Projeto Final

Nanodegree Engenheiro de Machine Learning

Guilherme Gonçalves Bonaldo

10/05/2018

Detecção de tumores em imagens histológicas de mamas utilizando redes neurais profundas

1. Definição

1.1. Visão Geral do Projeto

A melhora na capacidade de processamento de computadores nos últimos anos, unida com o grande avanço da área de visão computacional (principalmente pelo desenvolvimento de Redes Neurais Profundas), fazem com que computadores participem cada vez mais de soluções de reconhecimento de padrões e classificação de imagens.

Na área médica, diversos diagnósticos são resultados de análise de imagens, sejam elas raios-x, ultrassons, ressonâncias magnéticas ou histologias. Redes Neurais profundas já estão sendo utilizadas na área, com resultados que muitas vezes ultrapassam a capacidade humana de classificação [1].

Este trabalho tem como objetivo treinar, por meio da técnica de *transfer learning* de uma rede neural convolucional (RNC) com arquitetura VGG16 pré-treinada no dataset ImageNet, para encontrar a presença de câncer de mama em imagens de histologias mamárias, com o objetivo de ajudar diagnósticos médicos para esse tipo de doença. Como entrada, serão utilizadas imagens de histologias mamárias, que serão interpretadas pela rede, que por sua vez responderá uma probabilidade da imagem conter a doença.

1.2. Descrição do Problema

O câncer de mama é o segundo tipo mais frequente no mundo e o mais frequente na população feminina brasileira. Segundo estimativas do INCA, em 2012 e 2013, deverão ocorrer 52.680 casos novos de câncer de mama, com um risco estimado de 52 casos por 100 mil mulheres. Sendo considerado, em geral, um câncer de bom prognóstico quando diagnosticado e tratado precocemente, as taxas de mortalidade por câncer da mama continuam elevadas no Brasil. Provavelmente, essas taxas de mortalidade mantêm-se elevadas porque a doença ainda é diagnosticada em estádios avançados. A sobrevida média após cinco anos na população de países desenvolvidos é aproximadamente 85%. Entretanto, nos países em desenvolvimento, a sobrevida fica em torno de 60%. [2].

Resultados de exames como esse implicam em certos riscos que precisam ser conhecidos [3]:

- O resultado falso positivo pode gerar estresse e ansiedade no paciente, pois este recebe o diagnóstico positivo para a doença e não a possui.
- O resultado falso negativo gera uma falsa segurança no paciente e atraso do tratamento, diminuindo a chance de cura.

Para tentar resolver esse problema, fica claro que um diagnóstico certeiro e precoce é um grande aliado. Com o objetivo de ajudar os profissionais da área a produzirem um diagnóstico correto com ajuda de uma RNC, os seguintes passos serão necessários:

- Obtenção de imagens de histologias mamárias e seus rótulos [4].
- Baixar a arquitetura e pesos da rede VGG-16 pré-treinada no conjunto ImageNet utilizando a biblioteca Keras.
- Variar as camadas da rede que serão congeladas, treinar as restantes em uma versão reduzida do dataset (para otimizar o tempo de treinamento) e então inferir qual rede melhor se adaptou às imagens.
- Com as camadas congeladas ideais para a classificação em questão, treinar a rede com a máxima quantidade possível de imagens (que será limitada pelo processamento do computador) para melhorar a generalização do modelo.
- Realizar um *ensemble* das redes treinadas para verificar o desempenho.
- Disponibilizar um modelo que seja capaz de constatar, com boa precisão, a presença ou não de câncer em uma imagem de histologia mamária.

1.3. Métricas

Para avaliar o modelo, será utilizada a métrica F1 score (1). Nesse caso, pelo fato de o dataset já ser desbalanceado quanto a incidências de câncer, essa métrica já seria uma boa escolha para avaliação do modelo. Além disso, a ocorrência de falsos positivos e falsos negativos é extremamente indesejada, o que ajuda a afirmar a escolha da métrica.

