MC040 - Estágio de Iniciação Científica
Exploração de arquiteturas de redes neurais para melhoria de detecção de características em dados sísmicos

Aluno: Matheus Rotta Alves
Orientador: Prof. Dr. Edson Borin
Co-orientadora: Prof[a]. Dra. Sandra Avila

Primeiro Semestre de 2018

Resumo

O processo de migração em tempo de um dado sísmico empilhado requer um mapa de velocidades suave. Esse pode ser obtido escolhendo-se ápices de refração de alta confiança e interpolando-se os valores de velocidade escolhidos. O trabalho utilizado como ponto de partida automatizou o processo de escolha de tais ápices com machine learning, mais especificamente com redes neurais convolucionais.

O trabalho desenvolvido durante esta iniciação científica foi testar alguns métodos para checar se haveria melhoras. Entre esses testes foram feitos: penalização maior para falsos positivos, testes com redes completamente conectadas e transferência de aprendizado a partir de modelos pré-treinados com o benchmark ImageNet. Os resultados obtidos foram interessantes mas permaneceram abaixo do obtido com a LeNet do trabalho original.

1 Introdução

1.1 Contexto Geofísico

O problema em questão, do ponto de vista da geofísica, é o mapeamento da subsuperfície terrestre. Para tal fim, é feito um levantamento sísmico de uma região através de perturbações do meio a ser estudado. Essas perturbações então se propagam pelo meio, sofrem reflexão e voltam para a superfície, onde são captadas por geofones.

Isso nos permite mapear a subsuperfície, mas o dado gerado deve passar por diferentes técnicas de processamento, entre elas o CMP (common mid-point) e o CRS (common reflection surface) são destinadas a empilhar traços sísmicos que possuem relação específica entre si, como ponto médio comum ou superfície de reflexão comum. Com o dado já empilhado, pode-se optar por ainda fazer uma migração do dado empilhado em tempo, para a qual é necessário ter um mapa de velocidades suave. Para tal, é interessante usar um dado que tenha

apenas as difrações detectadas (a separação é feita com o método DRS), que se expressam na forma de hipérboles, pois a velocidade no ápice da difração é de alta confiança [1].

Para a detecção das difrações, um intérprete deve selecionar manualmente os ápices de maior confiança, já que alguns podem ter intersecção com caudas de outras hipérboles e isso pode interferir na velocidade aferida. É nessa parte do fluxo de trabalho que aprendizagem de máquina foi aplicada, a fim de automatizar a detecção de ápices, não onerando um profissional da área.

1.2 Pesquisa prévia

Assim, o trabalho original de onde se partiu fazia uma detecção automática de ápices. Isso foi feito interpretando o dado sísmico como uma imagem. Para tanto, foi feita a anotação de matrizes 64x64 de dados em duas categorias - ápice e não ápice -, e a arquitetura escolhida para treinar esses dados foi uma rede convolucional neural, a LeNet.

Após o treinamento, passava-se uma janela deslizante pelo dado sísmico com apenas refrações, e para cada "frame"se fazia a propagação para frente (forward propagation) na rede. Isso gera o dado inferido que para cada ponto (dada uma margem) no dado haveria um nível de confiança se é ou não ápice. A partir daqui o usuário escolhe um limiar de confiança para filtrar da maneira adequada.

1.3 Motivação de cada teste

A penalização dos falsos positivos foi um teste motivado por uma característica intrínseca do problema: ápices ruins têm o potencial de atrapalhar bastante a interpolação e gerar um mapa de velocidades ruim. Deste modo, era melhor escolher poucos ápices com muita confiança. Apesar de se poder escolher, ao final do treinamento, apenas ápices com confiança acima de um certo limiar (0.9995 de confiança de que é um bom ápice, por exemplo), queria-se saber se incorporar uma penalização adicional na função custo para falsos positivos geraria um resultado melhor, pois para alguns problemas é melhor penalizar com maior severidade algum tipo de erro [2].

