande êxito em diversas tarefas. Mas foram as CNNs que de fato revolucionaram o trato de imagens.

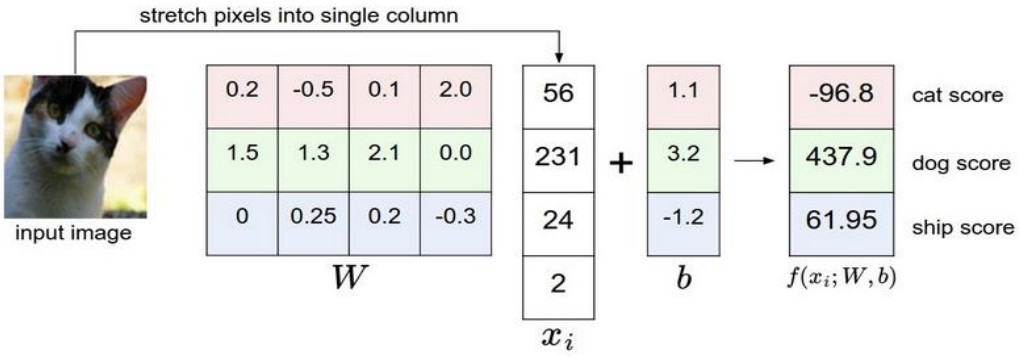


Figura 4: Exemplo de classificador linear.
Fonte: <http://cs231n.github.io/linear-classify/>

O termo CNN, como já foi dito, se refere às Redes Neurais Convolucionais, ou seja, redes neurais que usam convoluções [35]. A Figura 5 ilustra o resultado da operação de convolução, que gera o *output* a partir de multiplicações e somas do filtro convolutivo (K) com suas sucessivas sobreposições com o *input* (I). Esta operação veio resolver dois problemas: a necessidade frequente de um extenso pré-processamento das imagens e o número de parâmetros da rede.

O primeiro tem origem no fato de que pode ser muito útil para a rede saber se a imagem tem certos tipos de estruturas que a ajudem a diferenciar categorias - uma mesa, por exemplo, tem contornos retos muito mais frequentemente que gatos. Assim, certos tipos de filtros eram criados para captar essas interdependências locais entre pixels, o que poderia ser bastante trabalhoso e específico para cada tarefa. Os filtros convolutivos conseguem capturar essas estruturas de forma automatizada, se suficientemente treinados [45].

O segundo vem do fato de que multiplicações por matrizes tendem a crescer o número de parâmetros com bastante rapidez se elencadas como em Redes Neurais Profundas. As convoluções, por envolverem filtros geralmente bem menores que a imagem, utilizam também menos parâmetros.

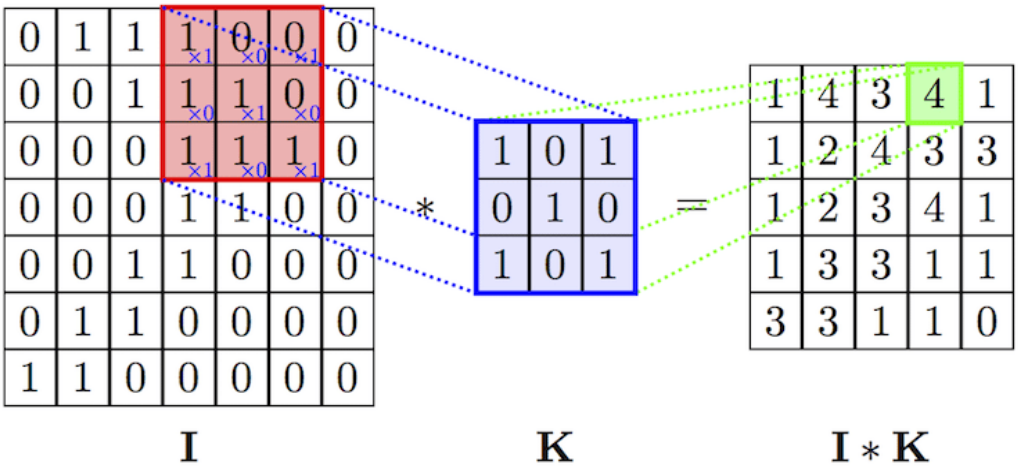


Figura 5: Operação de convolução.
Fonte:< https://www.researchgate.net/figure/An-example-of-convolution-operation-in-2D-2_fig3_324165524> Acessado em 10 de fev. 2021.

No entanto, como esses filtros convolutivos são pequenos em relação à imagem surge a questão de como captar relações entre pixels distantes uns dos outros. Com o objetivo de resolvê-la existem outros tipos de operação presentes em redes convolutivas, os chamados *poolings. Poolings* são operações que permitem reduzir o tamanho da imagem[2], de forma que os filtros aplicados pelas convoluções nessas imagens menores tenham uma região de influência maior em relação à imagem original.

Essa "área de influência" é comumente chamada de *receptive fields* (RFs), e serão cruciais na discussão sobre os desafios da segmentação semântica, já que com o objetivo de aumentá-los, os *poolings* mexem com a *estrutura de posição* da imagem (a posição de um pixel passa a representar cada vez mais pixels na imagem original conforme mais *poolings* são aplicados).

[2] Outras formas de reduzir o tamanho da imagem são possíveis, como convoluções com stride maior que 1.

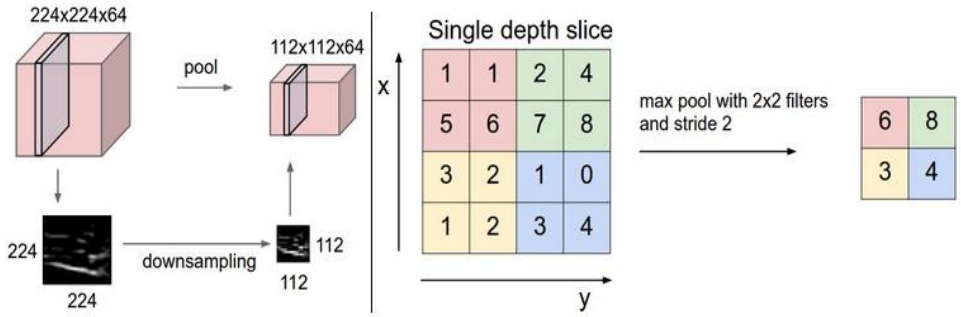


Figura 6 - Operação de maxpooling.
Fonte:< http://cs231n.github.io/convolutional-networks/>

Assim, está caracterizada de forma breve a estrutura de uma CNN para classificação: camadas de convoluções e *poolings* (e funções de ativação) que terão como função aprender a identificar estruturas em diversas escalas da imagem. Ao fim, há uma camada densa - ou seja, ao modo dos classificadores lineares - para se obter o resultado final [35].

3.2. CNNs em segmentação semântica

Como visto no capítulo anterior, a tarefa de segmentação semântica nada mais é do que uma classificação pixel a pixel da imagem de *input*. Sendo assim, é justificado que se usem redes análogas às usadas com tanto sucesso em classificação de imagens, embora sejam necessárias algumas modificações. A primeira modificação diz respeito às dimensões de *output*: se antes se tinha uma lista de "pontuações" para cada categoria, agora esta lista tem que ser gerada para cada pixel.

A segunda é mais sutil e diz respeito à estrutura da rede em si: se antes era esperada uma classificação invariante a aspectos de localização na imagem (translação, rotação e escala), agora a posição é de vital importância para classificar cada pixel. Como já foi discutido, camadas ao estilo *pooling* (e qualquer forma de *downsampling*) atrapalham este tipo de informação já que misturam informações entre pixels vizinhos. Então, outras estruturas terão que ser usadas para mitigar este efeito, fazendo com que a rede consiga minimamente reconstruir uma noção de posicionamento ao gerar o *output*. Aspectos da *classificação* e *localização* acabam por ter efeitos indesejados uma na outra.

O ótimo review por Lateef e Ruichek [8] traz diversos tipos de arquiteturas elaboradas pensando nestas questões, incluindo as duas ideias centrais que guiarão a escolha de arquitetura de redes neste trabalho: as *Fully Convolutional Networks* (FCNs) exemplificadas na Figura 7 e as estruturas *encoder*/*decoder*.

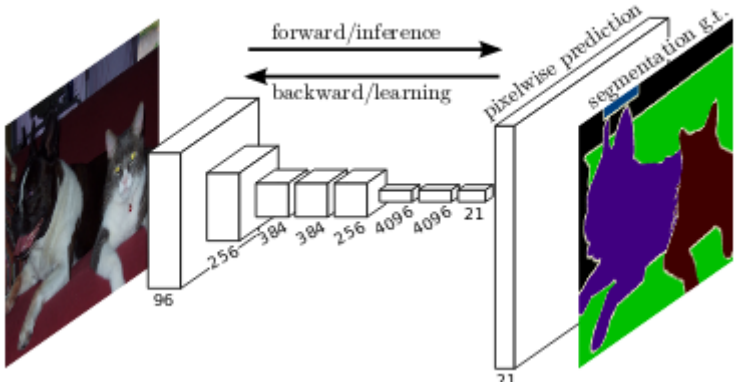


Figura 7: Fully convolutional Network.
Fonte: LONG J, SHELHAMER E, DARELL T, 2015 p.1

As FCNs se inspiram na estrutura de redes usadas em classificação de imagens para extrair as propriedades relevantes dos dados de entrada (*inputs)*. Sua diferença está na última camada que, em vez de ser densa, traz uma operação que retorna a imagem de volta ao tamanho necessário para a classificação pixel a pixel (*upsampling*), operação esta com parâmetros para serem aprendidos como no resto da rede. Estes requisitos são satisfeitos, por exemplo, pela operação de convolução transposta [51], que tem formulação análoga à convolução tradicional. Isto resolve o primeiro problema citado no início da seção e traz para a segmentação semântica a possibilidade de se utilizar redes já comprovadamente poderosas na classificação de imagens, já que as FCNs compartilham com elas a primeira parte de extração de atributos.

O passo a mais dado para tentar amenizar a incompatibilidade das operações de *downsampling* e informações posicionais em imagens vem na forma das estruturas de *encoder*/*decoder* incorporadas às FCNs. Essas estruturas tratam o *upsampling* da imagem da mesma forma que o *downsampling*: em etapas graduais. Isso permite a combinação de informações semânticas de camadas mais profundas com a informação espacial de camadas mais rasas, como introduzido pela famosa U-net [4] através de suas *skip connections*. Essas operações simples concatenam informações extraídas de camadas do *downsampling* com as de

upsampling, fazendo com que essas consigam reconstruir com mais facilidade informações posicionais da imagem.

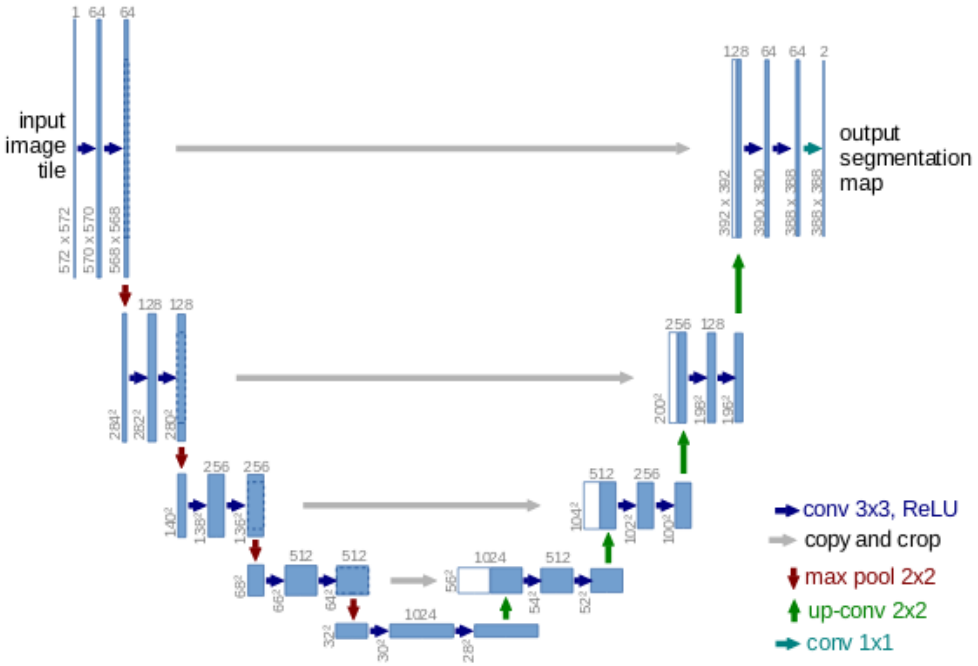


Figura 8: U-net.

Fonte: RONNEBERGER O, FISCHER P, BROX T, 2015 p.2

Sendo assim, este trabalho tem este tipo de arquitetura (especialmente a U-net, apresentada na Figura 8) como inspiração para o início da busca pela melhor rede possível para a tarefa. Estas redes não só têm se mostrado eficientes em outros *datasets*, mas também apresentam certa simplicidade em seus formatos, o que permite adicionar outros tipos de camadas e blocos facilmente. A divisão entre *encoder* e *decoder* também faz com que seja possível uma organização clara da experimentação, esta focada na busca das melhores estruturas individualmente e em combinação com seus duais.

Além disso, a inspiração das FCNs em redes concebidas para classificação de imagens faz com que seja trivial o uso de uma ferramenta que tem se mostrado poderosa neste tipo de *dataset* com reduzido número de imagens para treinamento: o *transfer learning*. Mais que o aproveitamento de redes conhecidas, será possível testar e avaliar o uso de pesos decorrentes de treinamentos em outros *datasets* e tarefas.

3.3. Trabalhos Relacionados

A automatização do processamento de imagens sísmicas foi tratada tradicionalmente com técnicas de Visão Computacional, abordagem essa ainda explorada recentemente, como em Wang, Long, AlRegib [9], que usa atributos de texturas e gradiente de texturas (GoT) para sintetizar contornos sísmicos. Porém, esforços têm sido cada vez mais concentrados em técnicas de aprendizado de máquina, inclusive com workshops sobre o assunto sendo organizados em conferências tradicionais da área, como apontado em [10].

Como neste trabalho, FCNs são as arquiteturas mais utilizadas neste tipo de *dataset*, como é o caso, por exemplo em [11], onde são utilizadas técnicas de deep learning para segmentação de fácies sísmicas (multicategóricas) em imagens 3D. Em [12], os autores trabalham com um dos *datasets* aqui estudados e lidam com acréscimos interessantes às FCNs mais comuns, como hipercolunas e *gates* de atenção, alguns dos quais serão explorados ao longo deste trabalho. Há estudos inclusive que mostram o resultado de técnicas de *transfer learning* em imagens sísmicas, como [13]. Por último, vale notar também trabalhos que usam *datasets* parecidos e mesma técnica mas para identificar estruturas diferentes: em [14], os autores usam FCNs para identificar falhas sísmicas.

