Experimentos com a Tree-CNN para Mitigar o Esquecimento Catastrófico

Everton Lima Aleixo[1]

[1]Instituto de Informática – Universidade Federal do Amazonas (UFAM) – Manaus – AM – Brazil Instituto de Computação

{everton.aleixo@icomp.ufam.edu.br

Abstract. *Neural networks has outperform humans in many task in last years. Despite that, humans has a hability that theses networks does not have, the capacity of continuos learning without forgetting old skills. Many efforts has been done in this direction, but almost all works present only experiments in easy datasets. For this reason, this paper aim to present tests in a more sophisticated dataset on a state of art technique proposed to provide the capacity of continuos learning, Tree-CNN.*

Resumo. *As redes neurais superaram os humanos em muitas tarefas nos últimos anos. Apesar disso, os humanos têm uma habilidade que essas redes não possuem, a capacidade de continuar aprendendo sem esquecer as habilidades antigas. Muitos esforços foram feitos nessa direção mas quase todos os trabalhos apresentam apenas experimentos em conjuntos de dados simples. Por esse motivo, este artigo tem como objetivo apresentar testes em um conjunto de dados mais sofisticados sobre uma técnica do estado da arte proposta para fornecer a capacidade de aprendizado contínuo, Tree-CNN.*

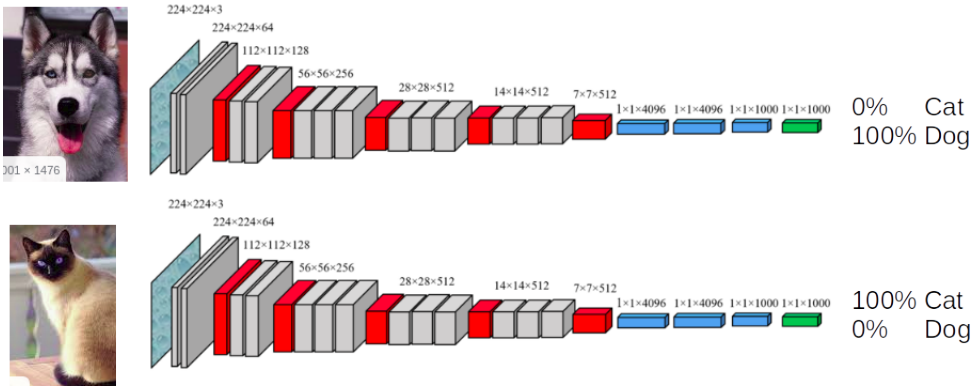
1. Introdução

Modelos de redes neurais profundas tem apresentado resultados fantásticos em diversas tarefas específicas, como por exemplo, na identificação de objetos em imagens [Oord et al. 2016] e geração de áudio [LeCun et al. 2015]. Diversos novos modelos são desenvolvidos e publicados anualmente para atingir uma acurácia ainda maior ou consumir menos recursos [Devlin et al. 2018, Tian et al. 2019].

Esses modelos precisam de uma fase de treinamento *offline*. Nesta fase, os modelos são treinados com amostras rotuladas, como ilustrado na Figura 1. Essas amostras devem apresentar uma distribuição mais similar possível das encontradas em fase de produção para poder ter uma boa acurácia [Goodfellow et al. 2013].

Entretanto, no mundo real, a premissa de termos amostras de todas as tarefas que precisam ser resolvidas ao passar do tempo não pode ser assumida. As amostras tendem a mudar significativamente suas distribuições além de novas classes poderem aparecer para resolver uma nova tarefa, por exemplo, pode ser necessário que o modelo da Figura 1 identifique carros. Dessa forma, se faz necessário um aprendizado contínuo e incremental [Polikar et al. 2000].

Figure 1. Fase de treinamento de uma rede neural. No topo é apresentado uma imagem de cachorro e é informado à rede que isso representa um cachorro. O mesmo processo ocorre na parte inferior mostrando padrões de um gato. Com isso, o modelo tende a aprender que imagens que seguem essas distribuições pertencem a essas classes.