Os testes feitos com redes completamente conectadas, em contraste a redes que teriam partes iniciais de extração de features para depois haver tomada de decisão, foram sugeridos pois se associou a detecção de ápices principalmente com a centralidade do evento, deste modo não seria necessário extrair features primeiro (hipótese A).

Os teste de transferência de aprendizagem foram realizados pois se percebeu que a hipótese A não se sustentava, e que tratar dados sísmicos como imagens (como foi feito no trabalho original) seria ainda a melhor opção. Deste modo, decidiu-se utilizar modelos pré-treinados no benchmark ImageNet, um dos benchmarks mais famosos e completos, para poder utilizar arquiteturas mais complexas, como a VGG-19 e a ResNet, mas sem treiná-las por inteiro, dada a limitação das máquinas. A ideia então seria aumentar de pouco em pouco o quanto seria treinado da rede. (em progresso).

2 Cronograma

- Janeiro:
  introdução aos conceitos de geofísica e implementação das técnicas de processamento de dados sísmicos (CMP); contribuiu para o melhor entendimento da área e da assimilação do jargão técnico.

- Fevereiro:
  prosseguimento à implementação de métodos de processamento (CRS) e início do curso de *machine learning* (Coursera [3]); foi uma introdução mais teórica aos conceitos básicos de aprendizagem de máquina.

- Março:
  continuação do curso de *machine learning* (Coursera) – interrompido e migrado para o curso de *Deep Learning* (Udacity [4]); deu um viés mais prático (*TensorFlow* [5]) e aproximou o grupo do estado da arte dessa área de pesquisa.

- Abril:
  continuação do curso da Udacity e decisão das frentes mais específicas de atuação dos alunos de Iniciação Científica, no meu caso: transferência de aprendizagem;

- Maio:
  implementação da penalização dos falsos positivos e testes com redes completamente conectadas no *Keras* [6];

- Junho:
  testes com transferência de aprendizagem e elaboração do relatório;

3 Conceitos Explorados

3.1 Redes Neurais

Redes neurais são modelos matemático-computacionais que se inspiram na maneira como nosso cérebro funciona, em que temos unidades de transmissão da informação (neurônios) e ligações entre eles (mais fortes ou menos fortes). Do mesmo modo, a rede neural tem suas unidades básicas, os nós (neurônios), e a conexão entre eles são determinadas por pesos.

Esses nós determinam camadas, tal que temos uma camada de entrada, uma ou mais camadas escondidas e uma camada de saída que seria a decisão tomada pela rede.

Redes neurais simples, pelo menos no escopo desse relatório, se referem a redes neurais que possuem apenas camadas completamente conectadas na sua composição. Um exemplo de rede neural completamente conectada pode ser visto na figura 1.

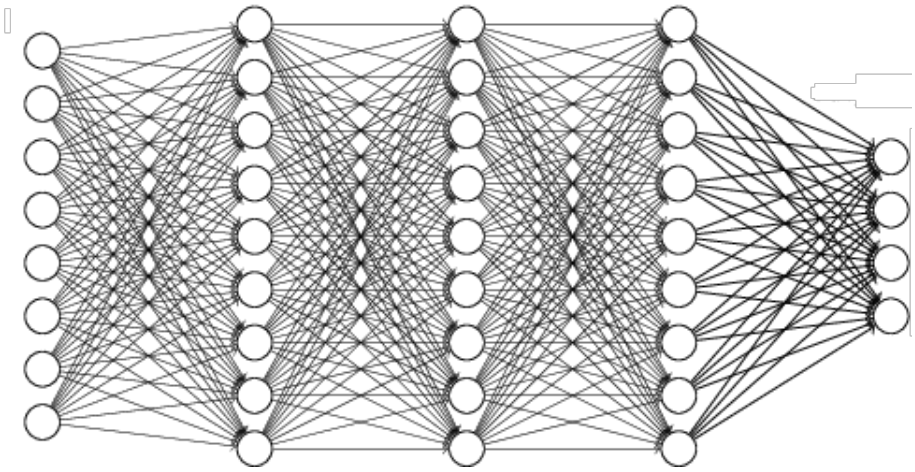

Figura 1: Rede Neural Simples com 3 camadas escondidas - Imagem Original em [7]