Sendo assim, pode-se concluir que esta dissertação se alinha com o que há de mais promissor na área de processamento de imagens sísmicas atualmente.

4. Metodologia

Neste capítulo são detalhados os processos experimentais responsáveis pela obtenção de dados (e sua subsequente análise) que suportam as conclusões deste trabalho. É feita primeiramente a descrição dos *datasets* e suas formas de pré processamento. Nas seções seguintes a metodologia de escolha da melhor rede é abordada, esta tendo ênfase na diminuição do espaço de possibilidades por meio da estipulação de um template, fixação de alguns hiperparâmetros e escolha de módulos a serem testados a partir de motivações teóricas e de uso comprovado na literatura. Por fim, outros processos usados para a melhora dos resultados são descritos, incluindo *transfer learning*, criação de exemplos sintéticos e pós-processamento.

4.1. *Datasets*

O primeiro passo no processo de modelagem é a análise dos dados disponíveis. Tipos de dados, como estão apresentados, estatísticas básicas e os objetivos de cada *dataset* foram informações valiosas no planejamento da experimentação.

Dois *datasets* são os focos principais deste trabalho: um obtido através da competição elaborada pelo website Kaggle denominada “TGS Salt Identification Challenge” e outro cedido pela Petrobras no contexto de um projeto elaborado em parceria com a PUC-Rio. O objetivo é que as redes aqui estudadas possam ser avaliadas, em um primeiro momento, num contexto competitivo que permite um dimensionamento de sua qualidade frente a outras redes, para depois serem aplicadas num problema concreto dentro do escopo do projeto.

Suas descrições, incluindo seus critérios de divisão entre treino, teste e validação são descritos a seguir.

4.1.1. *Dataset* Kaggle

Este *dataset* foi disponibilizado na forma de 22.000 imagens de dimensão 101x101 com apenas um canal de cor de tons de cinza com valores de 0 a 255 originalmente, mas normalizados para ficar entre 0 e 1.

Estes *patches* são pertencentes a uma figura maior de toda a captação subterrânea feita em determinada área. Há para 4.000 delas uma máscara correspondente indicando a presença ou não de sal, sendo elas naturalmente usadas como grupo de treino e validação e as 18.000 restantes como grupo de teste. As previsões neste grupo de teste deveriam ser enviadas ao site Kaggle onde são avaliadas de acordo com a métrica explicitadas pelos organizadores da competição (abordada oportunamente na seção 4.10). A Figura 9 exemplifica um desses *patches* com sua respectiva máscara, sendo a porção branca desta referente ao sal presente.

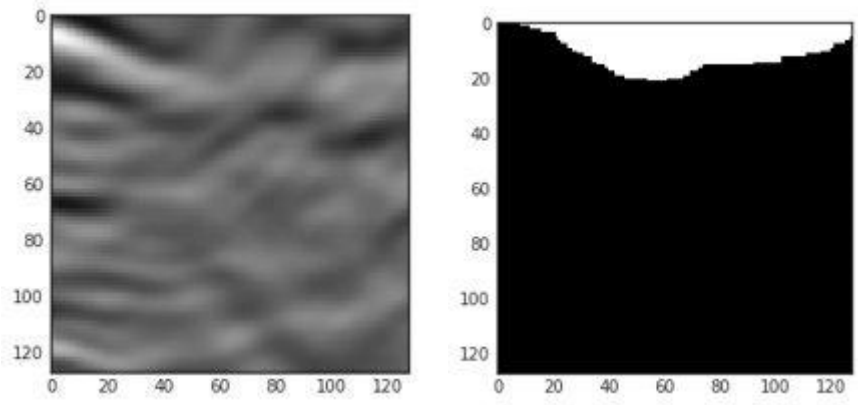


Figura 9 - Exemplo *dataset* Kaggle

As estatísticas mais relevantes referentes a este *dataset* se referem à quantidade do sal em comparação com a totalidade das imagens. Por este motivo, serão apresentados apenas os dados referentes ao grupo de treino/validação, a cujas máscaras se tem acesso. A Figura 10 apresenta o histograma que mostra o percentual de sal em imagem em cada *patch*.

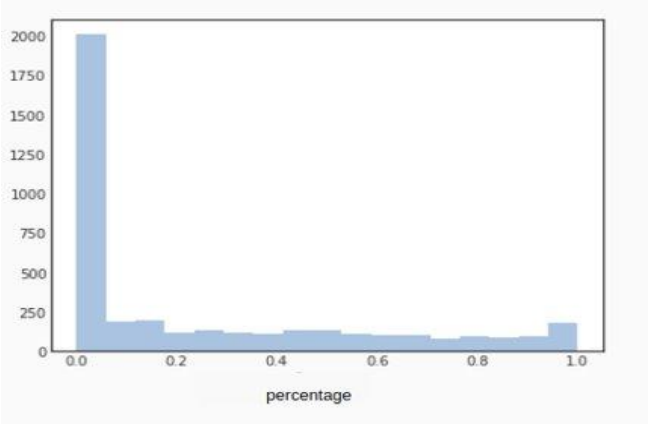


Figura 10: Histograma de porcentagem de sal por *patch* de imagem no *dataset* Kaggle

Este percentual é utilizado na divisão entre grupo de treino e de validação da seguinte forma: tendo separado os *patches* nas 20 classes representadas no histograma acima, a divisão é feita de forma que os dois grupos tenham distribuições de *patches* aproximadamente iguais. Esta foi a forma encontrada para que a seleção do modelo na validação fosse o mais fiel possível em relação ao que foi visto pela rede no treino.

Finalmente, para lidar com o número relativamente baixo de exemplos é usada uma técnica bastante comum de *data augmentation* [46]: o giro (*flip*) horizontal de imagens, como exemplificado na Figura 11. Esta operação é incorporada desde início do processo experimental por se mostrar vantajosa e qualitativamente razoável, já que se espera que propriedades sísmicas sejam invariantes sob reflexões especulares. Giros verticais, por outro lado, poderiam trazer problemas, já que a profundidade é uma variável importante nas propriedades físicas do solo. É importante salientar que o giro é feito somente no grupo de treino.

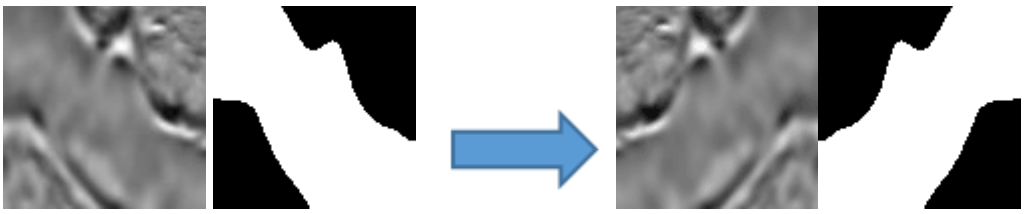


Figura 11 - Exemplo da operação de giro horizontal

No final, tendo separado 800 imagens para validação (20%), são usadas 3200 imagens originais que, sendo aumentadas via giro horizontal, totalizam 6.400 *patches* para treino.

4.1.2. *Dataset* Petrobras

Este *dataset* consiste em imagens geradas artificialmente por um software da Petrobras e se dividem em duas, uma com 1040 x 7760 e outra com 1216 x 6912 pixels de dimensão. Diferentemente do *dataset* Kaggle, este foi disponibilizado em sua plenitude, ou seja, imagens completas dos reservatórios e suas máscaras, abrindo uma miríade de possibilidades em relação à divisão em *patches* e a separação dos grupos de treino, validação e teste. A Figura 12 mostra as duas imagens e suas respectivas máscaras de segmentação.

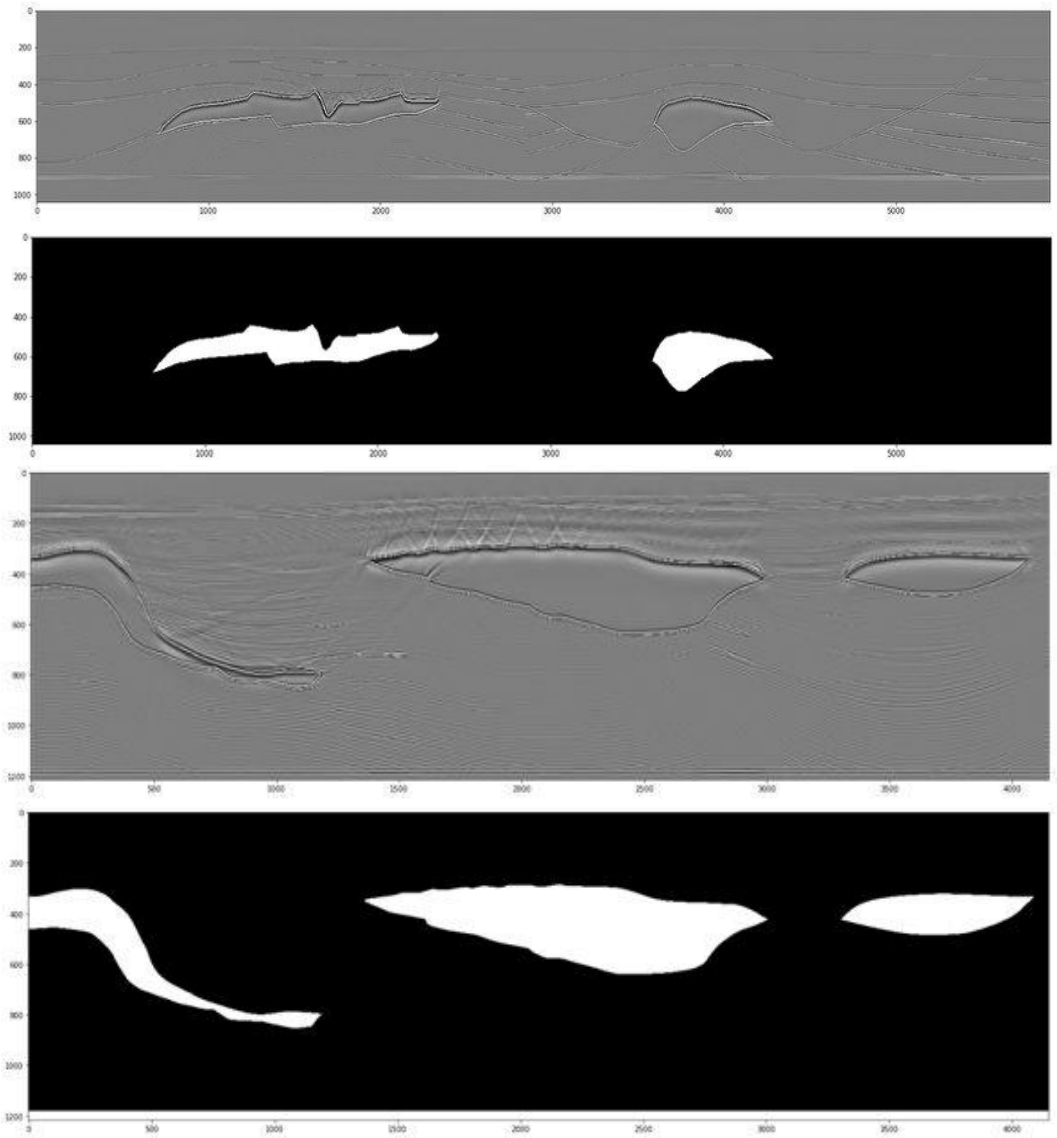


Figura 12 - Dataset Petrobras, com suas imagens e respectivas máscaras de sal

Em relação à divisão de *patches*, são usadas duas formas denominadas *random* e *stride*. Na *random*, um pixel é selecionado dentro da imagem completa para ser o centro do *patch*, sendo necessário apenas escolher o tamanho deste e o número de exemplos que se quer gerar. Além disso, uma *seed* foi determinada para que os experimentos sejam sempre feitos nas mesmas coordenadas e não haja variação nos resultados por essa fonte de aleatoriedade. Já na *stride*, como ilustrada na Figura 13, os *patches* são gerados lado a lado com certo tamanho dado como parâmetro, podendo haver uma sobreposição entre eles de acordo com um outro parâmetro que define a divisão.

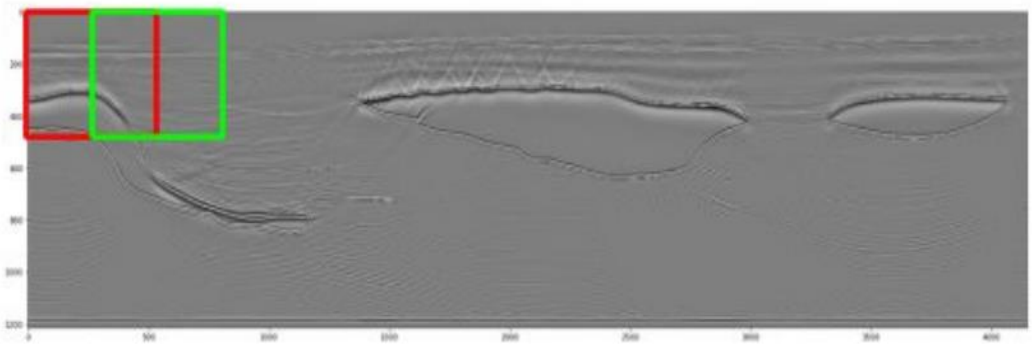


Figura 13 - Modo *stride* de divisão em *patches*

A divisão entre conjuntos de treino e validação é feita da seguinte forma: as imagens são divididas por uma linha em duas partes, separando camadas de sal (os maiores ficando para treino). O objetivo disso é não misturar os dois grupos e fazer com que a validação seja minimamente diferente do treino para incentivar a generalização do modelo e evitar o *overfitting*. As primeiras imagens da Figura 14 foram escolhidas como fontes dos conjuntos de treino/validação.

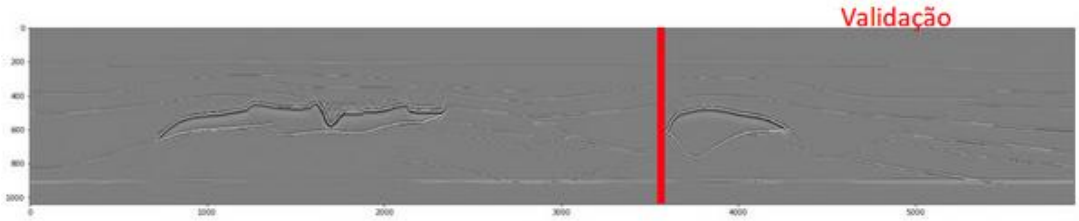


Figura 14 - Método de divisão treino/validação

Os dados foram normalizados para que seus valores fiquem entre 0 e 1 ainda na figura completa, sendo apenas divididos posteriormente. Além de normalizados, foi explorada a opção da aplicação de mapas de cores de forma a transformar a imagem de 1 para 3 canais, este formato sendo mais comum em CNNs para imagens. O mapa de cor utilizado foi o *'seismic'* do pacote matplotlib [55], exemplificado na Figura 15.