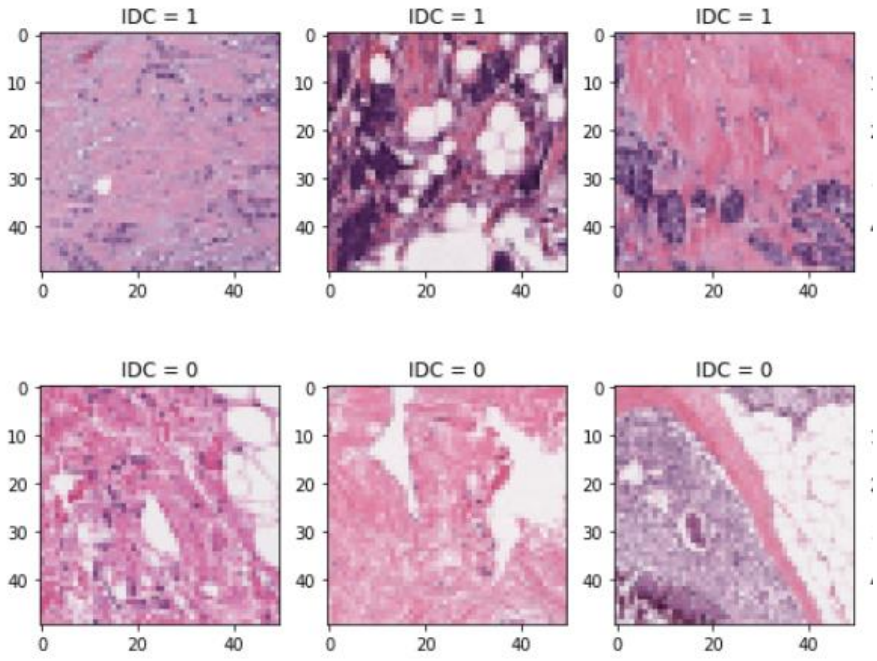
$$F1\ score = 2 * \frac{Precisão * Recall}{Precisão + Recall} \quad (1)$$

2. Análise

2.1. Exploração e visualização dos dados

Como enunciado anteriormente, os dados de entrada serão imagens de histologias mamárias coloridas, de tamanho 50x50 pixels, com diagnósticos positivos e negativos para presença de câncer de mama. O dataset é público e será obtido de [4]. Esse dataset possui mais de 270 mil imagens (.PNG), das quais aproximadamente 78 mil apresentam a presença de câncer. Alguns exemplos dessas imagens são exibidos a seguir:

Figura 1: Exemplos de imagens de histologias com resultados positivos (IDC = 1) e negativos (IDC = 0) para a incidência de câncer.



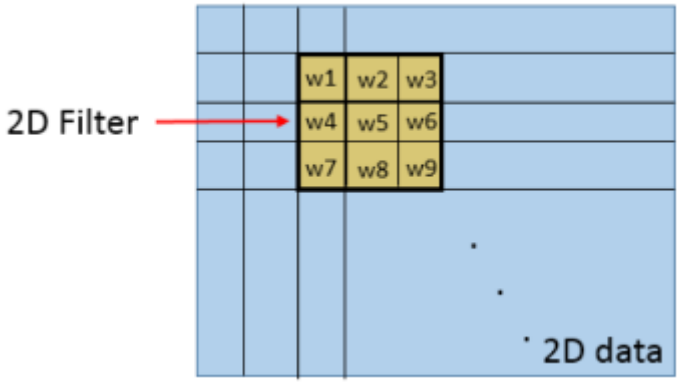
Fonte: Acervo pessoal.

2.2. Algoritmos e técnicas

2.2.1. Rede Neural Convolucional

O principal algoritmo deste trabalho será uma Rede Neural Convolucional (RNC). Esse tipo de rede apresenta blocos convolucionais compostos de sub-blocos, que contém camadas filtros (convolucionais) e por último uma camada de MaxPooling. Cada imagem é tratada como uma matriz de 3 dimensões, são elas a altura, largura e profundidade, esta última representada por camadas de cor, que apresentam um número fixo: 3 (vermelho, azul e verde). Filtros convolucionais de tamanho definido (neste trabalho: 3 pixels por 3 pixels) são janelas que se movem sobre os pixels da imagem na busca por padrões. Os padrões estipulados pelo filtro são o fruto do treinamento da rede, seus pesos, que filtram características da imagem, são iniciados aleatoriamente e algoritmos de otimização, como o gradiente descendente, encontram valores ótimos para essas variáveis. A quantidade de filtros é um parâmetro que define a estrutura da rede.