Redes neurais convolucionais recebem esse nome por possuírem camadas de convolução. Tais redes têm bom desempenho em tarefas como classificação de imagens pois as convoluções levam em conta a coerência espacial local das imagens, combinando pixels adjacentes e extraindo features relevantes como contornos. Um exemplo de rede neural convolucional pode ser visto na figura 2: repare que até S4 estamos ou fazendo convoluções ou *subsampling* (diminuição das amostras), e a partir daí temos camadas completamente conectadas que relacionam os mapas de *features* às predições em questão.

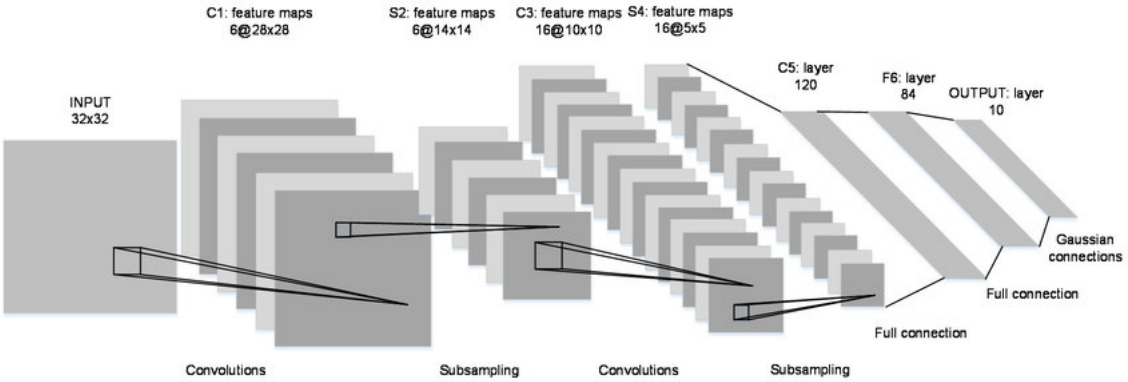

Figura 2: Rede Neural Convolucional LeNet de Yann LeCun - Imagem Original em [8]

3.2 Transferência de aprendizagem

A maioria dos algoritmos de aprendizagem partem do pressuposto que os dados de treino e os de inferência são, necessariamente, do mesmo tipo. No entanto, em muitas ocasiões, essa noção pode ser quebrada.

Por exemplo, pode-se ter uma tarefa de classificação em um domínio de interesse, mas há dados anotados suficientes apenas em um outro domínio, relacionado com o anterior. Quando isso ocorre, transferência de aprendizagem pode ser aplicada para atenuar esforços com anotação de dados. [9]

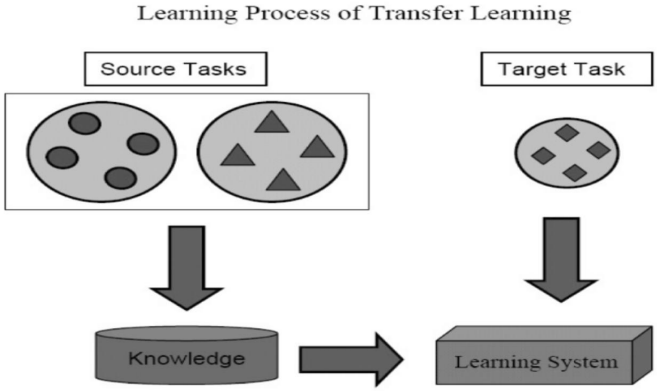

Figura 3: Esquema ilustrativo sobre transferência de aprendizagem - Imagem Original em [9]

Isso foi observado justamente no problema em questão de detecção de ápices: há poucos dados anotados e o custo associado a anotá-los é alto, visto que um profissional da área de geofísica teria que dedicar grande quantidade de tempo.