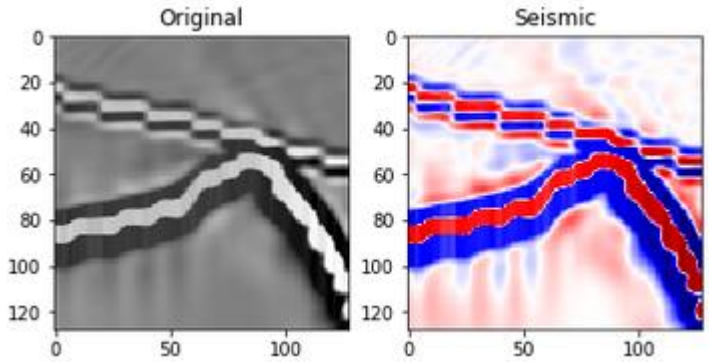


Figura 15 - Colormap “seismic” aplicado ao patch

Para o teste, o segundo conjunto de imagens/máscara da Figura 12 é utilizado. Assim como no *dataset* Kaggle, o *flip* horizontal também é utilizado como forma de *data augmentation* desde o início da experimentação.

4.2. Template das redes

Inspirado no artigo que apresenta a ResNeXt [6], este trabalho apresenta um template modularizado de forma a reduzir o espaço de designs de redes possíveis e, assim, permitir uma experimentação com metodologia clara que nos permita concentrar nos conceitos-chave. Esse template, por sua vez, é inspirado na topologia das redes VGG [26]/ResNets [27], seguindo duas regras: a) todo bloco que produza como *output* mapas de atributos do mesmo tamanho tem os mesmos hiperparâmetros, e b) cada vez que o mapa for reduzido/aumentado por um fator de 2, a espessura (isto é, número de filtros) do bloco é multiplicada/dividida pelo mesmo fator, de forma a manter mais ou menos a mesma complexidade.

Cada um destes blocos tem os seguintes hiperparâmetros: tamanho de filtro e quantidade de filtros. Além disso, cada bloco pode ter um *tipo de convolução*, que poderá trazer seus próprios hiperparâmetros. Camadas de módulos que possam auxiliar na classificação também caracterizam cada bloco (truques), sendo colocadas ao fim de cada um deles, a menos que haja algum motivo para alguma outra composição. Com estas poucas informações, é possível definir inequivocamente cada bloco.

Além disso, os blocos têm a estrutura residual da Resnet [27], módulo este criado para estabilizar redes com muitas camadas por facilitar o fluxo de gradiente no passo de *backpropagation* do algoritmo de aprendizado das CNNs. Como existem várias formas de organizar as ativações e a normalização do batch dentro do bloco Resnet, é seguida a indicação de um excelente estudo feito nessas diversas estruturas [16] pelo chamado bloco "*full pre-activation*", ilustrado na Figura 16.

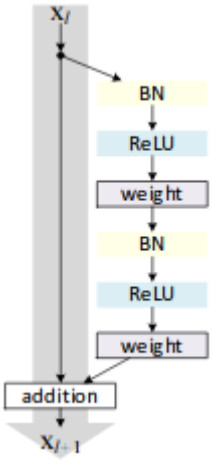


Figura 16 - Bloco residual "full pre-activation"
Fonte: HE K, ZHANG X, REN S, SUN J, 2016 p.8.

Além das convoluções usuais, as redes têm *downsamplings*/*upsamplings*, que são feitos por convoluções/deconvoluções com stride 2.

Camadas que mudem o tamanho dos mapas de atributos dividirão os conjuntos de blocos conforme as regras do template explicitadas acima. Estes conjuntos podem ter uma quantidade variável de blocos, o que será também um hiperparâmetro da rede denominado "profundidades". No fim de cada conjunto pode haver alguma estrutura dentre os módulos elencados na seção 4.4.

Entre o *encoder* e o *decoder* há também uma estrutura com apenas 1 bloco. Na literatura esta estrutura é referida como *bottleneck*. Todas as funções de ativação são ReLU [47], exceto pela última convolução, que tem uma sigmóide, já que valores entre 0 e 1 são esperados como saída ("probabilidade" da presença de sal).

Finalmente, as *skip connections* características da U-Net estão presentes, sempre saindo imediatamente antes de cada *downsample* para logo depois do respectivo *upsample*. Estruturas que se propõem a melhorar as *skip connections* também são exploradas.

A Figura 17 ilustra uma rede com profundidade [2, 2, 2, 1] e número de filtros iniciais 16:

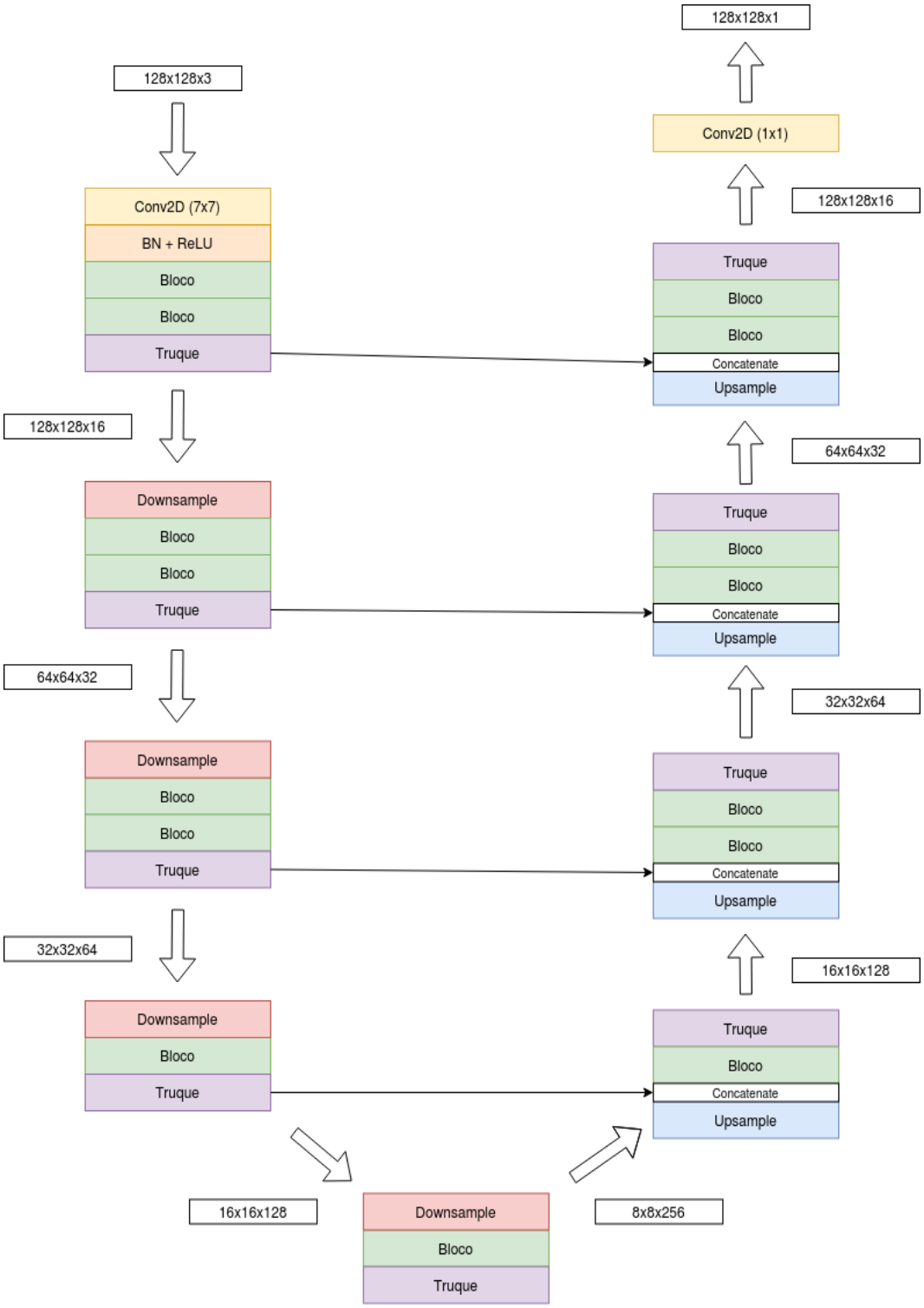


Figura 17 - Exemplo de template utilizado nas redes

4.3. Baseline

Para se ter um comparativo claro e identificar as melhorias trazidas à rede por cada componente adicionado, os primeiros experimentos são realizados em uma rede bastante simples, usada como *baseline*. Para este papel, a rede escolhida foi a U-net, pelo seu design simples e facilmente adaptável a outros tipos de camadas e blocos que são testados neste trabalho. Para se adequar ao template utilizado para a experimentação que será abordado mais à frente, foi adicionado a cada etapa da rede conexões residuais ao estilo Resnet, que proporcionaram melhoras em modelos desde o início do processo experimental. Os hiperparâmetros da rede baseline são:

- Profundidade: [2, 2, 2, 2, 2]
- *Downsample*: *Max pooling*, stride 2
- *Upsample*: Convolução Transposta, stride 2
- Número de filtros iniciais: 16
- Tamanho dos filtros: 3x3
- Módulos: Nenhum
- Convolução: Convolução 2D tradicional

Além disso, um segundo *baseline* é utilizado fugindo do template no que se refere ao *encoder*. Como será visto oportunamente, a técnica de *transfer learning* traz consigo o uso de redes pré-construídas e pré-treinadas no *encoder* para o processo de experimentação. Por seu imediato sucesso desde o começo da experimentação, foi decidido incluir este tipo de rede no *baseline* para que ganhos em seus *decoders* sejam igualmente bem avaliados.

4.4. Incrementação das redes

Como já foi dito, a abordagem para achar a melhor arquitetura de rede possível é a de incrementar sucessivamente as redes que têm os melhores resultados com novas estruturas. Como a quantidade de parâmetros e estruturas de

interesse são bem grandes ao mesmo tempo em que os recursos computacionais são limitados, tem-se a necessidade de limitar o espaço dos designs possíveis.

Nas próximas sub-seções as estruturas que se destacaram ao longo do processo experimental são introduzidas e seu uso é justificado a partir de suas composições e dos desafios que a segmentação semântica do sal apresenta.

4.4.1. *Squeeze-and-Excitation* e suas variações

Como abordado no capítulo 3, as camadas de convoluções das CNNs são compostas de vários filtros que são aplicados nos mapas de *features* (ou canais) de *input*. Esses filtros realizam a operação de convolução individualmente em cada canal e combinam os resultados através da soma de todos os *outputs* de forma igualitária, promovendo efetivamente uma fusão entre os canais. Porém, nem todos os mapas de *features* terão a mesma importância para a realização da tarefa e o módulo *Squeeze-and-Excitation* [17], ilustrado na Figura 18, foi criado como forma de lidar com essa questão.

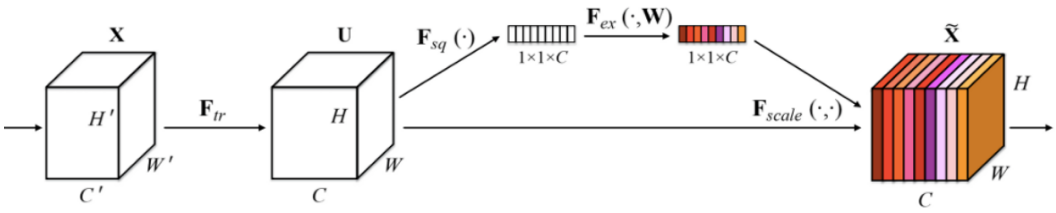


Figura 18 - Ilustração da operação Squeeze-and-Excitation
Fonte: HU J, SHEN L, SUN G, 2018 p.2.

A ideia é que haja uma forma de aprender as importâncias relativas de cada mapa de *features*. Para isso, primeiro se aplica uma operação de média global em cada canal, de forma a capturar as escalas de cada um deles e resultando em um tensor da forma 1 x 1 x C, com C sendo o número de canais do input. Essa é a operação chamada "*squeeze*", pois comprime a informação dos mapas em uma dimensionalidade menor.

O tensor resultando passa então por duas camadas densas, que servirão para aprender os pesos relativos de cada canal. Esses pesos atuarão nos mapas

originais correspondentes como um mecanismo de *gate*, “excitando” cada um deles com sua devida importância aprendida. O código abaixo exemplifica uma implementação desse mecanismo:

```
def se_block (input, channels, ratio):
      x = GlobalAveragePooling2D () (input)
      x = Dense (channels//ratio, activation = ‘relu’) (x)
      x = Dense (channels, activation= ‘sigmoid’) (x)
      output = multiply () ([input, x])
      return output
```

Além de ser responsável por melhorias de até 25% em redes tradicionais na competição da Imagenet em 2017, a ideia do Squeeze-and-Excitation trouxe outros frutos no ano seguinte, dando origem às “*Concurrent space and channel Squeeze-and-Excitation*” [18], ou scSE. Basicamente, a ideia de excitar os diferentes canais foi levada também à componente espacial, como ilustrado na Figura 19.

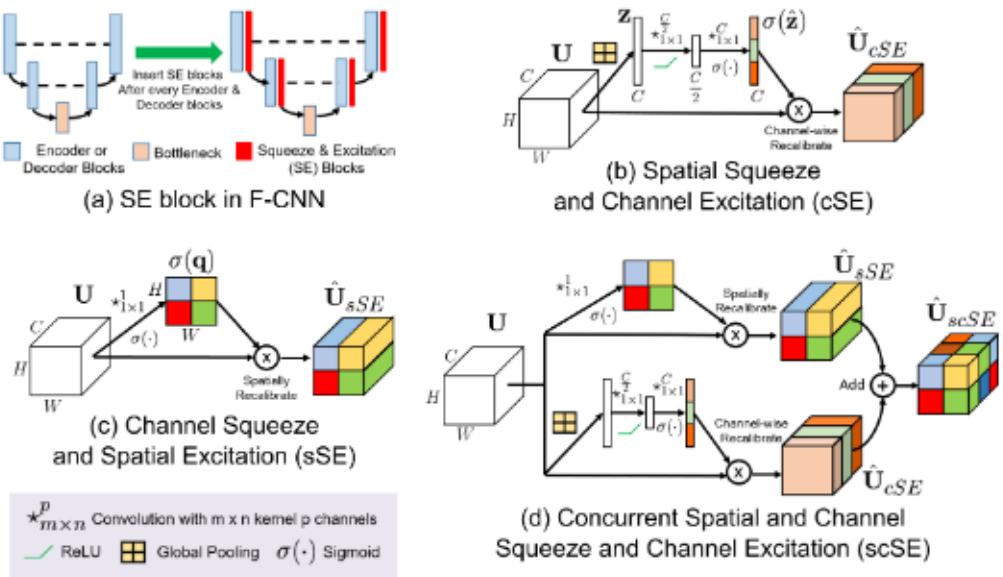


Figura 19 - scSE e suas variações, incluindo o squeeze-and-excitation original (chamada de “Spatial Squeeze and Channel Excitation”)
Fonte: ABHIJIT G R, NAVAB N, WACHINGER C, 2018 p.3

A ideia de tratamentos de canais e espaço separadamente se mostra especialmente atraente para tarefas de segmentação semântica, devido ao desafio

inerente de classificar e localizar simultaneamente já discutido no capítulo anterior.