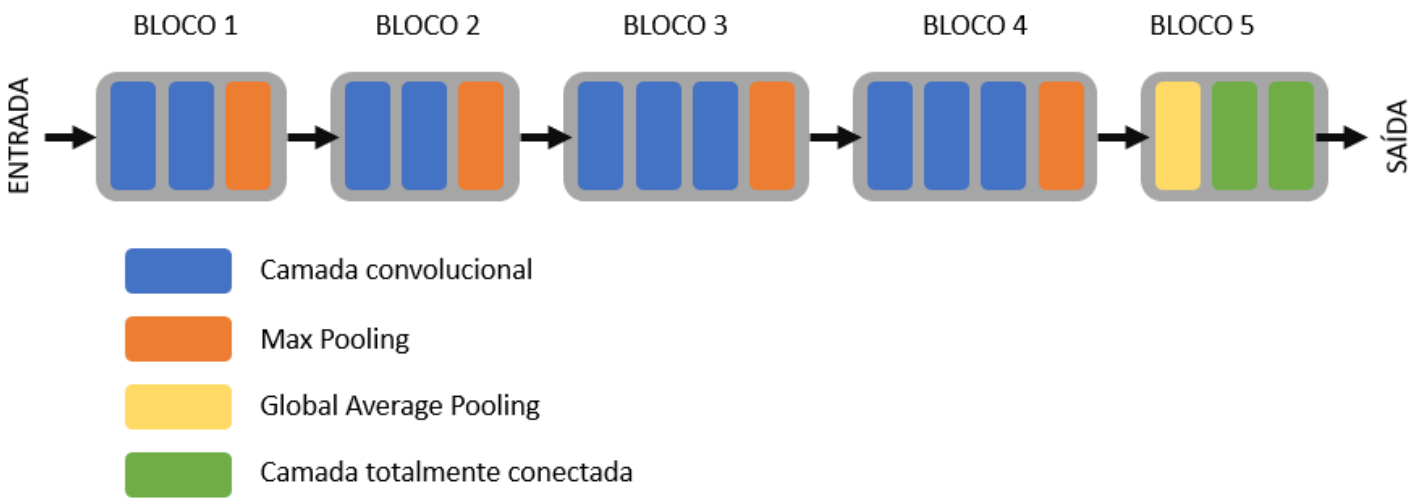
Figura 2: Exemplo de filtro Convolucional.



Fonte: https://github.com/udacity/br-machine-learning/blob/master/projects/capstone/report-example-2.pdf

Após camadas de filtros, ao final de um bloco convolucional, uma camada de MaxPooling é adicionada para enxugar o tamanho da camada convolucional que a antecedeu. Além disso, pode ser entendida como uma camada que filtra informações, selecionando as mais importantes. Na arquitetura utilizada (VGG-16), o último bloco contém um sub-bloco Flatten, que transforma os dados em um vetor de uma dimensão. Esse vetor é totalmente conectado (neste caso) a uma saída, que produz o resultado. A arquitetura VGG-16 é apresentada a seguir:

Figura 3: Arquitetura da rede utilizada neste trabalho.



Fonte: Acervo pessoal

As características mais especificas da rede convolucional utilizada nesse trabalho são apresentadas na tabela 1.

Tabela 1: parâmetros da VGG-16 utilizada.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 50, 50, 3)	0
block1_conv1 (Conv2D)	(None, 50, 50, 64)	1792
block1_conv2 (Conv2D)	(None, 50, 50, 64)	36928
block1_pool (MaxPooling2D)	(None, 25, 25, 64)	0
block2_conv1 (Conv2D)	(None, 25, 25, 128)	73856
block2_conv2 (Conv2D)	(None, 25, 25, 128)	147584
block2_pool (MaxPooling2D)	(None, 12, 12, 128)	0
block3_conv1 (Conv2D)	(None, 12, 12, 256)	295168
block3_conv2 (Conv2D)	(None, 12, 12, 256)	590080
block3_conv3 (Conv2D)	(None, 12, 12, 256)	590080
block3_pool (MaxPooling2D)	(None, 6, 6, 256)	0
block4_conv1 (Conv2D)	(None, 6, 6, 512)	1180160
block4_conv2 (Conv2D)	(None, 6, 6, 512)	2359808

```
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 3, 3, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
=================================================================
Total params: 14,977,857
_________________________________________________________________
```

Fonte: Acervo pessoal

2.2.2. Transfer learning

Com o objetivo de otimizar o tempo de treinamento, será utilizada a técnica de transfer learning. Esse método consiste em obter pesos de uma rede pré-treinada, congelar alguns desses pesos e treinar os pesos restantes da rede. Essa técnica é muito útil por dois principais motivos:

- Economia de tempo e processamento: buscar apenas alguns pesos da rede demanda menos poder computacional pois as técnicas atuais de otimização dos pesos ainda são muito custosas.
- As primeiras camadas das redes convolucionais geralmente criam filtros que identificam padrões muito comuns como frequências, bordas e curvas, portanto, não necessitam ser treinadas todas as vezes do zero, já que todas as imagens apresentam padrões compostos por essas figuras.

2.2.3. Ensemble

Ensemble é uma técnica utilizada para melhorar os resultados e a generalização de modelos. Essa técnica está sendo muito utilizada por profissionais de machine learning, em competições de sites como o Kaggle. Modelos vencedores costumam utilizar dessa técnica [5]. Esse algoritmo combina a previsão de diversos modelos para gerar uma previsão final. Essa técnica será utilizada no presente trabalho a fim de obter-se um modelo mais genérico e acurado.

2.3. Benchmark

O modelo de referência que será utilizado será o artigo [6], no qual o pesquisador utilizou técnicas de processamento de imagem e obteve uma tabela de falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos. Com esses dados é possível calcular o F1 score desse estudo e posteriormente compara-lo com o obtido por esse trabalho final. Também serão utilizados os resultados de dois usuários da plataforma Kaggle, que modelaram o mesmo problema [7] [8]. Além desse modelo, o trabalho será baseado no repositório apresentado durante a Udacity live do pesquisador Fabio Perez [9].

