Convoluções com esparsidade

Evair de Jesus Silva Cunha
Centro de Informática (Cin)
Universidade Federal de Pernambuco (UFPE)
Pernambuco, Brasil
ejsc@cin.ufpe.br

Abstract - Since 2012 with the work of Krizhevsky et. al the convolutional networks has raised for an prominence in the area of computer vision. In 2014, the winning algorithm (Inception) reached really close the average human classification error rate of five percent, about 6.67% what reinforces the growing interests on it. This paper is an brief analysis of the paper "Going deeper with convolutions"that result in the first version of Inception model, furthermore create a variant of the original model replacing the convolutions of 5x5 with two of 3x3 including an option to the model accept inputs of variable size. Lastly was performed benchmarks for the task of classification over the CIFAR10 dataset to evaluate the accuracies, three models were considered: Inception-Customized, VGG19 and ResNet18.

***Index Terms*—Computer Vision, CNN, GoogLeNet, Machine Learning**

I. Introdução

Os avanços no desempenho das redes convolucionais tiveram um salto desde o ano de 2012, quando Krizhevsky et al [1] mostrou a capacidade de sua rede a "AlexNet"que foi a vencedora da competição da ImageNet - ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2012), a partir de então uma larga variedade de aplicações em visão computacional passaram a empregar essa rede, dentre as áreas pode-se citar: segmentação [2], detecção de objetos [3], poses humanas [4], rastreamento de objetos [5] e classificação de video [6].

Focado no desempenho das redes convolucionais surgiram outros modelos que da sua maneira tentavam aumentar a acurácia e no mesmo sentido diminuir as taxas de erro desses modelos, seguindo esse curso em 2014 surgem duas arquiteturas que melhoraram significativamente utilizando em sua composição redes mais profundas e largas.

As arquiteturas tratadas são a VGGNet [7] e GoogLeNet [8], mesmo com a rede Inception sendo a vencedora da ILSVRC de 2014, ambas obtiveram desempenho bem próximo na competição de classificação. Assim como a AlexNet, as novas arquiteturas passaram a ser adotadas em uma nova gama de aplicações, pois com os novos ganhos na acurácia, isso significa a abertura de novas possibilidades para realizar tarefas que as arquiteturas anteriores não eram capazes de fazer.

Embora a arquitetura da VGG apresente uma simplicidade, pois utiliza camadas convolucionais 3x3 empilhadas umas sobre as outras com profundidade crescente, ela traz consigo um alto custo computacional. A GoogLeNet, por exemplo, utiliza apenas 5 milhões de parâmetros a par de comparação a AlexNet usou 60 milhões, o que representa uma redução de 12 vezes, já a VGG utilizou a impressionante quantidade de 180 milhões de parâmetros , uma das razões para esse valor de parâmetros ser baixo é porque esta rede usa o Average-Pooling em vez de camadas totalmente conectadas na parte superior da CNN, eliminando uma grande quantidade de parâmetros menos importantes [9]. Essa característica fez com que a rede Inception fosse a mais adequada para cenários onde existe pouca quantidade de memória e poder computacional, uma das aplicações possíveis foi a implementação em cenários de Big data [10] e aparelhos móveis [11].

Este artigo propõe-se a realizar uma análise e reprodução com experimentação da arquitetura descrita no artigo *Going deeper with convolutions*, além de realizar uma breve comparação da métrica de acurácia com as arquiteturas da VGG, Inception e a ResNet [12].

Além disso, como desafio, foi criada uma variante da rede Inception original que tem capacidade de aceitar entradas de tamanho variável e a redução ainda maior do custo computacional com a substituição das camadas convolucionais de 5x5 por duas camadas empilhadas de 3x3, tendo como benefícios a redução da quantidade de pesos necessários, deste modo reduzindo o custo computacional e necessidade de memória, as características extraídas serão altamente locais auxiliando na captura de características pequenas e granuladas na imagem, e por fim com o uso de kernels sequencialmente permite que a rede aprenda características mais complexas.