4.4.2. *Global Attention Upsample*

Seguindo a ideia de ajudar a rede a priorizar melhor os diferentes mapas de *features* geradas ao longo de seu treinamento, a *Global Attention Upsample* (GAU) [31] é proposta para ser aplicada agora às *skip connections*.

Como abordado anteriormente, as *skip connections* tem como principal função alimentar os mapas de *features* do *decoder* com os gerados no *encoder*, de forma a auxiliar a recuperação das informações espaciais da imagem. Desta forma, as GAUs têm como objetivo melhorar este auxílio, fazendo com que a informação do encoder não seja simplesmente anexada, mas também sirva de base para a "excitação" dos canais do decoder em um processo análogo às *Squeeze-and-Excitation*, como observado na Figura 20.

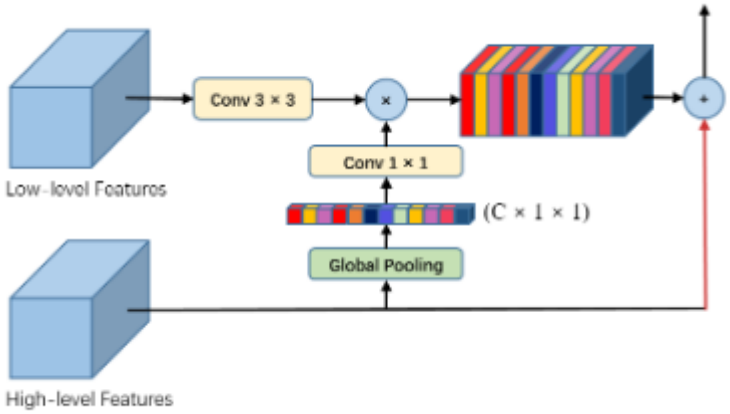


Figura 20 - O módulo *Global Attention Upsample*
Fonte: LI H, XIONG R, AN J, WANG L, 2018 p.5

4.4.3. *Feature Pyramid Attention*

Além de módulos que tenham como proposta a melhoria dos blocos e das *skip connections* das FCNs, também foram testados os que tencionam aperfeiçoar os *bottlenecks*. Dentro dessa categoria se destaca a *Feature Pyramid Attention*

(FPA) [31], que busca obter análise simultânea de várias escalas da imagem, ao mesmo tempo que aplica uma espécie de mecanismo de atenção.

Esta análise de diferentes escalas ao mesmo tempo é realizada com a aplicação de filtros de diferentes tamanhos em mapas de *features* de diferentes dimensões, que serão posteriormente combinados via operações de soma, como mostra a Figura 21. Este processo é importante em razão da já discutida perda da informação de localização dos pixels devido às operações de downsampling em diferentes dimensões da imagem.

Já o mecanismo de atenção é dado pela multiplicação pixel a pixel do mapa de feature do *input* após a passagem por uma convolução 1x1, de forma análoga à sSE da seção 4.4.1.

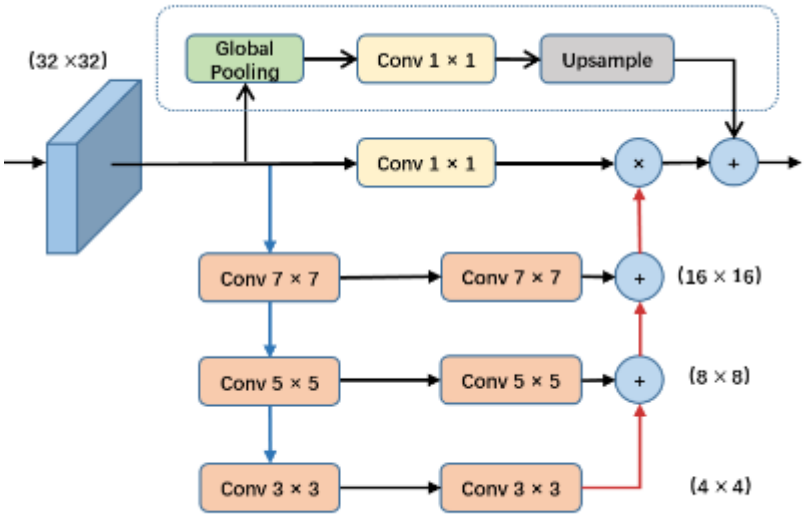


Figura 21 - O módulo Feature Pyramid Attention. As dimensões dos mapas de *features* assinaladas não se referem necessariamente às utilizadas nas redes deste trabalho.
Fonte: LI H, XIONG R, AN J, WANG L, 2018 p.4

A presença do módulo FPA no *bottleneck* não é por acaso [31]. O fato de ser a parte da rede com mapas de *features* de menor resolução permite que filtros de dimensões maiores como o 7x7 sejam aplicados sem grande aumento na carga computacional. Além disso, o módulo se aproveita dos altos RFs dos filtros no *bottleneck* para que as diferentes escalas utilizadas neles sejam referentes a grandes porções da imagem de input, trazendo bastante contexto para o mecanismo de atenção.

4.4.4. Convoluções Separáveis comuns e transpostas

Por fim, uma outra maneira de modificar as redes é através do uso de convoluções diferentes das comuns. A que mais se destacou dentro desta categoria foi a Convolução Separável. Este tipo de convolução teve destaque em seu uso na rede Xception [5], criada em 2017 e que superou a performance de algumas das melhores redes de classificação de imagens da época.

Elas separam a convolução em duas fases referentes aos canais e aos pixels da imagem (Figura 22), fazendo com que ” as correlações entre canais e as correlações espaciais sejam mapeadas separadamente“ [5]. Embora não tenha sido idealizada com este intuito, este tipo de convolução é uma ferramenta valiosa para atacar o desafio de aliar classificação e localização na segmentação semântica, já que criam filtros pixel a pixel na operação de *pointwise convolution*. Isso se confirma com os bons resultados que serão apresentados mais à frente.

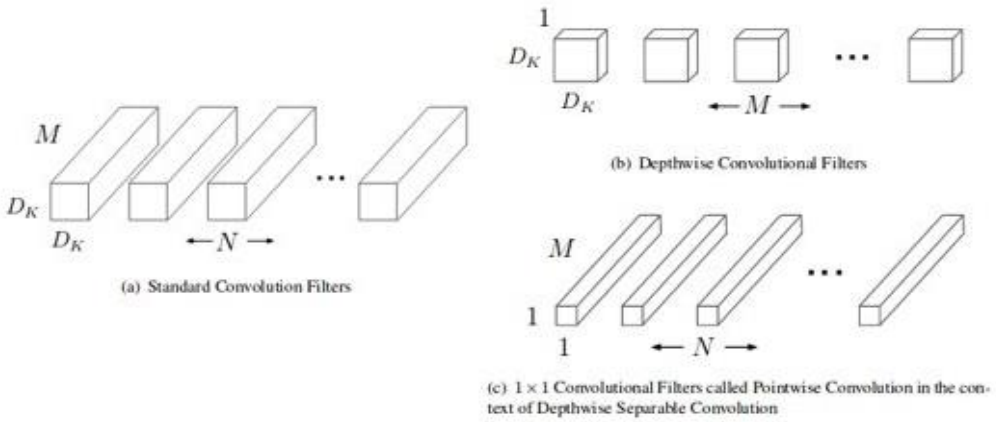


Figura 22 - a) Convolução tradicional, feita em um volume abarcando canais e o tamanho do filtro. b) Convoluções individuais em cada canal: primeiro passo das convoluções separáveis. c) Convoluções 1x1 ao longo dos canais: segundo passo das convoluções separáveis

Fonte: <https://towardsdatascience.com/classifying-skin-lesions-with-convolutional-neural-networks-fc1302c60d54> Acessado em 2 de abril de 2021.

Além da convolução separável comum, foi implementada para este trabalho a versão transposta para ser usada como opção de *upsampling*. Essa versão consiste na substituição das das convoluções *depthwise* (Figura 22-b) comuns por convoluções transpostas [51], mantendo as *pointwise* (Figura 22-c).

4.5. Busca de Hiperparâmetros

Há diversos hiperparâmetros nas redes utilizadas que influenciam diretamente a sua capacidade de aprendizado. A quantidade de filtros utilizada em cada camada, por exemplo, influencia na quantidade total de parâmetros, o que, por sua vez, pode fazer a diferença entre um modelo que generaliza suas previsões e outro com problemas de *overfitting*.

Mesmo com tal importância, ainda é muito difícil com os conhecimentos e técnicas disponíveis atualmente saber no ato de concepção da rede quais os melhores valores a serem utilizados. Embora haja uma estimativa razoável que possa ser feita para início dos experimentos, geralmente é uma boa ideia fazer um ajuste para adequar a rede ao tipo e tamanho de *dataset*, tarefa, objetivos, etc.

Desta forma, em determinados pontos do processo experimental é feita uma busca de hiperparâmetros. Como envolve diversos hiperparâmetros que são interdependentes[3], este processo permite um grande número de combinações de valores possíveis. Isso nos obriga a limitar cada hiperparâmetro a um número reduzido de valores que contemplem uma amplitude razoável de possibilidades, de forma a limitar essa etapa a um tempo razoável. No caso deste trabalho foram utilizados 3 tamanhos e 5 quantidades distintas de filtros convolutivos.

Tendo feito essa limitação de parâmetros, é utilizada a técnica conhecida como *grid search* [48] para realizar a busca de hiperparâmetros. A técnica consiste em tentar as diferentes combinações de parâmetros disponíveis e selecionar a que produz melhores resultados nas métricas de avaliação.

4.6. *Transfer learning* e redes pré-treinadas

Assim como a escolha da arquitetura geral das redes usadas neste trabalho são influenciadas pela tarefa de classificação de imagens, o próprio treinamento das redes também pode se aproveitar de tarefas mais bem estudadas e *datasets* mais extensos. Para isso, é empregada a técnica de *transfer learning* [29].

A técnica se refere, de forma ampla, a "qualquer estratégia que usa o conhecimento adquirido resolvendo um problema para subsequentemente resolver

[3] Tanto quantidade quanto tamanho de filtros influenciam em número de parâmetros, por exemplo, embora tenham também outros impactos gerais na rede.

um problema diferente". Uma caracterização formal da técnica e a descrição de diferentes formas que ela pode tomar podem ser encontradas em [30], mas neste trabalho ela tem um significado bem simples: pré-treinamento da rede utilizada (e, mais especificamente, parte dela) em outros *datasets* e tarefas, de forma que a transferência do conhecimento obtido seja feita através dos pesos aprendidos nesta etapa.

Este recurso se torna especialmente útil nos casos em que se tem acesso limitado a dados e quando o tempo de treinamento da rede é relativamente longo. Isso porque o *transfer learning* permite o aprendizado de filtros com complexidades que talvez não fossem possíveis de se obter em *datasets* pequenos, além de fazer com que a rede seja inicializada em um estágio em que certos filtros de *features* essenciais à tarefa podem já ter sido obtidos, adiantando o processo. Vale apontar que ambos os casos se aplicam aos experimentos realizados neste trabalho.

Diante dos recursos disponíveis, foi feita a escolha de se utilizar do *transfer learning* apenas no decoder das FCNs utilizadas. Isso se justifica pela disposição de diversas redes pré-treinadas em classificação de imagens no pacote Keras utilizado na elaboração dos experimentos. Porém, conceitualmente nada impede que o *transfer learning* seja feito em toda a rede, como é o caso em [28], que utiliza o *dataset* PACAL-VOC de segmentação semântica para pré-treinar tanto *encoder* como decoder. Por ser custoso e demorado, esse *transfer learning* completo não foi realizado.

Sendo assim, o pré-processamento se dá em redes cujas arquiteturas se demonstraram bastante eficazes em tarefas de classificação de imagem. Isso se justifica por tais redes compartilharem a tarefa do encoder das FCNs de extrair *features* úteis da imagem de *input*.

Além disso, elas são pré-treinadas no *dataset* Imagenet [7], que consiste em mais de 14 milhões de fotografias divididas em mais de 20 mil categorias e 1 milhão de bounding-boxes anotados manualmente via plataformas de *crowdsourcing*.

O Imagenet é utilizado em competições anuais chamadas "I*magenet Large Scale Visual Recognition Challenge*" que têm por objetivo incentivar técnicas de computação visual ao mesmo tempo que provém *benchmarks* para diferentes tarefas como classificação de imagens, localização de objeto e detecção de

objetos. Para cada uma dessas competições são usados determinados subconjuntos da base, com datasets de aproximadamente 1 milhão de imagens divididas em 1000 classes não-sobrepostas.

Dentre as redes disponíveis com o pré-processamento na Imagenet, duas se destacaram tanto pelos resultados quanto pela compatibilidade dos conceitos em que se baseiam com a tarefa a que esse trabalho se dedica: são elas a Xception [5] e a ResNeXt [6].

Como dito anteriormente, a Xception usa convoluções separáveis, cujas propriedades de interesse estão descritas na seção 4.4.4. Já a ResNeXt data de 2016 e obteve o segundo lugar na competição da Imagenet daquele ano trazendo o conceito de cardinalidade, ilustrada na estrutura apresentada na Figura 23.

