Campinas, 28 de Junho de 2019
Campinas, Brazil, June 28, 2019

Comparação de Classificadores Covolucionais em Imagens de Células Contaminadas por Malária

Eneias F. R. Júnior e Lucas C. Lima

Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
CEP 13083-852 – Campinas, SP, Brasil

e170486,l182411@dac.unicamp.br

Abstract – This article intends to show the differences of performance between Mobile Net, VGG16 and a created convolutional architecture in the case of classifying infected Malaria cells.These architectures listed above are very famous architectures in computer vision, used in a large contexts solving problems of image classification. However, large nets, like VGG16, have some trouble with convergence in cases that the dataset is not that large. The proposal is that a small CNN can perform better than these famous CNN's, either in accuracy or in computational cost.

Keywords – Machine Learning, Classification, CNN, Malaria, Cells

1. Introdução

A malária é uma doença que atinge diretamente as células de uma pessoa que foi picada pelo mosquito vetor da doença. Os sintomas começam com uma simples febre podendo chegar a uma insuficiência renal e em, casos extremos, a morte. O trabalho realizado em todo mundo para a profilaxia, tratamento e identificação da malária reúne esforços em todas as áreas. Partindo desse contexto surgiram abordagens em visão computacional que buscavam resolver o problema de classificação de células infectadas com malária.

Com tantas arquiteturas consolidadas e que performaram tão bem em tarefas realmente difíceis de classificação de imagens, é natural aplicar estas em outras tarefas, como a situação em questão.

2. Proposta

Seguindo a liturgia de surgimento de novas redes, a proposta inicial foi criar uma nova arquitetura que performe bem um conjunto de dados e, em seguida, comparar o desempenho com outras redes e comparar acurácia, custo computacional, aplicabilidade e capacidade de generalização.

2.1. Pré Processamento dos Dados

Os 27 mil dados foram armazenados localmente e sofreram um processo de *data augmentation*. As imagens foram salvas rotacionadas de 30 e 60 graus da forma com que foram salvas, além da sua orientação original, na busca de eliminar uma espécie de inadequação do modelo ao identificar errado células que foram capturadas em um ângulo diferente das outras. Outra coisa que poderia ter sido feita, em busca de reduzir custo computacional, é trabalhar com as imagens em escala de cinza. Isso reduziria o custo computacional por fator quadrado, mas não pareceu inteligente, uma vez que é clara a diferença de cores entre as células contaminadas e as saudáveis.

2.2. Processo de Criação

O banco de dados utilizado se encontra na plataforma de desafios de ciência de dados *Kaggle* [1]. Lá também existem trabalhos de outras pessoas na tentativa de explorar os dados com diversas abordagens.

Após observar algumas soluções utilizando redes convolucionais, a criação de uma rede convolucional com 4 camadas convolucionais, número crescente de filtros e número decrescente do tamanho da máscara convolucional se mostrou excelente no custo benefício entre custo computacional e acurácia. O modelo em questão performa no banco de dados da seguinte forma: Os seus filtros possuem kernels maiores no começo, o que permite a ele a análise de células deformadas em suas membranas pelo vírus da malária.

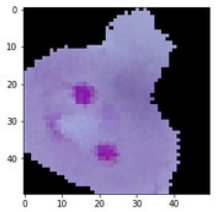

Figura 1. Célula infectada com Malária

Conforme as camadas aprofundam na imagem, aumenta-se o número de filtros ao passo que se diminui o tamanho da máscara convolucional. Essa

estratégia permite encontrar os padrões de anéis roxos que se formam nas imagens.