II. Trabalhos Relacionados

Existem diversos trabalhos que utilizam a Inception como base para novas arquiteturas para tarefas de visão computacional.

Dentre os trabalhos com os módulos Inception que atingiram o estado da arte, pode-se citar [13] que trouxe a proposta de aceleração de treinamento das redes com a adição de *batch normalization* para regularizar a mudança de covariação durante o treino, acrescentando ainda que é feita a remoção do dropout e a taxa de aprendizado é aumentada e com poucas épocas de treino foi possível atingir o estado da arte anterior e posteriormente se estabelecer como o novo estado da arte, esse que se tornaria a segunda versão ou Inception-V2.

Outro trabalho notável foi o de [14], a versão Inception-V3, neste trabalho os autores fizeram a regularização do modelo pela suavização dos rótulos, ou seja, eles propuseram um mecanismo de regularização pela estimação do efeito marginalizado do *label-dropout* durante o treinamento, outra proposta foi a fatorização da primeira camada convolucional de 7x7 para outras de 3x3, semelhante ao que foi experimentado nesse artigo, mas com a fatorização para 5x5 e por fim ainda mantendo parte da ideia original o uso de classificador auxiliar.

A Inception-v4 [15] é uma arquitetura de rede neural convolucional que se baseia nas versões anteriores da família Inception, simplificando a arquitetura e usando mais os módulos característicos *Inception* do que a versão Inception-v3, pondo uma ênfase maior nas conexões residuais para acelerar o processo de treinamento nesse tipo de rede.

Para este artigo foi utilizado a parte da ideia central da Inception-V3 apresentada com o objetivo de implementar uma variante que será mostrada na seção II.

III. Materiais e métodos

Com intuito de realizar a experimetação e avaliar se os resultados correspondem ao original e estão próximos ao de outras arquitetura será feita a implementação do modelo de CNN da GoogLeNet, por meio do Pytorch e com o auxílio do PrototypeML, teremos o treinamento da rede com os datasets do CIFAR10, que é um conjunto de dados de visão computacional bem estabelecido usado para reconhecimento de objetos. Este dataset consiste de 60.000 imagens coloridas de 32x32 contendo dentro dela 10 classes de objetos, com 6.000 imagens por classe, sendo 50.000 imagens para o treinamento e 10.000 imagens para teste, as classes são 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship e 'truck' que podem ser vistos na Figura 1. Esse dataset foi coletado e disponibilizado por Alex Krizhevsky, Vinod Nair e Geoffrey Hinton.

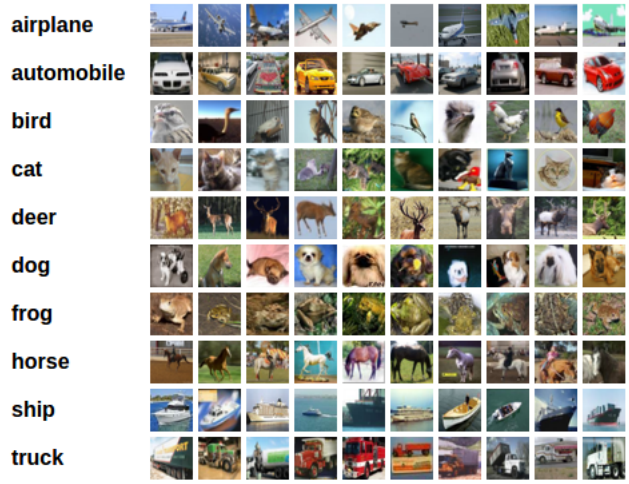

Fig. 1. Imagens do dataset CIFAR10, cada linha correspondendo as classes.