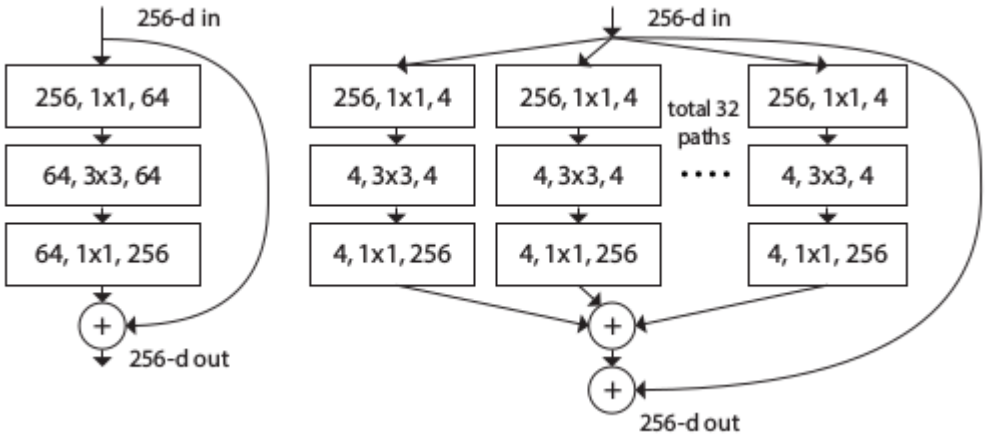


Figura 23 - Bloco da ResNeXt (à direita) que exemplifica a cardinalidade, contrastando com um bloco ResNet tradicional (à esquerda)
Fonte: XIE S, GIRSHIK R, DOLLÁR P, TU Z, HE K, 2017 p.1

O artigo que apresenta a rede [6] justifica a criação da estrutura acima como uma forma de combinar a simplicidade de redes como a VGG [26] e a ResNet [27], cujas arquiteturas consistem na repetição de blocos semelhantes, com a possibilidade de customização e ajuste fino de arquiteturas possibilitadas pelos modelos Inception [19]. Isso é alcançado com um design semelhante ao dos blocos Inception, porém com os diversos caminhos de convoluções tendo a mesma topologia. A customização fica por conta do novo hiperparâmetro da *cardinalidade*, que indica quantos caminhos existirão por bloco.

Além da proposição do modelo, o artigo mostra que o aumento da complexidade dos modelos produz resultados diferentes se feito via aumento de número de filtros (largura da rede) ou da cardinalidade, tendo este último os melhores valores. Sendo assim, a calibragem da rede levando em conta este novo hiperparâmetro tem a possibilidade de levar a redes mais eficientes.

Um dos grandes desafios da tarefa proposta é a necessidade de se usar redes relativamente complexas ao mesmo tempo em que os dados são, por sua intrínseca dificuldade de anotação, escassos. Diante disso, o problema de *overfitting* se torna uma ameaça constante, o que torna o eficiente uso de parâmetros demonstrado pela ResNext atraente para os objetivos deste trabalho.

Estas redes pré-treinadas, como já foi dito, servem de *encoder* para a rede final, sendo complementadas com o *bottleneck* e o decoder no estilo U-net. *Skip connections* também serão utilizadas, usando o critério de se utilizar sempre a camada anterior a um *downsample* conectada com a contraparte no decoder após cada *upsample*. A Figura 24 exemplifica este processo no caso da rede Xception.

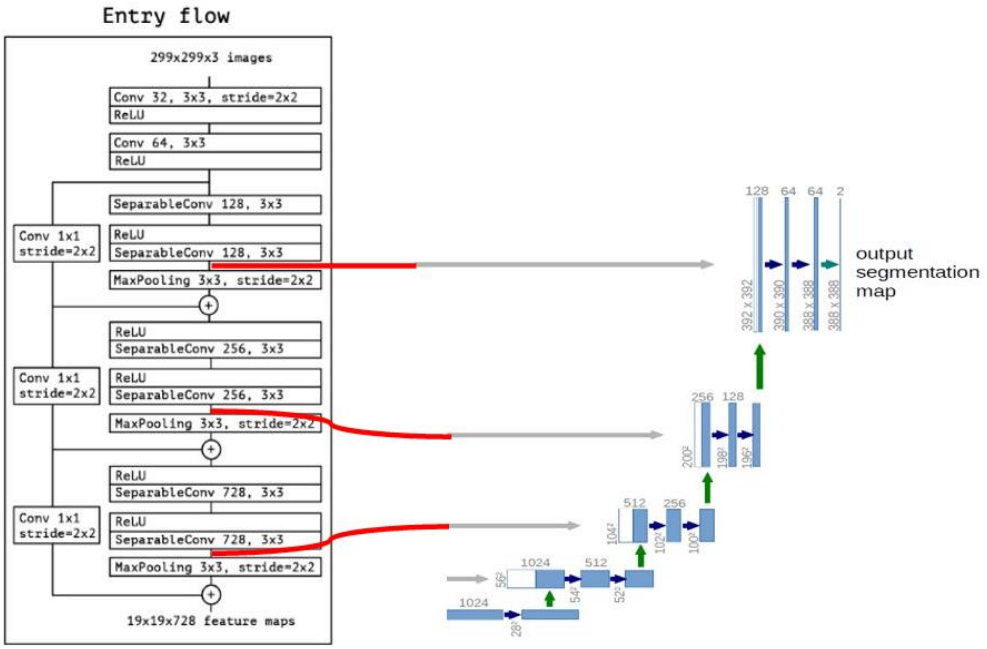


Figura 24 - Combinação da rede Xception com um decoder simples da U-net. As skip connections sempre estarão localizadas antes de um downsampling

4.7. Criação de dados sintéticos

Para além do aumento de dados via giro horizontal dos *patches*, é utilizada uma segunda forma de criação de novos exemplos sem a modificação dos já existentes. A partir de um trabalho realizado no mesmo contexto de colaboração entre a PUC-Rio e a Petrobras [33] foram criados exemplos sintéticos totalmente novos a partir dos conjuntos de treino.

Em linhas gerais, os dados sintéticos são gerados em duas etapas: a primeira gera máscaras sintéticas a partir de Autoencoders Variacionais [52]. A segunda usa um modelo chamado "*Conditional Normalizing Flow*" [53] para aprender uma função que é capaz de transformar as imagens sísmicas em máscaras de sal e sua respectiva função inversa. Com essa função em mãos, é possível gerar as imagens sísmicas sintéticas a partir de cada uma das máscaras sintéticas, de forma a criar um novo *dataset* para auxiliar o treinamento das redes utilizadas nesta dissertação.

A aplicação no *dataset* Petrobras foi análoga ao artigo referenciado. Embora o estudo lide com *patches* de tamanho 64 x 64 pixels, para este trabalho foi feito um novo treinamento para *patches* de 128 x 128 pixels, já que estes foram os que resultaram nos melhores modelos, como será visto mais adiante. Além disso, eles foram criados com o colormap "*seismic*", este também responsável pelos melhores resultados. O número de *patches* adicionado ao treino foi o mesmo utilizado pelos autores do artigo original: 160. Embora pareça um número pequeno, todas as respectivas máscaras deste conjunto sintético contém sal, o que significa um acréscimo significativo para esta classe de imagem. A Figura 25 traz exemplos de *patches* sintéticos gerados para o *dataset* Petrobras.

Figura 25 - Exemplos de *patches* sintéticos para o dataset Petrobras

Para o *dataset* Kaggle, decidiu-se uma abordagem diferente devido às limitações dos próprios dados (já divididos em *patches* e com muitas máscaras

vazias) e dos recursos (são treinos que levam até semanas e com necessidade de ajustes[4]). Primeiramente, foram preteridos os *patches* que mostrassem pouca fronteira entre as classes "sal" e "não-sal", já que estas transições são as estruturas de maior interesse. Isso foi feito descartando os *patches* com mais de 90% ou menos de 10% de sal.

Outros problemas enfrentados foram o tamanho dos *patches* (que aumentam bastante o tempo de processamento) e a pouca variedade entre eles. Estes dois problemas foram solucionados pela criação de 4 *patches* diferentes a partir de cada um dos selecionados, estes referentes aos quatro cantos da figura recortados em tamanho 64 x 64 pixels, como mostra a Figura 26.

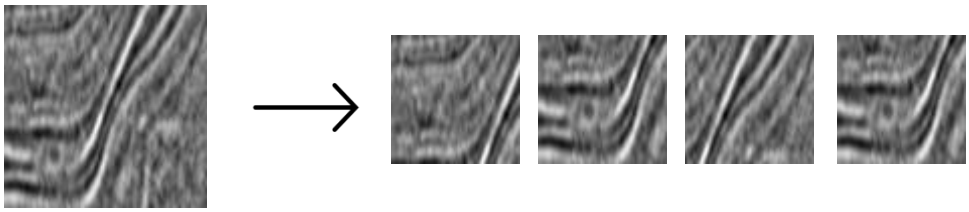


Figura 26 - Recorte dos cantos 64 x 64 pixels

Os *patches* sintéticos 64 x 64 foram aumentados via interpolação para o tamanho 128 x 128 usado nos experimentos com este *dataset*. Foram geradas um total de 2000 imagens com suas respectivas máscaras e, embora tenham sido testados diversas quantidades a serem adicionadas ao treino, os resultados melhoraram apenas para redes do baseline.

4.8. Pós-processamento

Para além da predição das máscaras de sal pelas redes, é possível melhorar os resultados a partir do uso inteligente dessas predições e de algoritmos simples que "retocam" essas máscaras a partir de suposições razoáveis sobre as estruturas salinas. Estes métodos usam *patches* sobrepostos no grupo de teste e análises em figuras completas e por isso são aplicados somente no *dataset* Petrobras.

[4] A geração de *patches* sintéticos foi feita no ambiente Google Colab Pro, que na época da elaboração deste trabalho conta com processadores Intel(R) Xeon(R) 2.30 GHz, 32Gb de memória e GPUs do tipo Tesla P100-PCIE-16Gb [49]

4.8.1. *Patches* sobrepostos e comitês de máscaras

Assim como nos conjuntos de treino e validação, é possível escolher como dividir o grupo de teste desde que a imagem sísmica completa esteja disponível. Neste caso, os *patches* de teste podem ser gerados com sobreposições, de forma que o mesmo pixel seja predito com diferentes contextos, tendo, portanto, maiores chances de se obter a resposta correta.

A partir deste princípio, o conjunto de teste é dividido em *patches* com diferentes graus de sobreposição antes de serem processados pelo preditor, o que determinará quantas vezes cada pixel será avaliado em diferentes contextos. Estas diferentes máscaras preditas servirão como comitês para a decisão da classificação de cada pixel: ele será classificado como "sal" se tiver metade ou mais das máscaras indicando essa previsão. A Figura 27 ilustra como é feita essa sobreposição de forma a gerar 4 máscaras de predição para cada pixel, o que demanda pelo menos 2 delas prevendo sal para que este seja o veredito final.

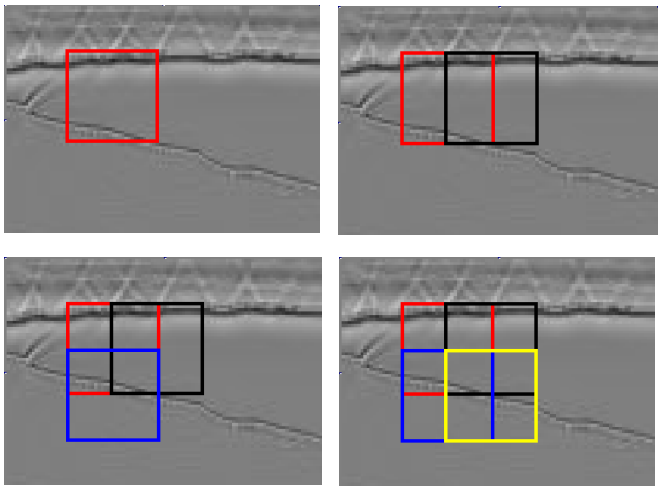


Figura 27 - Divisão em *patches* sobrepostos do conjunto de teste. Com essa sobreposição de 50% *patch*, cada pixel será avaliado 4 vezes.

4.8.2. Erosão e dilatação

Como abordado no capítulo 2, a natureza do processo físico usado na aquisição de dados sísmicos favorece a identificação de fronteiras entre diferentes materiais, sendo mais complicada a diferenciação das estruturas internas das

rochas. Sendo assim, algumas das classificações dos preditores treinados tendem a identificar bem as fronteiras e ao mesmo tempo cometer pequenos erros em suas vizinhanças, com partes internas das estruturas salinas acusando ausência de sal e vice-versa, como exemplificado na Figura 28.

Figura 28 - Exemplos de falhas próximas a fronteiras das estruturas salinas

A partir da hipótese de que as estruturas salinas a serem identificadas são, em sua maioria, sólidas, é proposto o uso de algoritmos de erosão e dilatação para a correção das falhas próximas a fronteiras e eventuais resíduos de sal incorretamente classificados. Estes algoritmos funcionam de formas análogas à convolução: um filtro é passado ao longo da imagem aplicando operações que dependem da seção da imagem delimitada pelo tamanho deste filtro.

Para a erosão, o pixel central será classificado como ‘1’ (ou “sal”) apenas se toda a vizinhança determinada pelo filtro também for ‘1’, caso contrário será erodida (classificada como ‘0’, ou “não-sal”). A dilatação será a operação oposta, com o pixel central sendo classificado como ‘1’ se pelo menos 1 pixel na vizinhança tiver o valor ‘1’.

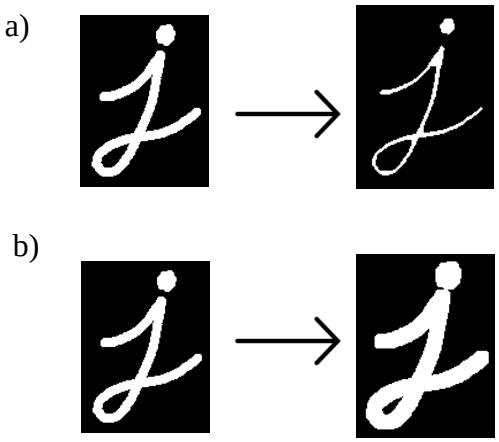


Figura 29 - a) Exemplo da operação de erosão. b) Exemplo da operação de dilatação
Fonte: < https://ai-pool.com/d/erosion-and-dilation-in-tensorflow> Acessado em 2 de abril de 2021

Esses algoritmos foram aplicados em sucessão de forma a resolver as falhas. Primeiramente, a operação de erosão foi usada por 7 iterações de forma a eliminar pequenos resíduos de sal erroneamente classificados na máscara. Em segundo lugar, foi aplicado a dilatação por 25 iterações, para preencher os espaços dentro das estruturas salinas. A erosão então é aplicada novamente o mesmo número de vezes da dilatação para retirar as eventuais dilatações excessivas da segunda etapa. Estes parâmetros para dilatação e erosão foram determinados experimentalmente por tentativa e erro e usaram filtros de dimensões 3 x 3 pixels.

4.9. Treinamento

O treinamento da rede é feito em no máximo 100 épocas, sendo interrompido se a função de perda da validação não cair por 15 épocas. Estes números foram escolhidos com base no histórico de experimentos, que mostrou decaimentos da função de perda durando até 80 épocas em algumas arquiteturas. Foram adicionadas 20 épocas a mais e um dispositivo de parada antecipada para garantir o decaimento integral em quaisquer arquiteturas e configurações.