A última camada é uma camada de classificação binária com dois neurônios. As classes foram utilizadas como variáveis categóricas, portanto, a classificação se deu em dois neurônios.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 50, 50, 16)	208
max_pooling2d_1 (MaxPooling2	(None, 25, 25, 16)	0
conv2d_2 (Conv2D)	(None, 25, 25, 32)	2080
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 32)	0
conv2d_3 (Conv2D)	(None, 12, 12, 64)	8256
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 64)	0
conv2d_4 (Conv2D)	(None, 6, 6, 128)	32896
max_pooling2d_4 (MaxPooling2	(None, 3, 3, 128)	0
dropout_1 (Dropout)	(None, 3, 3, 128)	0
flatten_1 (Flatten)	(None, 1152)	0
dense_1 (Dense)	(None, 500)	576500
dropout_2 (Dropout)	(None, 500)	0
dense_2 (Dense)	(None, 2)	1002

Total params: 620,942
Trainable params: 620,942
Non-trainable params: 0

Figura 2. Arquitetura do Modelo Criado

É válido perceber que o modelo é relativamente pequeno. Com pouco mais de 620 mil parâmetros ajustáveis, a arquitetura não se mostra sofisticada com relação a evitar overfitting, como camadas de dropout no meio da arquitetura ou camadas de batch normalization. Ainda é importante ressaltar que, foram testadas algumas funções de ativação para a última camada, mas a melhor perfirmance foi com a função softmax para a camada final e ReLU para as camadas intermediárias. Como otimizador do modelo, foi utilizado Gradiente Adaptativo pelo seu melhor desempenho na maioria das situações. Como função de custo, foi utilizado Entropia Cruzada Binária, por se tratar de um problema de classificação binária.

2.3. Treino e Teste

Durante o processo de treino, a primeira coisa que é notória é que, mesmo com 38581 dados, o modelo demora muito pouco em cada época, mesmo utilizando uma unidade de processamento gráfica de apenas 2Gb reservados(conforme imagem 3). A segunda coisa que chama muito a atenção é a velocidade da convergência do modelo, que já atinge

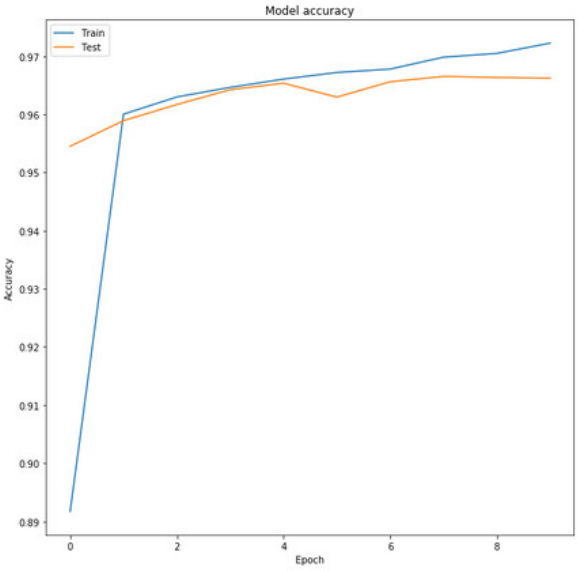

Figura 3. Acurácia ao longo das épocas utilizando o modelo criado

95,89 por cento de acurácia no conjunto de teste na segunda época.

2.4. Outros modelos

O modelo VGG16 [3], foi exposto ao mesmo dataset, com os mesmos otimizadores, batch size e número de épocas. Entretanto, o modelo sequer convergiu. A VGG16 possui pouco mais de 14.7 milhões de parâmetros otimizáveis, porém as suas épocas duraram mais de 20 minutos e não houve progresso na métrica, que estagnou em 56,88 por cento, nem na função de custo.

Por outro lado, o modelo que fez frente ao criado foi a MobileNet [2]. Criada para performar tão bem quanto possível quanto as famosas AlexNet, ResNet e a própria VGG16, a MobileNet tem a proposta de conseguir economizar custo computacional e, ainda assim, manter uma capacidade de classificação tão boa quanto as outras. O seu "truque"consiste em uma camada convolucional que opera ponto a ponto, assim, economizando na complexidade do algoritmo e mantendo a função da convolução. Ela possui dois hiperparâmetros ajustáveis, que transformam a MobileNet em redes ainda menores.