Quando modelos, mesmo muito bons, são submetidos ao aprendizado contínuo e não apresentam nenhuma técnica ou mecanismo que viabilize esse tipo de aprendizado, eles sofrem de um fenômeno conhecido por **esquecimento catastrófico** [McCloskey and Cohen 1989].

Muitas pesquisas nessa área vem sendo realizada para reduzir esse fenômeno, como por exemplo [Li and Hoiem 2017, Mellado et al. 2017, Imai and Nobuhara 2018, Roy et al. 2019], porém avaliação desses métodos ainda é algo a ser melhorado. Em [Kemker et al. 2018] é proposto que esses métodos sejam avaliados em conjunto de dados mais complexos do que apenas em conjuntos de dados simples como, por exemplo, MNIST [LeCun et al. 1998] e CIFAR [kri]. Além disso, ele avalia diversos trabalhos desenvolvidos até 2018.

Este trabalho tem como objetivo avaliar o método proposto em [Roy et al. 2019] utilizando o conjunto de dados proposto em [Kemker et al. 2018], o CUBs-200 [Wah et al. 2011] por ter sido um método proposto após a publicação de [Kemker et al. 2018] e reportar resultados do estado da arte.

O restante desse trabalho esta organizado da seguinte forma: na Seção 2 é apresentado o modelo proposto em [Roy et al. 2019]; na Seção 3 é apresentado a base de dados utilizada e a metodologia de avaliação, como proposto em [Kemker et al. 2018]; os resultados dos testes empíricos são apresentados em 4; e por fim, na Seção 5 são apresentados as considerações finais.

2. Tree-CNN

O modelo proposto por [Roy et al. 2019] utiliza uma estrutura hierárquica baseada em árvores, onde cada nó possui uma rede neural convolutiva. Essa rede recebe como entrada uma imagem e tem como saída um classificador *softmax* de tamanho n.

Esse n representa quando filhos este nó da árvore apresenta. Cada filho pode ser um rótulo, que representa uma classe, ou um novo nó que terá as mesma propriedades. Nas seções seguintes são apresentadas as principais rotinas da Tree-CNN.

2.1. Aprendizado Contínuo na Tree-CNN

Quando chegam *M* novas classes com *I* imagens à serem aprendidas pelo modelo, as imagens dessas classes são segregadas por classe. Cada classe é processada pelo nó corrente, inicialmente a raiz. Isso produz uma matriz de dimensão $G^{K,I}$, quando todas as classes são produzidas temos um cubo de dimensões $O^{K,M,I}$, que presenta como o nó classificaria as imagens dessas novas classes.

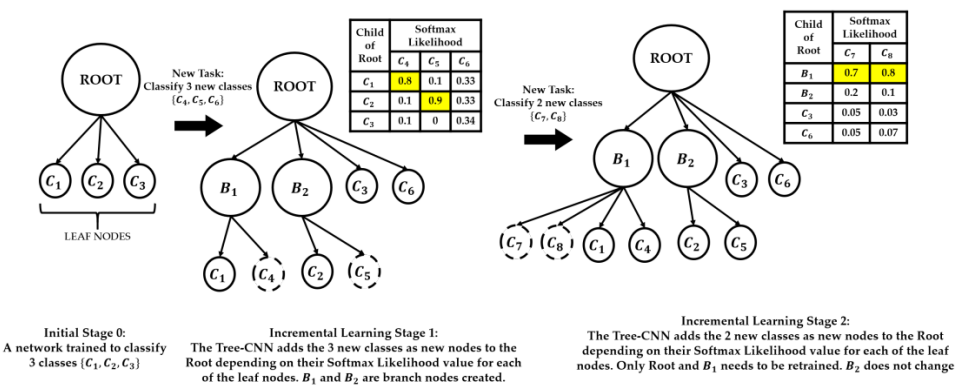
Desse cubo, é extraído a média das inferências das imagens, gerando uma nova matriz de dimensões $O^{K,M}_{avg_i}$. Com esta matriz é gerado uma nova matriz baseada no *softmax*, $L^{K,M}$. Essa nova matriz representa o *likelihood*, ou seja as classes de *K* que são mais semelhantes a cada classe de *M*.