Ademais, como os resultados obtidos indicaram que tratar o dado sísmico como imagem é boa opção, foi decidido então que se transferiria conhecimento adquirido de redes treinadas em grandes datasets de imagens para o contexto da detecção de ápices em dados sísmicos.

4 Abordagem

4.1 Redes Neurais Simples

As redes completamente conectadas testadas neste trabalho foram implementadas com o Keras, trocando no código original a parte em que se chamava a rede original com a rede a ser testada, mantendo o código que verificava a acurácia da inferência nos dados de validação.
Para ajustar os dados, que estavam configurados para uma convolução, bastou linearizar a imagem.

4.2 Penalização dos Falsos Positivos

Como mencionado, o trabalho original usava uma implementação da rede convolucional LeNet, e o teste feito com a penalização dos falsos positivos foi apenas uma modificação no código em tensorflow da rede original que somava uma penalização a mais para os falsos-positivos. Assim, "Clean Loss"é o custo proveniente do cálculo da entropia cruzada, "Penal. FP's"é o custo adicional para os falsos positivos e "Loss"é a soma de ambos, que era passada como o custo para o algoritmo.

4.3 Transferência de Aprendizagem

A rede VGG-19 foi a primeira escolhida por possuir várias camadas de convolução da qual se poderia extrair features. Primeiro se carregou os pesos da VGG-19 já treinados para o benchmark ImageNet. Deste modo, o domínio fonte, ou tarefa fonte, aqui seria o desafio ImageNet, que possui mil classes diferentes de imagens, enquanto que o domínio alvo é o problema de detecção de ápices.

Em cada teste a quantidade de camadas congeladas foi diminuindo, e passamos a treinar mais camadas. O Keras tem uma funcionalidade que permite selecionar que camadas serão treinadas e quais não serão. Também foi necessário replicar os dados duas vezes pois a entrada esperada pela rede era uma imagem comum com três canais.

5 Resultados Preliminares

Os primeiros testes feitos foram com a penalização dos falsos positivos. Na tabela 1 pode-se observar os resultados obtidos, levando em conta que alpha é a severidade da penalização e que alpha=0 é equivalente ao experimento original.

Tabela 1:

	Acurácia ao Final do Treinamento	Validação: Não-Ápices	Valid.: Ápices	Loss	Clean Loss	Penal. FP's
alpha=1000	80.2%	97.48%	58.25%	8.211	0.525	7.776
alpha=300	96.64%	94.96%	93.20%	4,426	0,198	4,228
alpha=100	94.1%	95.80%	93.20%	2,686	0,217	2,469
alpha=10	93.2%	94.96%	93.20%	0,387	0,179	0,208
alpha=1	95%	92,44%	97,09%	0,242	0,213	0,029

Os testes com redes fully-connected produziram os seguintes resultados:

Tabela 2:

1 camada escondida:				acurácia na validação	
unidades	ativação	época final	perda	Não Ápice	Ápice
4096	relu	500	0,2980	81.51%	78.64%
2048	relu	500	0,1991	83.19%	79.61%
6144	relu	500	0,3704	81.51%	78.64%

Tabela 3:

2 camadas escondidas:					acurácia na validação	
unidades 1a	unidades 2a	ativação	época final	perda	Não Ápice	Ápice
4096	2048	relu	500	0,4959	82.35%	75.73%
6144	4096	relu	500	0,7136	85,71%	78,64%
8192	4096	relu	500	0,4431	83,19%	80,58%

Os testes com transferência de aprendizado, que por enquanto só foram feitos com a rede VGG-19, produziram os seguintes resultados preliminares:

Tabela 4:

extraindo features da block5_pool (ou seja, treinando apenas o topo)	acurácia na validação	
top: fc 4096 unidades	Não Ápices	Ápices
Acurácia:	94.12%	81.55%
top: fc 4096 + 4096 unidades	Não Ápices	Ápices
Acurácia:	90.76%	86.41%

Tabela 5:

camadas congeladas: até a block5_conv2 (18 primeiras)	acurácia na validação	
top: fc 4096 + 4096 unidades	Não Ápices	Ápices
Acurácia:	90.76%	81.55%

Tabela 6:

camadas congeladas: até a block4_conv2 (14 primeiras)	acurácia na validação	
top: fc 4096 + 4096 unidades	Não Ápices	Ápices
Acurácia:	89.08%	90.29%

6 Discussão

Os dados da tabela 1 referentes a penalização dos falsos positivos apresentaram resultados esperados, em que quanto maior o alpha, mais a rede está inclinada a predizer "Não-Ápice", o que aumenta a acurácia dessa classe e diminui a quantidade de falsos positivos.

Um valor que mostrou um bom equilíbrio para o problema foi alpha=300, que reduziu a quantidade de falsos positivos mas sem prejudicar muito a acurácia da classe positiva "Ápice".

Os testes com redes completamente conectadas, tabelas 2 e 3, foram inconclusivos entre si, mas mostram que redes neurais simples são inferiores ao uso de convolução, pois apresentaram em média 80% de acurácia em geral, enquanto que a LeNet original apresentava aprox. 94% de acurácia (no conjunto de validação).

Enfim, os testes feitos com transferência de aprendizagem provam que é possível transferir aprendizado do contexto de classificação de imagens em geral para o contexto de detecção de ápices, já que apresentaram em média 86% de acurácia na validação. No caso em que se deu a maior liberdade de treinamento (tabela 6), pois congelamos apenas as 14 primeiras camadas, foi o melhor resultado ($\approx 90\%$), mas deve-se analizar o custo benefício também pois quanto mais camadas liberamos, mais demora o treinamento.

7 Propostas para o segundo semestre

Imagino que para o segundo semestre eu possa continuar explorando técnicas de transferência de aprendizagem. Por exemplo, posso diminuir ainda mais o número de camadas congeladas na VGG-19, testar outras redes como a ResNet-50, ou as da arquitetura Inception (Inception_Resnet_V2 e Inception_V3).

7.1 Cronograma

- Julho:
  Testes de transferência de aprendizagem com a ResNet-50. Adicionar a penalização dos falsos positivos às implementações de *transfer learning*;
- Agosto:
  Testes de transferência de aprendizagem com as redes de arquitetura Inception;
- Setembro:
  Em aberto;
- Outubro:
  Documentação e organização do código feito até este ponto; em aberto também.
- Novembro:
  Elaboração do Relatório Final de Iniciação Científica.

Referências

[1] Leiv-J. Gelius Jorge H. Faccipieri, Dany Rueda Serrano and Martin Tygel. Recovering diffractions in crs stacked sections. 2013. `http://www.earthdoc.org/publication/publicationdetails/?publication=68073`.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] Coursera machine learning course. `https://www.coursera.org/learn/machine-learning`.

[4] Udacity deep learning course. https://br.udacity.com/course/deep-learning--ud730.

[5] Tensorflow, an open source machine learning framework. https://www.tensorflow.org.

[6] Keras: The python deep learning library. https://keras.io.

[7] Mathworks. https://www.mathworks.com/matlabcentral/fileexchange/64247-simple-neural-network.

[8] Viet Tra, Jaeyoung Kim, Sheraz Khan, and Jongmyon Kim. Bearing fault diagnosis under variable speed using convolutional neural networks and the stochastic diagonal levenberg-marquardt algorithm. 17:2834, 12 2017.

[9] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.