A rede utilizada para comparação neste trabalho é a rede que vem por padrão no *Keras API*, utilizando *Tensorflow* como *backend*. Ela consta de 3.209 milhões de parâmetros treináveis, com a seguinte arquitetura.

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figura 4. Arquitetura Mobile Net [2]

Entretanto, a rede foi importada sem a última camada, onde foi modificada para uma camada de classificação binária e utilizando softmax como função de ativação.

O desempenho no dataset de treino e validação converge muito bem, mesmo com cerca de 5 vezes o número de parâmetros otimizáveis. Nota-se algumas mudanças na acurácia do banco de validação após algumas épocas, mas o modelo se mostra consistente, performando acima dos 90 por cento logo após a terceira época.

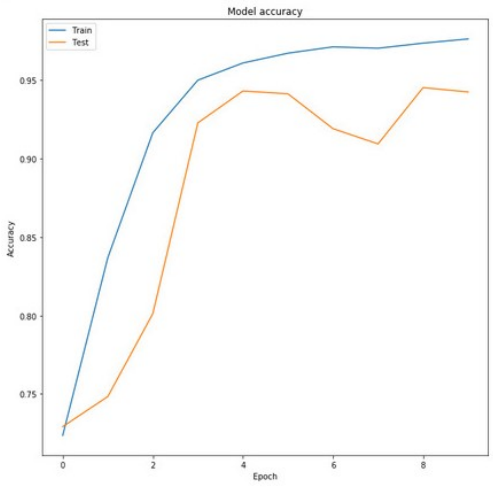

Figura 5. Acurácia ao longo das épocas utilizando a Mobile Net

2.5. Resultados

A avaliação dos modelos se deu, então, no banco de dados de teste, que corresponde a 20 por cento do dataset original. Os modelos foram avaliados sob duas perspectivas: acurácia e matriz de confusão. De ambas perspectivas, o modelo criado supera a MobileNet. Com respeito a acurácia, o modelo criado conseguiu 96.48%, enquanto a MobileNet ficou com 94.00%. Na matriz de confusão, onde é possível observar como é o desempenho do modelo por classe, o modelo criado classifica corretamente os saudáveis com 96.19% de precisão, enquanto classifica corretamente os infectados com 96.90% de precisão. Já a MobileNet performa com 95.62% de precisão nos saudáveis, mas com 91.81% nos infectados.

O Keras permite visualizar o que está sendo processado por cada filtro convolucional, assim, sendo possível inferir porque uma arquitetura consegue classificar melhor um caso ou outro. Os filtros das primeiras camadas operam de forma parecida em ambos os modelos, conforme as figuras abaixo:

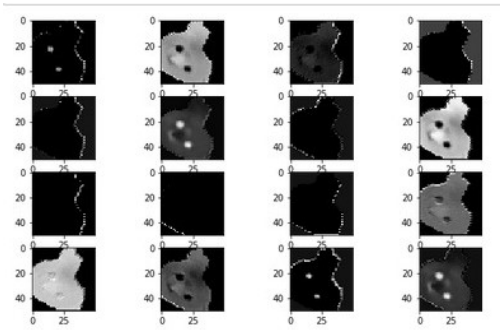

Figura 6. Imagens na primeira camada da Nova Arquitetura

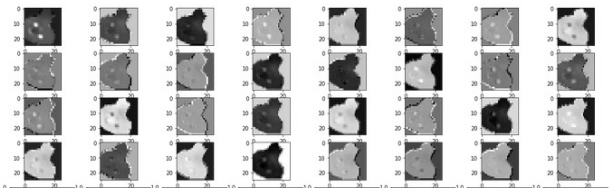

Figura 7. Imagens na primeira camada da Mobile Net

Porém, algumas camadas abaixo, é notório que a MobileNet passa a interpretar padrões que não são mais inteligíveis, humanamente falando, enquanto a rede criada ainda avalia os padrões mais internos da célula. Se tratando de Deep Learning, não é possível afirmar que, pelo fato de um dado não ser inteligível por humanos, ele se torna sem sentido, mas é passível de questionamento a validade da profundidade de um modelo para dados deste tamanho.