Com cada coluna dessa matriz $L^{K,M}$ é formado um elemento que é adicionado uma lista *S*. Cada elemento possui os seguintes atributos: *label* (rótulo da classe que a coluna representa); *value* (os três maiores valores da coluna em ordem decrescente); *nodes*(indices dos três valores de *value*, que representam as classes que o modelo acredita serem mais parecidas com a nova classe que esta sendo adicionada). Por fim, a lista *S* é ordenada pelo maior valor da primeira posição do vetor *value*.

Essa lista é esvaziada elemento a elemento. Cada vez que um elemento é retirado do começo da lista, a classe desse elemento é adicionada ao modelo seguindo uma das três ações abaixo:

1. $(S[0].value[0] - S[0].value[1]) < \alpha$: A nova classe tem muita afinidade ao *nodes[0]*, por tanto é adicionado a ele. Se o *nodes[0* for um rótulo, é criado um novo nó e o novo rótulo é adicionado a este juntamente com o rótulo de *nodes[0]*.
2. $((S[0].value[0] - S[0].value[1])AND(S[0].value[1] - S[0].value[2])) < \beta$: Significa que a nova classe tem afinidade tanto pelo *nodes[0]* quanto pelo *nodes[1]*. Nesse caso, se *nodes[1]* for um rótulo e *nodes[0] não estiver cheio*, ele é agregado ao *nodes[0]* (*merge*) e a nova classe é inserida em *nodes[0]*. Caso contrário, o novo rótulo é adicionado no *nodes* que possuir menos rótulos.
3. *Em último caso*: o nó cresce horizontalmente, sendo adicionado um novo filho ao nó corrente. Esse novo filho é um nó folha com o rótulo da nova classe.

Figure 2. Figura extraída de [Roy et al. 2019] para exemplificar o processo de adição de classes no modelo Tree-CNN.



Após a inserção da nova classe, a coluna que era representada pelo elemento é removida de *L*, assim como as linhas que representam os filhos do nó corrente que já estão "cheios". Então a lista *S* é regerada e o processo se repete. Quando o processo tenta adicionar a nova classe a um filho que aponta para um outro nó, esse elemento é processado da mesma forma utilizando o nó filho como nó corrente, até que essa nova classe se encaixe como um nó folho, ou seja, um rótulo.

A Figura 2 mostra um exemplo simples e pequeno de adição de 2 grupos de classes em um modelo que já foi criado conhecendo C_1, C_2 e C_3. Na figura podemos ver a transformação do modelo inicial para o momento após inserir as classes C_4, C_5 e C_6 e no canto direito após uma segundo inserção de classes, as C_7 e C_8.

2.2. Treinamento e Inferência na Tree-CNN

Tanto para o treinamento quanto para a inferência no modelo Tree-CNN o processo ocorre por nó, sempre iniciando da raiz. Seja *Q* um conjunto de imagens e *E* seus respectivos rótulos.

Durante o treinamento, o nó utiliza um mapeador de grupos para rótulos para transformar *E* em *E'*, onde os rótulos de mesmo grupo para o nó em questão são agrupados para terem o mesmo rótulo. Por exemplo, suponha que a raiz tenha dois nós filho, o *zero* que representa os rótulos 0 e 1 e o *um* que representa o rótulo 2. Todos os rótulos 0 e 1 de *E* são transformados em 0 em *E'*; e todos os 2 são transformados em 1. Então a rede do nó é treinada com *Q* e *E'*. As imagens de *Q* que possuírem rótulos em *E'* que levam para um nó não folha, são passados para o nó filho juntamente com seus rótulos originais em *E*. Dessa forma, os nós filhos também são treinados.