É feito o embaralhamento dos patches a cada época e há redução da taxa de aprendizado se a função de perda não baixar por 5 épocas consecutivas. O modelo escolhido é o que obteve o melhor resultado na métrica escolhida no grupo de validação e o otimizador escolhido foi o RMSprop [20] com taxa de aprendizado inicial fixada em 0.001.

Para a função de perda foi escolhida a soma entre a entropia cruzada binária e a *dice loss* [21], esta feita sob design para mimetizar o comportamento da métrica IoU de forma diferenciável. Os gráficos na Figura 30 mostram como de fato existe uma clara correlação entre os valores da função de perda e o IoU nas curvas de aprendizado de modelo.

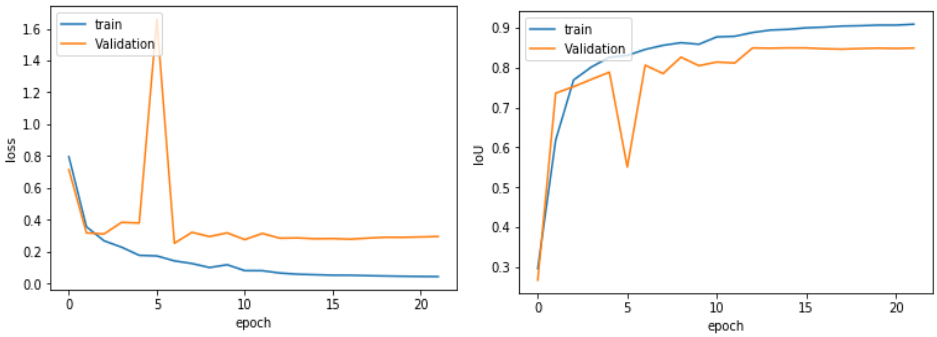


Figura 30 - Gráficos da função de perda e do IoU nos conjuntos de treino e validação em função das épocas do treinamento das redes neurais

Quanto à composição da função de perda, a soma simples entre duas funções diferentes sem nenhum tipo de normalização se justifica pela escala semelhante dos valores da entropia cruzada binária e a *dice loss* nos treinamentos das redes utilizadas neste trabalho. A Figura 31 mostra como as duas funções se reforçam com comportamentos semelhantes, com as pequenas diferenças sendo responsáveis por pequenos ajustes na função final. Esta composição gerou melhora nos resultados durante as fases preliminares do processo experimental e por isso foi usada como padrão em experimentos posteriores.

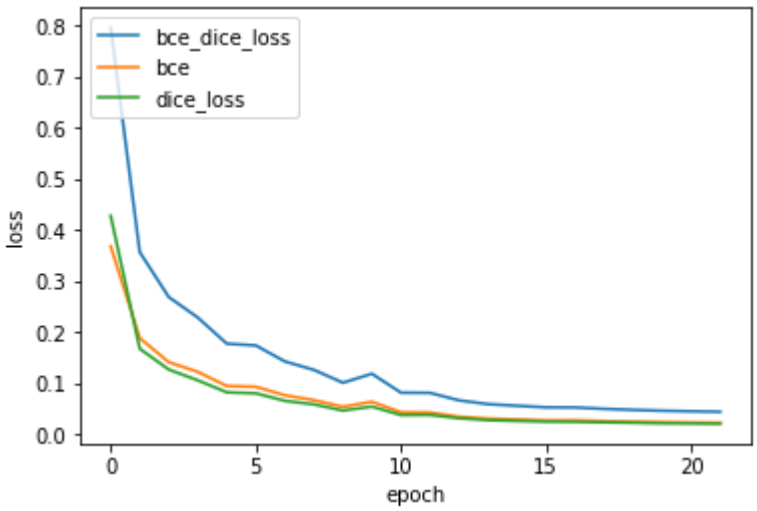


Figura 31 - Composição da função de perda utilizada nos experimentos

4.10. Métricas e Avaliação

Embora o resultado final da previsão nas imagens seja binária em cada pixel (sal e não-sal), as redes construídas para este trabalho apresentam na sua saída valores entre 0 e 1, estes representando o grau de certeza da presença de sal

em cada localidade do input. Sendo assim, o resultado final é obtido somente após a seguinte avaliação: *a partir de que valor dado pela rede irá configurar uma previsão de sal?*

Esta resposta é dada a partir de uma análise de limiares no grupo de validação: após a escolha do modelo a ser utilizado[5], são testados diversos valores de limiares acima dos quais a presença de sal será confirmada. Estes diferentes resultados finais são confrontados com a resposta real e o limiar referente ao melhor deles será o utilizado para avaliar o grupo de teste. A Figura 32 traz o gráfico da métrica de avaliação no grupo de validação pelo threshold escolhido e a tradução desta escolha nas predições efetivas das máscaras.

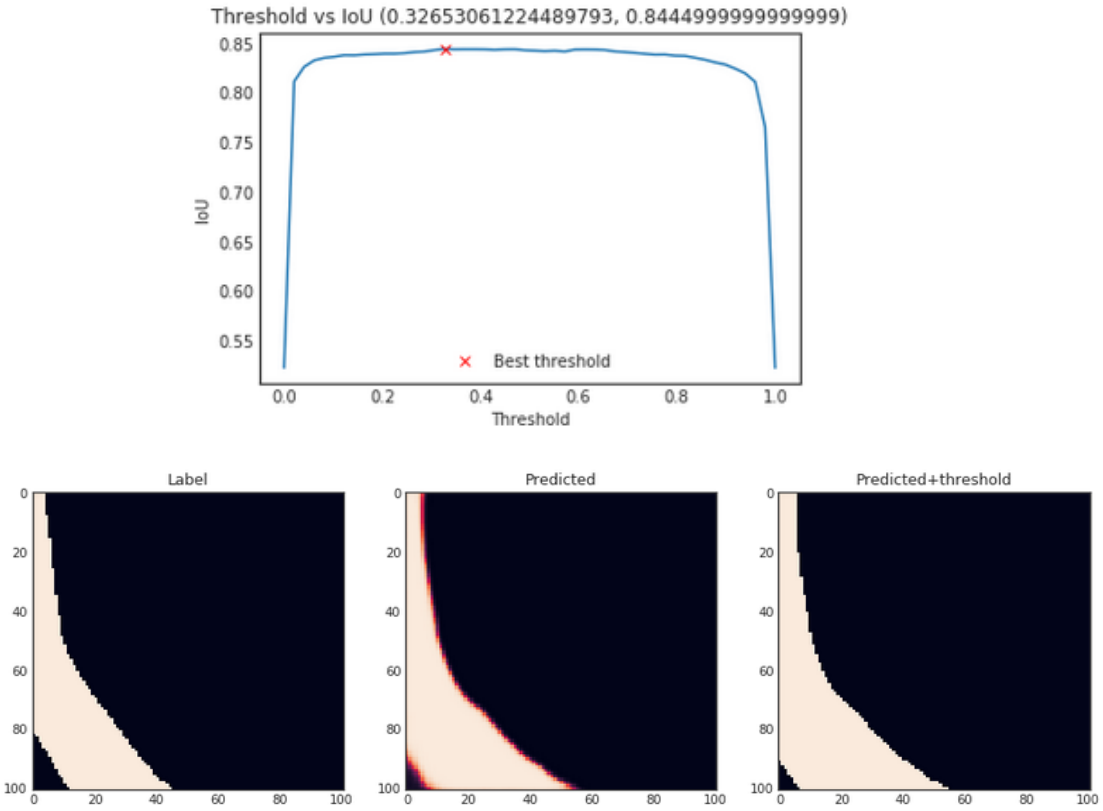
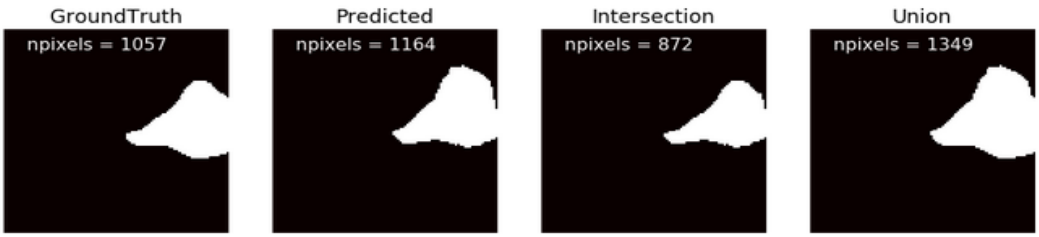


Figura 32 - Análise de limiar e seu efeito na predição final

[5] O modelo escolhido será o da época que obtiver o melhor resultado na avaliação no grupo de validação. Para esta seleção, foi fixado um valor de limiar de 0,4 baseado nos melhores valores observados ao longo de muitas experimentações. A análise de limiar completa não foi feita ao longo de todo o treinamento pela ausência de melhorias que justificassem o aumento do custo computacional.

Para realizar todas as avaliações mencionadas no parágrafo anterior foi utilizada como base[6] a métrica *Intersection over Union*, mais conhecida como **IoU**. Ela é uma medida padrão para problemas de segmentação e consiste na razão entre a interseção e a união dos conjuntos de pontos previstos como sal na máscara predita e na máscara verdadeira, tendo o valor máximo de 1. O exemplo mostrado na Figura 33 ajuda a visualizar este cálculo. Para o caso de máscara vazia, ou seja, sem nenhum sal, a predição ganha pontuação máxima se for igualmente vazia e zerada se obtiver pelo menos um pixel de sal. Os *patches* são avaliados individualmente e a média será o resultado final.



$$IoU(A,B) = \frac{A \cap B}{A \cup B} = \frac{872}{1349} = 0.6464$$

Figura 33 - Ilustração da métrica IoU.

Fonte: https://www.kaggle.com/pestipeti/explanation-of-scoring-metric

[6] O métrica adotada pelo nos experimentos com o dataset Kaggle foi a mesma utilizada na avaliação dos modelos enviados ao site de forma a facilitar comparações com redes em seu *leaderboard.* Ela consiste em atribuir 0.1 ponto para cada 0.05 que o IoU ultrapassar de 0.5. Por exemplo, um IoU de 0.64 terá como pontuação 0.3, enquanto um IoU de 0.96 terá a pontuação 1. Mais detalhes disponíveis em [37]

5. Experimentos

Neste capítulo são avaliadas não apenas as arquiteturas e processamentos de imagem escolhidos, mas também alguns dos passos intermediários, de forma a entender que tipos de contribuições cada processo dá ao resultado final. Essas avaliações são feitas em ambos os *datasets* utilizados neste trabalho, com exceção da última seção que se aplica apenas ao *dataset* Petrobras. Todos os resultados são avaliados na métrica IoU introduzida na seção 4.10, com exceção do resultado final Kaggle, que utiliza a variação da IoU usada na avaliação do próprio site.

Primeiramente são apresentados os resultados das redes que serviram de baseline, bem como os primeiros ajustes ao pré-processamento das imagens, que permanecem presentes no resto do processo experimental. Desta forma, os modelos de baseline utilizados neste trabalho são baseados na arquitetura U-Net, como descrita na seção 4.2.

Em seguida são reportados o resultado das diferentes arquiteturas obtidas a partir do acréscimo de diferentes módulos e convoluções, sucedido pelo ajuste de parâmetros das redes escolhidas como as mais efetivas para cada *dataset*.

A penúltima seção mostra o resultado da geração de exemplos sintéticos a partir das redes SegFlow [33].

Finalmente, apresenta-se o resultado da aplicação das técnicas de pós-processamento introduzidas na seção 4.8 no *dataset* Petrobras e ilustra-se graficamente resultado final no conjunto de teste.

5.1. Baseline e ajuste de pré-processamento

Devido às motivações apresentadas no na seção 4.3, essa seção apresenta duas arquiteturas de redes neurais usadas para treinar os modelos utilizados como *baseline* para este trabalho: uma seguindo o template da U-Net pura e uma segunda utilizando a técnica de *Transfer Learning*, ou seja, combinando a rede base com uma rede pré-treinada como *encoder*. As Tabelas 1 e 2 trazem os resultados dos experimentos que mostram o ganho imediato com o uso do *Transfer Learning* e justificam o uso dessa técnica desde o início do processo de experimentação. No *dataset* Petrobras as redes U-Net combinadas com a ResNeXt

(chamadas aqui de U-ResNeXt) obtiveram os melhores resultados e são usadas como base de experimentação, enquanto a combinação da U-Net com Xception (U-Xception) faz esse papel para o *dataset* Kaggle.

Dataset Petrobras:

Rede	**IoU on Validation**
U-Net	82.77
U-ResNeXt	81.26
U-ResNeXt + *transfer learning*	91.86

Tabela 1 - Resultados da técnica de *transfer learning* no *dataset* Kaggle

Dataset Kaggle:

Rede	IoU on Validation
U-Net	69.90
U-Xception	68.38
U-Xception+ *transfer learning*	81.32

Tabela 2 - Resultados da técnica de *transfer learning* no *dataset* Kaggle

É interessante perceber a importância do pré-treinamento no *dataset* Imagenet para a melhora do resultado, já que apenas o uso das redes Xception e ResNeXt no encoder não foram o suficiente para ganhos significativos.

Além disso, nesta etapa são escolhidos os pré-processamentos a serem utilizados para o resto do processo experimental. Enquanto no *dataset* Petrobras são avaliados o uso ou não do mapa de cores e o método de obtenção dos *patches*, apenas o mapa de cores é avaliado no Kaggle.

Dataset Petrobras:

	U-Net	U-ResNeXt
Sem mapa de cor	83.62	88.19
Mapa de cor 'Seismic'	84.85	**90.43**

Tabela 3 - Avaliação do uso do mapa de cores no *dataset* Petrobras

	# *Patches* treino	U-Net	U-ResNeXt
Sem sobreposição	480	84.85	90.43
Sobreposição 50%	1920	82.77	**91.86**
Sobreposição 75%	7680	80.29	88.41

Tabela 4 - Resultados dos diferentes graus de sobreposição dos *patches* do *dataset* Petrobras em sua criação no modo *stride.*

Nº de *patches* gerados ao acaso	U-Net	U-ResNeXt
480	82.25	89.65
1920	84.18	91.76
7680	83.56	**92.12**

Tabela 5 - Resultado da geração aleatória de *patches* dependendo do número de *patches* criados no *dataset* Petrobras. Os números de *patches* foram escolhidos de forma a se obter uma comparação justa com os *patches* no modo *stride.*

Como a rede U-ResNeXt apresentou visível superioridade, optou-se por avançar o processo experimental com os parâmetros de pré-processamento que maximizam os resultados nela. Desta forma, temos como padrão no processo experimental o uso do mapa de cores 'seismic' e a divisão de *patches* aleatória com 7680 *patches*, além da aumentação de dados com o giro horizontal presente desde o início.