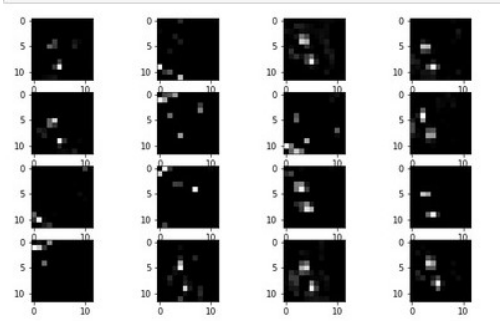

Figura 8. Imagens na última camada da Nova Arquitetura

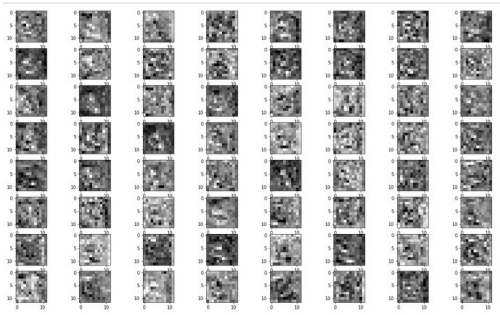

Figura 9. Imagens da 20ª camada da Mobile Net

2.6. Citações e Referências Bibliográficas

As citações devem ser por referência numérica e as referências devem ser completas e uniformes, organizadas pela ordem alfabética do sobrenome.

3. Análises e Trabalhos Futuros

A primeira análise a ser feita é: modelos muito grandes podem não convergir, mesmo com muitos dados. Várias mudanças foram feitas na VGG16 na tentativa de fazê-la convergir, desde mudar a função de ativação a mudar o otimizador, todas falharam. Então, supõe-se a razoabilidade de o modelo ser incapaz de aprender com dados de tamaho reduzido, visto que as imagens são 50x50. É compreensível, contudo, que a rede não seja capaz de classificar bem estes dados, uma vez que ela foi idealizada para outro contexto.

Também é possível observar que, nem sempre mais camadas significam melhor desempenho. A MobileNet é mais que 5 vezes maior que o modelo criado, porém, tem um desempenho menor na tarefa de classificação das células. Diferente da arquitetura da VGG16, esta possui muito menos parâmetros e converge sem problemas, então a sua deficiência, em relação ao modelo criado, provavelmente está na forma com que estão organizados os Kernels e os filtros. A quantidade de filtros e as convoluções ponto a ponto podem fazer a rede perceber falsos padrões na imagem.

As sugestões de trabalhos futuros são:

- Adicionar camadas de dropout na arquitetura da MobileNet, na tentativa de evitar uma espécie de overfitting causada pela própria arquitetura;

- Tentar implementar a MobileNet pré treinada no banco de dados da ImageNet e fazer transfer learning;

- Adcionar, ao modelo criado, camadas de dropout e de batch normalization;

4. Conclusões

É possível concluir, portanto, que, para tarefas específicas, como a classificação das células em questão, existem arquiteturas que devem performar melhor a depender do banco de dados em questão. Ultrapassando, inclusive, o desempenho de arquiteturas famosas.

Referências

[1] Malaria cell images dataset. `https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria`. (acessado em 28/06/2019).

[2] Andrew G. Howard. Mobilenets: Efficient convolutional neural networks for mobile visionapplications. `https://arxiv.org/pdf/1704.04861.pdf`. (acessado em 28/06/2019).

[3] Karen Simonyan. Very deep convolutional networks for large-scale image recognition. `https://arxiv.org/pdf/1409.1556.pdf`. (acessado em 28/06/2019).