Para a inferência, as imagens em *Q* são processadas pela rede do nó, as que são inferidas como nós folhas, rótulos, são classificadas como tal. As que são classificadas como nós não folha, as imagens são passadas para o nó filho alvo e este reprocessa esse subconjunto de imagens para inferir um rótulo. O processo se repete recursivamente até que todas as imagens sejam rotuladas. Na próxima seção é apresentado como o modelo será avaliado e a base de dados ao qual será submetido.

3. Base de dados e Metodologia

Conforme sugerido por [Kemker et al. 2018], neste trabalho será utilizado uma base de dados mais complexa, a base *Caltech-UCSD Birds-200-2011* [Wah et al. 2011].

Essa base de dados é uma versão estendida do conjunto de dados CUB-200 [Welinder et al. 2010], com aproximadamente o dobro do número de imagens por classe e novas anotações. Na Figura 3 é apresentado algumas amostras das imagens desta base de dados. Ela pode ser sumarizado da seguinte forma:

- **Número de categorias**: 200;
- **Número de imagens**: 11.788.

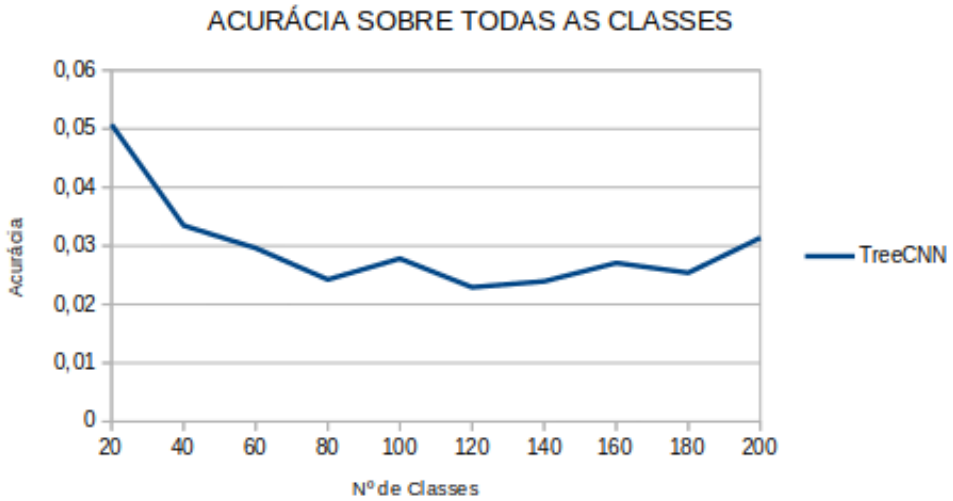
Para o treinamento do modelo, a base de dados será dividida em 10 grupos de 20 classes. Cada grupo será adicionado ao modelo sequencialmente e retreinado com as classes aprendidas até aquele momento. Ao final das 200 classes terem sido inseridas, o modelo será avaliado pela acurácia em todas as classes.

Figure 3. Exemplos de imagens da base de dados *baseCaltech-UCSD Birds-200-2011*.

Os hiper-parâmetro α e β são definidos com 0.1, assim como proposto por [Roy et al. 2019]. O modelo de rede neural utilizado em cada nó será uma rede neural convolucional profunda com a arquitetura da ResNet [He et al. 2016]. Como otimizador esta sendo utilizado o *SGD* com um fator de aprendizado de 0.01, decaimento de 1^{-6} e momento de 0.9. Como função de perda foi adotado a entropia cruzada de categorias. Cada treinamento foi realizado utilizando uma estratégia de parada antecipada com o parâmetro de *paciência* configurado para 15.

4. Resultados

Figure 4. Acurácia da TreeCNN a medida que novas classes são inseridas.



Conforme esperado, o modelo tende a não esquecer o conhecimento prévio por durante as novas fases de aprendizado ele ter acesso a amostras das classes antigas.