A implementação foi pensada para ser feita por blocos, isso seria para auxiliar no tratamento de erros de código, veja a Figura 2, outro fator levado em conta foi o uso de GPU, haja vista que redes convolucionais se beneficiam muito desse hardware para a aceleração do processo, ao contrário de uma CPU. Desta forma, o Colab [16] que é uma plataforma fornecida pelo Google, foi o ambiente em que isso foi propício, pois o código pode ser implementado por partes e existe oferta gratuita de GPU.

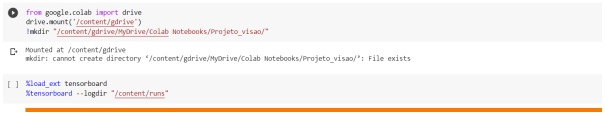

Fig. 2. Códigos em blocos.

Com a definição do dataset e qual será a plataforma para desenvolvimento do modelo, pode-se traçar um fluxo de como será o treinamento e se o modelo se adequa aos objetivos propostos. A Figura 2 demonstra a forma qual se pretende alcançar os objetivos propostos neste projeto.

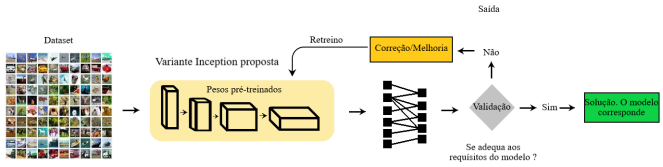

Fig. 3. Fluxo do esquema de trabalho.

A maneira mais fácil de se melhorar o desempenho de uma rede é aumentar o seu tamanho. Isso é, aumentar a profundidade com mais níveis e aumentar a largura com mais elementos por nível, dada a grande quantidade de dados rotulados. Entretanto existem algumas desvantagens, pois com esse aumento de tamanho significa que a quantidade de parâmetros também irá aumentar, tornando a rede mais propensa a *overfitting*; outra desvantagem é que um aumento disforme na rede levará a um aumento drástico no uso de recursos computacionais.

A ideia do Inception era gerar uma rede profunda sem que para isso houvesse um empilhamento massivo de camadas, nesse caso a solução passou pela utilização de camadas convolucionais de 1x1 e 3x3, o que ao invés de ser profundo se torna 'largo'.

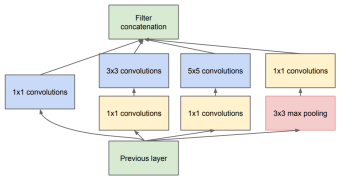

Fig. 4. Módulo Inception original descrito em [8].

Muito do desempenho obtido pela GoogLeNet foi obtido pela generosa redução de dimensão. A fatorização de maneira eficiente das convoluções, no caso de adicionar após os kernels de 1x1 outro kernel 3x3 gera uma expressiva representação local. Sendo assim, como forma de explorar essa fatorização os kernels de 5x5 foram substituídos por dois kernels de 3x3 para aumentar a eficiência computacional, uma vez que cada peso corresponde a uma multiplicação por ativação, portanto uma redução nas convoluções corresponde a redução de parâmetros e aceleração no treinamento.

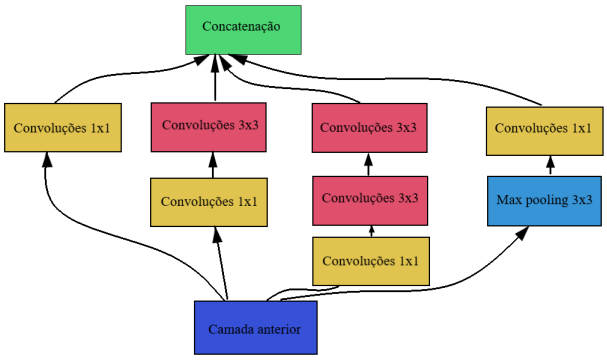

Fig. 5. Variante proposta para o modelo Inception

Levando em conta que são 22 camadas de profundidade, ou seja, apenas com parâmetros ou 27 contando com os pooling, sendo 9 blocos Inception, com essa alteração proposta houve uma redução de 7.008.824 parâmetros treinaveis, que foram aferidos na arquitetura com kernel de 5x5, para 6.701.240, isso representa uma leve redução de 4.38% na quantidade de parâmetros.