Com essa configuração, tem-se um IoU baseline de 83.56 para U-Net e 92.12 com a U-ResNeXt.

Dataset Kaggle:

	U-Net	U-Xception
None	73.56	79.89
Seismic	70.82	77.95

Tabela 6 - Resultados do uso do mapa de cores no *dataset* Kaggle

Ao contrário do observado no caso Petrobras, os experimentos com mapa de cores no *dataset* Kaggle não apresentou melhoras. Sendo assim, obtém-se um baseline de 73.56 na U-Net e 79.89 na U-Xception.

5.2. Resultados da escolha das redes

Com as arquiteturas de base definidas, o próximo passo será incrementá-las com diferentes estruturas e suas combinações de forma a obter o melhor resultado possível. Nesta seção serão apresentados os módulos testados que se destacaram na experimentação, bem como as combinações que apresentaram alguma melhora.

Durante a pesquisa foram tentadas diversas formas de construção de *encoders* que pudessem competir com os pré-treinados na Imagenet. Embora alguns deles (com destaque do uso do módulo Squeeze-and-Excitation) tenham melhorado o decoder da U-Net, nenhum chegou perto das redes clássicas já prontas em termos de métrica IoU. Sendo assim, esta seção reportará apenas melhorias no decoder da U-ResNeXt e da U-Xception para a predição nos *datasets* Petrobras e Kaggle, respectivamente.

Dataset Petrobras:

Rede	IoU
U-ResNeXt	92.12
+ cSE	90.16
+ scSe	89.74
+ FPA	90.32
+ GAU	91.52

Tabela 7 - Avaliação do uso de diferentes módulos no decoder da U-ResNeXt aplicada ao *dataset* Petrobras

Dataset Kaggle:

Rede	IoU
U-Xception	80.12
+ cSE	81.69
+ scSe	80.77
+ FPA	81.27
+ GAU	80.82
+ cSE + Convolução transposta separável	82.15

Tabela 8 - Avaliação do uso de diferentes módulos no decoder da U-Xception aplicada ao *dataset* Kaggle

A análise destes resultados mostra que, enquanto no *dataset* Petrobras a versão mais simples da rede teve performance melhor que suas variações, as redes aplicadas ao *dataset* Kaggle obtiveram melhorias com o uso de módulos extras mais complexos, inclusive com composição de diferentes estruturas.

Estes resultados podem estar relacionados com as diferentes complexidades dos diferentes conjuntos de dados, com o *dataset* Petrobras apresentando menos detalhes nas imagens sísmicas.

5.3. Procura de hiperparâmetros

Nesta seção são apresentados os resultados da busca de parâmetros de quantidade de filtros usados no primeiro bloco e do tamanho dos filtros usados nas camadas convolutivas. Como especificado na seção 4.3 relativa aos baselines, as configurações iniciais usadas são 16 filtros no primeiro bloco e filtros de dimensão 3 x 3. As redes utilizadas nesse experimento são as escolhidas na seção anterior.

Dataset Petrobras

Qtd Filtros/Tamanho filtro	(3, 3)	(5, 5)	(7, 7)
8	90.04	91.49	91.75
10	89.38	91.14	88.59
16	92.12	**92.25**	89.44
20	89.60	89.79	90.61
32	88.88	89.12	87.98

Tabela 9 - Resultado da busca de parâmetros na rede escolhida para o *dataset* Petrobras

Dataset Kaggle

Qtd Filtros/Tamanho filtro	(3, 3)	(5, 5)	(7, 7)
8	81.56	80.25	80.19
10	82.05	80.89	80.65
16	82.15	82.17	80.77
20	**82.75**	81.98	80.56
32	81.68	81.32	81.23

Tabela 10 - Resultado da busca de parâmetros na rede escolhida para o *dataset* Kaggle

É interessante observar a melhora dos resultados no *dataset* Petrobras com o uso de filtros maiores no decoder. Além dessa mudança, foi feito apenas um acréscimo no número de filtros usados nos experimentos do *dataset* Kaggle. O

fato de aumentos ainda maiores no número de filtros não resultar em melhores pontuações geralmente é relacionado com o *overfitting*, que acaba por ser um dos maiores desafios deste tipo de *dataset* devido à reduzida quantidade de dados.

5.4. Adição de dados sintéticos ao grupo de treino

Com as redes devidamente escolhidas e calibradas, o próximo passo é analisar o impacto dos patches sintéticos gerados pela rede SegFlow, que será treinada com dados dos grupos de treino conforme descrito na seção 4.6. Os grupos de teste são então avaliados com e sem a adição dos dados sintéticos de forma a avaliar seu impacto. Seguindo o exemplo do artigo onde o SegFlow é apresentado [33], foram feitos também testes também na U-Net original além de nas redes mais exitosas de forma a entender melhor o impacto geral da técnica.

A quantidade de dados sintéticos adicionados ao conjunto de treino do *dataset* Petrobras foi a mesma da sugerida no artigo original: 160. Já no *dataset* Kaggle, o melhor resultado foi obtido sem o uso de nenhum exemplo sintético, então serão reportados os resultados que apresentaram melhora no baseline.

Dataset Petrobras

Rede	Exemplos sintéticos	IoU
U-ResNeXt	Sim	**95.64**
U-ResNeXt	Não	91.76
U-Net	Sim	88.11
U-Net	Não	82.02

Tabela 11 - Resultados no teste para a adição de exemplos sintéticos no conjunto de treino do *dataset* Petrobras.

Dataset Petrobras

Rede	Exemplos sintéticos	IoU-Kaggle
U-Xception	Sim	81.74
U-Xception	Não	**83.02**
U-Net	Sim	72.16
U-Net	Não	71.69

Tabela 12 - Resultados no teste para a adição de exemplos sintéticos no conjunto de treino do *dataset* Kaggle.

Com isso, reporta-se o resultado final de 83.02 no conjunto de teste Kaggle. Vale lembrar que os resultados no grupo de teste Kaggle foram avaliados na métrica do site, como descrito na seção 4.10. O resultado de 95.64 nos *patches* do *dataset* Petrobras ainda serão pós-processados de forma a serem apresentados no contexto da figura completa.

5.5. Reconstrução das imagens e refinamento no *dataset* de teste Petrobras

Os resultados finais do dataset Petrobras são dados pelo processamento dos *patches* preditos, que levarão à montagem da máscara completa da imagem de teste.

A abordagem mais imediata seria dividir a imagem cuja segmentação se intenciona fazer em *patches* dispostos lado a lado para então montar o "quebra-cabeça" com as predições de máscaras. Este método resulta na última imagem da Figura 34. Porém, como a predição de um pixel depende do contexto em que ele está inserido, há a possibilidade de obtenção de melhores resultados a partir do

uso de diferentes cenários - ou diferentes posicionamentos de *patches* - para se realizar a predição. Como descrito na seção 4.8.1, a classificação final do pixel será dada pela "votação" das máscaras geradas. Essa sobreposição é realizada ao modo "stride" descrito na seção 4.1.2.

Sobreposição (%)	IoU
0	88.40
50	94.21
67	95.35
75	95.36

Tabela 13 - resultados na métrica IoU da máscara completa do conjunto de teste com uso de *patches* com diferentes sobreposições entre eles.

A Tabela 13 mostra os ganhos gerados pela geração de *patches* com diferentes sobreposições no grupo de teste, que levam a diferentes quantidades de patches usados para predizer cada pixel. A Figura 34 mostra como estas predições ficam quando somados os pixels de predições das máscaras sobrepostas, enquanto a Figura 35 mostra o resultado do comitê de máscaras com a sobreposição que teve maior IoU.

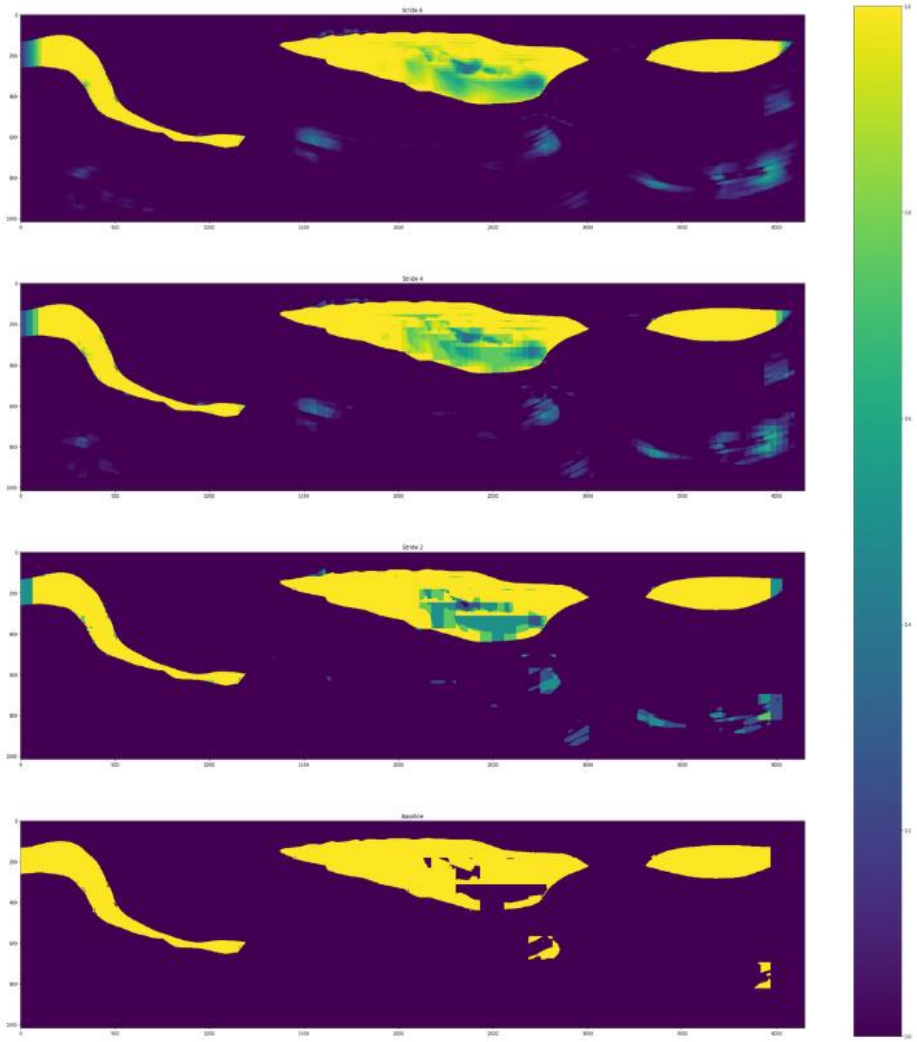


Figura 34 - Soma das máscaras geradas pelo modelo a partir de *patches* com diferentes sobreposições. De cima para baixo as sobreposições são de 75%, 67%, 50% e 0

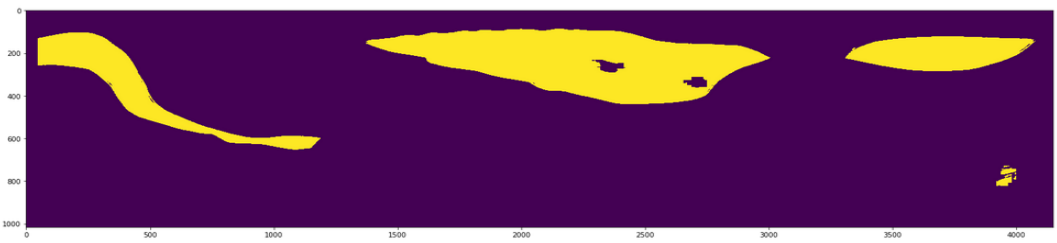
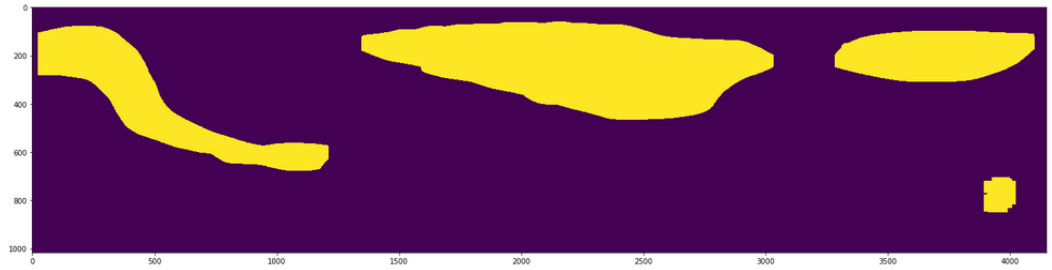


Figura 35 - resultado do comitê de máscaras com IoU de 95.36

Como se pode ver na Figura 35, ainda há falhas a serem corrigidas na máscara predita. Isso é melhorado com o uso dos algoritmos de erosão e dilatação descritos na seção 4.8.2. A Figura 36-a mostra a dilatação feita para cobrir os buracos das estruturas salinas, enquanto a Figura 36-b mostra o resultado final após a aplicação do filtro de erosão.

a)



b)

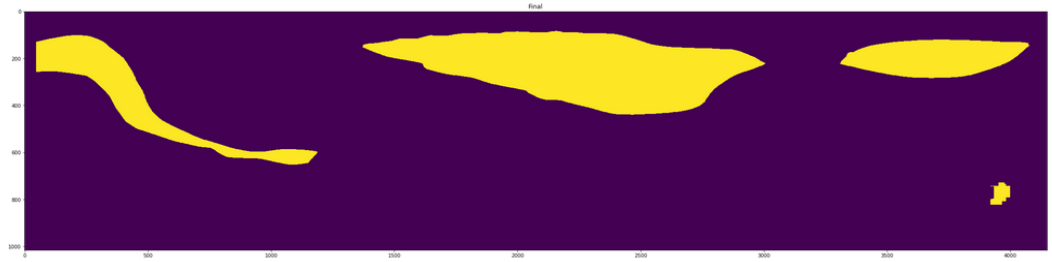


Figura 36 - Resultado das operações de a) dilatação e b) erosão.

O resultado na métrica IoU na figura final é de 96.53, com sua maior falha sendo a estrutura inexistente no canto inferior esquerdo. Com exceção desta falha, o resto da figura é notavelmente semelhante à máscara verdadeira mostrada novamente na Figura 37.