Na Figura 4 é apresentado o gráfico da acurácia do modelo a medida em que novas classes vão sendo adicionadas. Por estar usando uma estratégia de parada antecipada o

número de épocas variou de acordo com a adição de novas classes, porém nenhuma vez passou de 20 épocas.

A acurácia ficou realmente muito ruim, mostrando que a rede interna não conseguiu aprender nesse conjunto de dados. Novos modelos podem ser explorados para tentar melhorar os resultados.

5. Considerações Finais

Apesar do sucesso das redes neurais artificiais na tarefa de classificação de imagens, esses modelos ainda sofrem muito com o esquecimento catastrófico quando submetidos a uma aprendizagem contínua, principalmente quando novas classes são adicionadas sob demanda.

Existem diversas pesquisas para desenvolver métodos e modelos que reduzam os efeitos desse problema [Li and Hoiem 2017, Mellado et al. 2017, Imai and Nobuhara 2018]. Porém, segundo [Kemker et al. 2018], eles não são avaliados de forma uniforme ou são avaliados apenas em conjuntos de dados muito simples como o MNIST [LeCun et al. 1998].

Neste trabalho, foram realizados teste empíricos no modelo [Roy et al. 2019], que foi apresentado após o trabalho de [Kemker et al. 2018], seguindo as diretrizes dele. A implementação em python do modelo pode ser encontrado no Github [1].

Testes realizados no conjunto de dados *baseCaltech-UCSD Birds-200-2011* mostrou uma acurácia fraca mesmo no primeiro subconjunto dos 10 inseridos sequencialmente. Isso mostra que é um conjunto de dados bem mais complexo. Novos modelos de redes podem ser explorados para tentar melhorar os resultados.

O modelo proposto por [Roy et al. 2019] apesar de mitigar os efeitos do esquecimento catastrófico ele requer que amostras das classes aprendidas sejam mantidas no sistema para que ao aprender novas, possa utilizar as imagens das classes antigas durante a fase de retreino. Essa técnica é conhecida como *rehearsal* e seu maior problema é o consumo de recursos.

Um próximo passo de melhoria neste modelo, como sugerido pelos próprios autores, é criar um sistema complementar de aprendizado para realizar um *pseudo-rehearsal*. Dessa forma, um outro modelo iria aprender a gerar amostras que seguissem as distribuições das classe já aprendidas, não sendo necessário manter amostras. Outras arquiteturas de redes neurais também podem ser avaliadas para compor o nó da *Tree-CNN*.

References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

[1] $https://github.com/evertonaleixo/Tree-CNN/blob/master/Tree\_CNN.ipynb$

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Imai, S. and Nobuhara, H. (2018). Stepwise pathnet: Transfer learning algorithm to improve network structure versatility. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 918–922. IEEE.

Kemker, R., McClure, M., Abitino, A., Hayes, T. L., and Kanan, C. (2018). Measuring catastrophic forgetting in neural networks. In *Thirty-second AAAI conference on artificial intelligence*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Mellado, D., Saavedra, C., Chabert, S., and Salas, R. (2017). Pseudorehearsal approach for incremental learning of deep convolutional neural networks. In *Latin American Workshop on Computational Neuroscience*, pages 118–126. Springer.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Polikar, R., Udpa, L., Udpa, S., and Honavar, V. (2000). An incremental learning algorithm for supervised neural networks. *IEEE Trans. on SMC (C), Special Issue on Knowledge Management*.

Roy, D., Panda, P., and Roy, K. (2019). Tree-cnn: A hierarchical deep convolutional neural network for incremental learning. *Neural Networks*.

Tian, Z., Shen, C., Chen, H., and He, T. (2019). FCOS: Fully convolutional one-stage object detection. In *Proc. Int. Conf. Computer Vision (ICCV)*.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The Caltech-UCSD Birds-200-2011 Dataset.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. (2010). Caltech-UCSD Birds 200. (CNS-TR-2010-001).