A. *Metodologia de treinamento*

Devido a grande quantidade de imagens na ImageNet o dataset CIFAR10 foi escolhido para substituir, tendo em vista o longo tempo que seria treinar com a ImageNet. O modelo proposto foi treinado algumas vezes e em média leva uma hora para o total de 10 épocas. O treinamento usou o Gradiente descendente estocástico com momentum de 0.9, o modo assíncrono ainda não existe nas bibliotecas do Pytorch. O learning rate é configurado como 0.001 e a cada 8 épocas o learning rate é diminuído em 0.1.

Durante os treinamentos percebeu-se que acima de 15 épocas o modelo começa a entrar em overfitting e não há melhora nas métricas de perda, outro detalhe é que o modelo VGG durou o dobro de tempo para finalizar o treinamento quando comparada aos outros modelos.

IV. Resultados

Durante os treinamentos as imagens foram reajustadas para o tamanho de 256 e então cortadas a partir do centro com tamanho de 224, os batches tiveram o tamanho fixo de 4 e os mini-batches foram de 2000. Para os outros modelos essa configuração seguiu a mesma ideia, embora tenha sido acrescido alguns detalhes a mais como rotação horizontal aleatória.

Após as experimentações sobre o modelo Inception proposto, os valores de perda começaram a convergir perto da casa dos 120.000 passos, o que é um valor bem alto.

Fig. 6. Rampa de descida do valor de perda para o treino.

Com a finalização do treino foi observado que o menor valor de perda para o treino foi de 0.024 e valor de acurácia em 91.43% isso foi atingido após os 134.000 passos do algoritmo. Na validação, o modelo apresentou um valor de perda bem menor, algo em torno de 0.02 com um valor médio de acurácia em 83.86%, a classe com maior acurácia foi ”car”e ”truck”ambas com aproximadamente 91.5%, já as menores foram ”cat”e ”dog”, algo que pode ser devido ao formato semelhante desses animais. A seguir é apresentada a Tabela I que contempla todas as categorias e seus respectivos desempenhos.

Tabela I
TABELA DO DESEMPENHO DAS CLASSES

Acuracy of plane	87 %
Acuracy of car	94 %
Acuracy of bird	82 %
Acuracy of cat	84 %
Acuracy of deer	86 %
Acuracy of dog	77 %
Acuracy of frog	83 %
Acuracy of horse	84 %
Acuracy of ship	87 %
Acuracy of truck	94 %

Partindo para os outros modelos o VGG não obteve um desempenho tão satisfatório, no treinamento, por exemplo, o modelo acabou ficando em valores de perda de 0.51 e acurácia de 54.46%, na validação os valores seguiram uma tendência constante e registraram 1.69 para perda e 51.52% para acurácia, os comportamentos podem ser visualizados na Figura 8. Pode-se observar que após a terceira época o valor de acurácia começa a diminuir progressivamente.

Comportamento diferente foi observado no modelo da ResNet, porque os valores foram baixos para perda e altos para acurácia, na última época do treinamento os valores de perda registrados foram de 0.00, e a acurácia atingindo 96.35%,

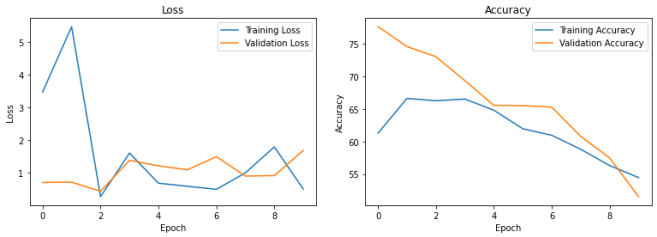

Fig. 7. Valores de perda e acurácia na VGGNet.

significando uma boa generalização do modelo. Na validação o modelo conseguiu bater em perda 0.04 e 82.54%, valor de acurácia, bem próximo do Inception proposto, mas com o valor de perda bem menor se comparado aos dois anteriores.