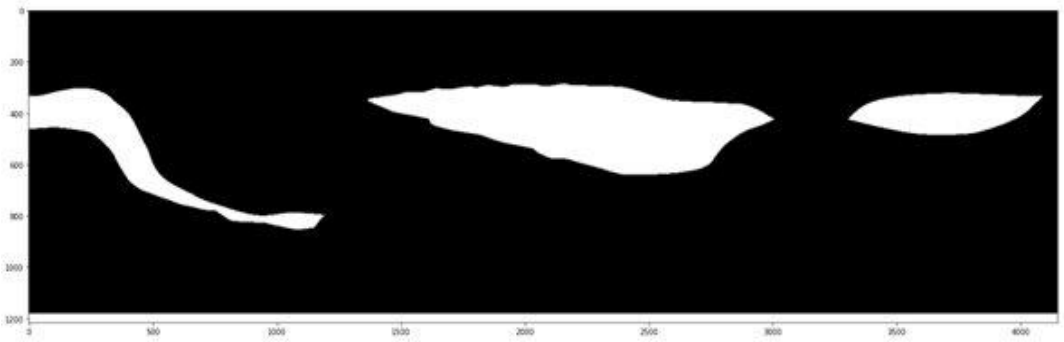


Figura 37 - Máscara verdadeira do conjunto de teste do *dataset* Petrobras

6. Conclusões

O uso de CNNs tem se expandido para muito além do domínio em que foram concebidas. Não apenas em diferentes tarefas de visão computacional mas também em áudios e até processamento de linguagem natural [42, 43]. Tirar proveito dos frequentes avanços desta área se torna assim uma forma de resolver diversos problemas com uma alta chance de sucesso.

Esta dissertação mostra que com o uso metódico de diferentes técnicas acessíveis se pode fazer significativos avanços em tarefas extremamente complexas para serem realizadas até por humanos. Estes tipos de técnicas têm sido mais utilizadas recentemente na área da geofísica e prometem ter impacto significativo em suas práticas.

6.1. Proposta

Este trabalho propõe o uso de FCNs como forma eficiente de se resolver tarefas de segmentação semântica em imagens de origem sísmica. Foi demonstrada a grande eficiência do conceito do *transfer learning* em *datasets* mais vastos e bem estudados para superar a falta de dados anotados, bem como o uso das redes Xception e ResNeXt como boas formas de extrair atributos das imagens. Além disso, módulos e estruturas adicionais foram explorados, com suas utilizações justificadas pela relação de suas arquiteturas com o problema em questão, o que pode levar a melhorias e até elaborações de mais módulos como os abordados neste trabalho. Por fim, técnicas de pós-processamento elevaram ainda mais a qualidade dos resultados, levando a um preditor extremamente eficiente para os tipos de *datasets* utilizados.

6.2. Contribuições

No *dataset* Kaggle, foi obtido um IoU de 83.02% na predição dos patches. Apesar desta pontuação não levar a altas posições no leaderboard deste desafio, é importante dizer que o foco deste trabalho foi em aprendizado estritamente

supervisionado. Discussões nos fóruns do Kaggle bem como artigos publicados mostram o uso de técnicas de aprendizado semi-supervisionado que utilizavam as próprias imagens separadas para o teste como parte do treino. Embora sejam estratégias legítimas e embasadas, principalmente em ambientes de competição, foi feita a escolha por não utilizá-las nesta pesquisa. Pontuações na casa dos 83% foi reportado em um dos tópicos mais comentados [50] como o teto atingido por alguns usuários sem as técnicas de aprendizado semi-supervisionado.

Já no *dataset* Petrobras, técnicas como o *transfer learning* e criação de exemplos sintéticos foram usados com sucesso, ao ponto de levar a um IoU de 96.53% na imagem de teste, o que representa uma melhora significativa em relação às redes de baselines.

É interessante o contraste percebido nas técnicas e estruturas que funcionaram ou não em cada um dos *datasets*. No Kaggle, com imagens mais complexas, redes com estruturas mais elaboradas foram frutíferas, enquanto a criação de dados sintéticos não, talvez pela dificuldade de recriar essa complexidade. Já no *dataset* Petrobras, redes relativamente mais simples (sem módulos a mais) foram as que produziram resultados melhores, além do sucesso na criação de exemplos sintéticos.

Além dessas análises, se concretiza como contribuição a disponibilização do código para o projeto da parceria da PUC-Rio com a Petrobras.

6.3. Trabalhos Futuros

Tendo em vista o grande sucesso da técnica de *transfer learning* nos *encoders*, propõe-se o uso do mesmo processo para a rede completa. Como abordado brevemente na seção 4.6, para se fazer o *transfer learning* na rede completa, será necessário o uso de *datasets* robustos de segmentação semântica (a exemplo do Imagenet para classificação) como o PASCAL-VOC, que exige certa complexidade em seu processamento e treinamento. A boa notícia é que já existem resultados promissores em *datasets* de imagens médicas, como mostrado em [28]. Podem também ser usados *datasets* sísmicos mais completos e maiores para este tipo de pré-treinamento, como o F3 [23].

A segmentação de imagens médicas pode ser uma inspiração para o campo das imagens sísmicas, como a própria U-Net demonstrou. Isso porque normalmente lidam com imagens cujas estruturas mais importantes são fronteiras e texturas, assim como nos datasets abordados nesta dissertação.

Por fim, as técnicas de aprendizado semi-supervisionado despontam como formas eficazes de se melhorar a eficiência das redes trabalhadas. Técnicas como o pseudo-labeling [34], que usam comitês de diferentes modelos para inferir máscaras de uma parte do conjunto de teste são extremamente facilitadas pela pronta disponibilidade de *encoders* pré-treinados, cujas diferentes características podem fazer com que se ajudem mutuamente, mitigando suas fraquezas individuais.

7. Bibliografia

[1] JONES, Ian F.; DAVISON, Ian. Seismic imaging in and around salt bodies. **Interpretation**, v. 2, n. 4, p. SL1-SL20, 2014.

[2] Resultados da busca por “salt” na SEG Library: <https://library.seg.org/action/doSearch?AllField=salt> Acessado em 2 de abril de 2021

[3] LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. Fully convolutional networks for semantic segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2015. p. 3431-3440.

[4] RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-net: Convolutional networks for biomedical image segmentation. In: **International Conference on Medical image computing and computer-assisted intervention**. Springer, Cham, 2015. p. 234-241.

[5] CHOLLET, François. Xception: Deep learning with depthwise separable convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2017. p. 1251-1258.

[6] XIE, Saining; GIRSHIK, Ross; DOLLÁR, Piotr. Aggregated residual transformations for deep neural networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2017. p. 1492-1500.

[7] <www.image-net.org> Acessado em 2 de abril de 2021

[8] LATEEF, Fahad; RUICHEK, Yassine. Survey on semantic segmentation using deep learning techniques. **Neurocomputing**, v. 338, p. 321-348, 2019.

[9] WANG, Zhe; HEGAZY, Tamir; LONG Zhiling; ALREGIB, Ghassan. Noise-robust detection and tracking of salt domes in postmigrated volumes using texture, tensors, and subspace learning. **Geophysics**, v. 80, n. 6, p. WD101-WD116, 2015.

[10] CIVITARESE, Daniel; SZWARCMAN, Daniela; BRAZIL, Emilio Vital. Semantic segmentation of seismic images. **arXiv preprint arXiv:1905.04307**, 2019.

[11] CIVITARESE, Daniel; SZWARCMAN, Daniela; BRAZIL, Emilio Vital; SILVA, Reinaldo Mozart D. Seismic facies segmentation using deep learning. **ACE**, 2018.

[12] KARCHEVSKIY, Mikhail; ASHRAPOV, Insaf; KOZINKIN, Leonid. Automatic salt deposits segmentation: A deep learning approach. **arXiv preprint arXiv:1812.01429**, 2018.

[13] CIVITARESE, Daniel; SZWARCMAN, Daniela; BRAZIL, Emilio Vital; SILVA, Reinaldo Mozart D. Transfer learning applied to seismic images classification. **AAPG Annual and Exhibition**, 2018.

[14] LI, Shengron; YANG, Changchun; SUN, Hui; ZHANG, Hao. Seismic fault detection using an encoder–decoder convolutional neural network with a small training set. **Journal of Geophysics and Engineering**, v. 16, n. 1, p. 175-189, 2019.

[15] HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016. p. 770-778.

[16] HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Identity mappings in deep residual networks. In: **European conference on computer vision**. Springer, Cham, 2016. p. 630-645.

[17] HU, Jie; SHEN, Li; SUN, Gang. Squeeze-and-excitation networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2018. p. 7132-7141.

[18] ROY, Abhijit Guha; NAVAB, Nassir; WACHINGER, Christian. Concurrent spatial and channel ‘squeeze & excitation’in fully convolutional networks. In: **International conference on medical image computing and computer-assisted intervention**. Springer, Cham, 2018. p. 421-429.

[19] SZEGEDY, Christia; LIU, Wei; JIA Yangqing; SERMANET, Pierre. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2015. p. 1-9.

[20] TIELEMAN, Tijmen; GEOFFERY, Hinton. RMSprop gradient optimization. 2014. Disponível em <http://www.cs.toronto.edu/tijmen/csc321/slides/lecture_slides_lec6.pdf> Acessado em 2 de abril de 2021.

[21] SHEN, Chen; ROTH Holger; ODA, Hirohisa; ODA, Masahiro; HAYASHI, Yuichiro; MISAWA, Kazunari; MORI, Kensaku. On the influence of Dice loss function in multi-class organ segmentation of abdominal CT using 3D fully convolutional networks. **arXiv preprint arXiv:1801.05912**, 2018.

[22] SILVA, Reinaldo Mozart; BARONI, Lais; FERREIRA, Rodrigo S; CIVITARESE, Daniel; SZWARCMAN, Daniela; BRAZIL, Emilio V. Netherlands dataset: A new public dataset for machine learning in seismic interpretation. **arXiv preprint arXiv:1904.00770**, 2019.

[23] <https://terranubis.com/datainfo/Netherlands-Offshore-F3-Block-Complete> Acessado em 2 de abril de 2021.

[24] <http://host.robots.ox.ac.uk/pascal/VOC/> Acessado em 2 de abril de 2021

[25] PENG, Chao; ZHANG, Xiangyu; YU, Gang; LUO, Guiming; SUN, Jian. Large kernel matters--improve semantic segmentation by global convolutional network. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2017. p. 4353-4361.

[26] SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

[27] HE, Kaimin; ZHANG Xiangyu; REN, Shaoqing; SUN, Jian. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016. p. 770-778.

[28] GOYAL, Manu; YAP, Moi Hoon; HASSANPOUR, Saeed. Multi-class semantic segmentation of skin lesions via fully convolutional networks. **arXiv preprint arXiv:1711.10449**, 2017.

[29] KARIMI, Davood; WARFIELD, Simon K.; GHOLIPOUR, Ali. Critical Assessment of Transfer Learning for Medical Image Segmentation with Fully Convolutional Neural Networks. **arXiv preprint arXiv:2006.00356**, 2020.

[30] PAN, Sinno Jialin; YANG, Qiang. A survey on transfer learning. **IEEE Transactions on knowledge and data engineering**, v. 22, n. 10, p. 1345-1359, 2009.

[31] LI, Hanchao; XIONG, Pengfei; AN, Jie; WANG, Lingxue. Pyramid attention network for semantic segmentation. **arXiv preprint arXiv:1805.10180**, 2018.

[32] <https://www.kaggle.com/c/tgs-salt-identification-challenge> Acessado em 2 de abril de 2021

[33] HENRIQUES, Luis Felipe; MILIDIÚ, Ruy Luis.; COLCHER, Sérgio; BULCÃO, André; BARROS, Pablo. SegFlow: A Conditional

Normalizing Flow Model for the Data Augmentation of Samples in Synthetic Seismic Images. **arXiv preprint arXiv:2106.08269**, 2021.

[34] BABAKHIN, Yauhen; SANAKOYEU, Artsiom; KITAMURA, Hirotoshi. Semi-supervised segmentation of salt bodies in seismic images using an ensemble of convolutional neural networks. In: **German Conference on Pattern Recognition**. Springer, Cham, 2019. p. 218-231.

[35] Bengio, Y., Goodfellow, I., & Courville, A. (2017). *Deep learning* (Vol. 1). Massachusetts, USA:: MIT press. Capítulo 9. Disponível em https://www.deeplearningbook.org/contents/convnets.html Acessado em 2 de abril de 2021.

[36] PAN, Sinno Jialin; YANG, Qiang. A survey on transfer learning. **IEEE Transactions on knowledge and data engineering**, v. 22, n. 10, p. 1345-1359, 2009.

[37] <https://www.kaggle.com/pestipeti/explanation-of-scoring-metric> Acessado em 2 de abril de 2021

[38] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, p. 1097-1105, 2012.

[39] TREML, Michael et al. Speeding up semantic segmentation for autonomous driving. In: **MLITS, NIPS Workshop**. 2016.

[40] HE, Yihui; ZHU Chenchen; WANG, Jianren; SAVVIDES, Marios; ZHANG, Xiangyu. Bounding box regression with uncertainty for accurate object detection. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2019. p. 2888-2897.

[41] <https://paperswithcode.com/sota/image-classification-on-imagenet> Acessado em 2 de abril de 2021

[42] YIN, Wenpeng; KANN, Katharina; YU Mo; SCHÜTZE, Hinrich. Comparative study of CNN and RNN for natural language processing. **arXiv preprint arXiv:1702.01923**, 2017.

[43] PALAZ, Dimitri; COLLOBERT, Ronan. **Analysis of cnn-based speech recognition system using raw speech as input**. Idiap, 2015.

[44] <https://cs231n.github.io/linear-classify/> Acessado em 2 de abril de 2021

[45] <https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1> Acessado em 2 de abril de 2021

[46] <https://dataaspirant.com/data-augmentation-techniques-deep-learning/> Acessado em 2 de abril de 2021

[47] <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/> Acessado em 2 de abril de 2021

[48] <https://medium.com/fintechexplained/what-is-grid-search-c01fe886ef0a> Acessado em 2 de abril de 2021

[49] <https://towardsdatascience.com/colab-pro-is-it-worth-the-money-32a1744f42a8> Acessado em 2 de abril de 2021

[50] <https://www.kaggle.com/meaninglesslives/getting-0-87-on-private-lb-using-kaggle-kernel> Acessado em 2 de abril de 2021

[51] <https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11> Acessado em 2 de abril de 2021

[52] <https://www.deeplearningbook.com.br/variational-autoencoders-vaes-definicao-reducao-de-dimensionalidade-espaco-latente-e-regularizacao/> Acessado em 2 de abril de 2021

[53] ARDIZZONE, Lynton; LÜTH, Carsten; KRUSE, Jakob; ROTHER, Carsten; KÖTHE, Ullrich. Guided image generation with conditional invertible neural networks. **arXiv preprint arXiv:1907.02392**, 2019.

[54] <https://towardsdatascience.com/learning-curve-to-identify-overfitting-underfitting-problems-133177f38df5> Acessado em 2 de abril de 2021

[55] <https://matplotlib.org/> Acessado em 2 de abril de 2021