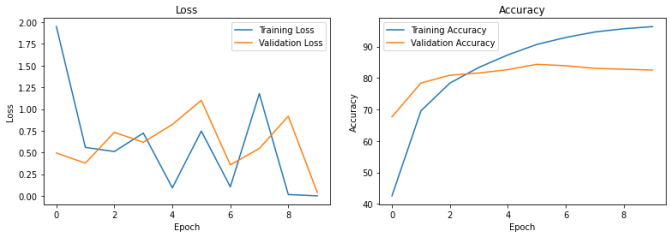

Fig. 8. Valores de perda e acurácia na ResNet.

Um comparativo dos desempenhos de acurácia entre os modelos Inception-E, VGGNet e ResNet é apresentado na Figura 9, sendo a VGGNet a que teve o menor desempenho para o cenário que foi executado, não significa que a arquitetura não possa ter um melhor desempenho apresentados outros cenários.

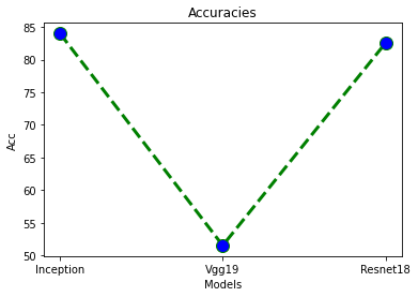

Fig. 9. Comparação entre os modelos.

O modelo Inception-E se sobressai por uma margem pequena de 1.32% em relação a ResNet como observado e consegue por uma diferença de 32.34% sobre a VGG, sendo a ResNet o modelo com menor valor de perda dentre as três.

V. Conclusão

O resultados mostraram que o modelo Inception-E é competitivo, seguindo a ideia de tornar mais esparso assim seguiu, como consequência foi reduzido o uso de parâmetros, com a aceleração do treinamento, uma das vantagens desse modelo é que ele produz bons resultados com modesto uso de recursos quando comparado a modelos maiores. Um dos possiveis trabalhos futuros é o incremento de um braço com kernel 7x7 fatorizado com kernels de 3x3, que poderiam auxiliar na melhora de desempenho do modelo.

Referências

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[2] J. Chen, Z. Wan, J. Zhang, W. Li, Y. Chen, Y. Li, and Y. Duan, "Medical image segmentation and reconstruction of prostate tumor based on 3d alexnet," *Computer Methods and Programs in Biomedicine*, vol. 200, p. 105878, 2021.

[3] R. Wang, J. Xu, and T. X. Han, "Object instance detection with pruned alexnet and extended training data," *Signal Processing: Image Communication*, vol. 70, pp. 145–156, 2019.

[4] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen, "Synthesizing training images for boosting human 3d pose estimation," in *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, pp. 479–488.

[5] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," 01 2013, pp. 1–9.

[6] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[9] R. E. V. d. Silva, "Um estudo comparativo entre redes neurais convolucionais para a classificação de imagens," 2018.

[10] Y. Movshovitz-Attias, Q. Yu, M. C. Stumpe, V. Shet, S. Arnoud, and L. Yatziv, "Ontological supervision for fine grained classification of street view storefronts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1693–1702.

[11] P. Nousi, E. Patsiouras, A. Tefas, and I. Pitas, "Convolutional neural networks for visual information analysis with limited computing resources," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 321–325.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567

[15] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016. [Online]. Available: http://arxiv.org/abs/1602.07261

[16] E. Bisong, "Google colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer, 2019, pp. 59–64.