3. Metodologia

3.1. Pré-processamento dos dados

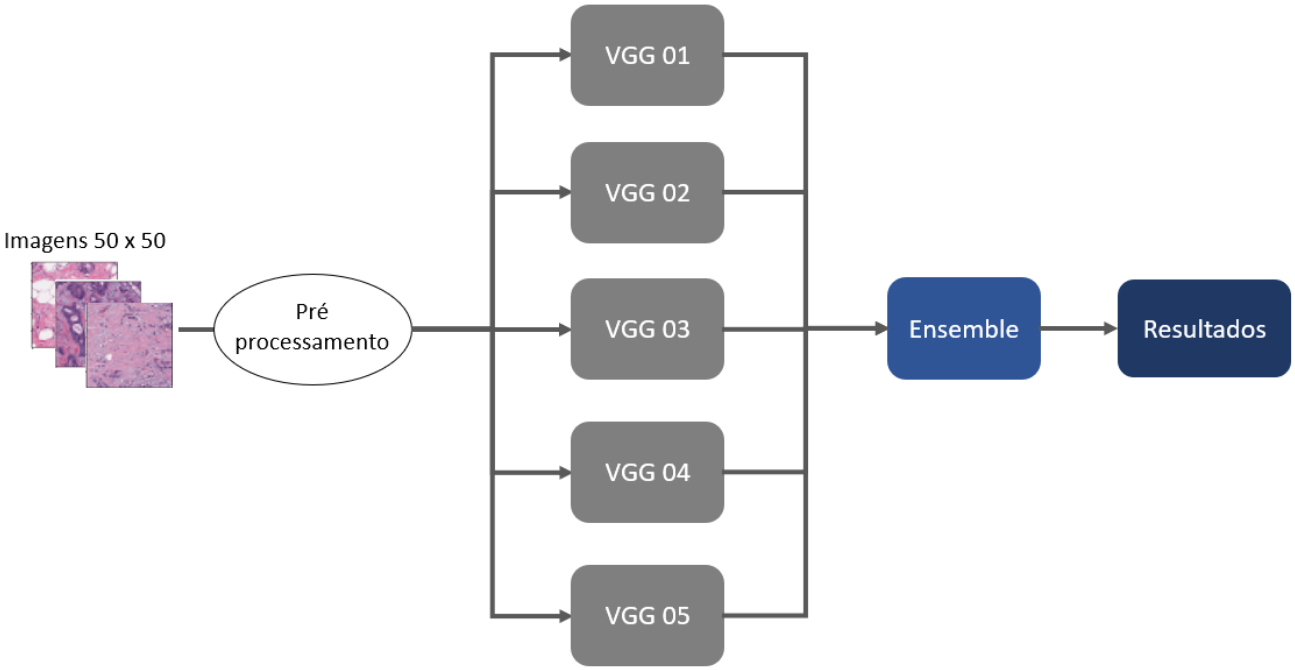
A etapa de pré-processamento consistirá das seguintes etapas:

- Os rótulos das imagens serão obtidos a partir de seus nomes, sendo 1 indicando a presença de câncer e 0 sua ausência.
- As imagens serão divididas em conjuntos de treino, validação e teste, na proporção aproximada de 70%, 15% e 15% respectivamente.
- As imagens serão normalizadas, com a finalidade de obter-se valores de cores de pixels entre 0 e 1.
- As imagens serão transformadas em tensores, para serem processadas em baixo nível pela biblioteca TensorFlow.

3.2. Implementação

O diagrama a seguir mostra a arquitetura do modelo.

Figura 4: Arquitetura do modelo utilizado neste trabalho.



Fonte: Acervo pessoal.

As redes convolucionais treinadas são detalhadas a seguir:

Tabela 2: parâmetros variados nas redes utilizadas.

Rede	Blocos congelados para treinamento	Drop Out	Número de imagens de treino	Número de imagens de validação	Número de imagens de teste
VGG 01	3	0 %	10.000	2.000	5.000
VGG 02	3	25%	10.000	2.000	5.000
VGG 03	1	0 %	10.000	2.000	5.000
VGG 04	2	0 %	10.000	2.000	5.000
VGG 05	3	0%	50.000	10.000	18.000

Fonte: acervo pessoal

Vale notar que a rede VGG 05 - que é uma versão da rede com melhor desempenho em 10.000 imagens - foi treinada com aproximadamente 18% do dataset total, por falta de capacidade de processamento do computador utilizado. O ideal era treina-la com 70% das 277 mil imagens disponíveis (os outros 30% seriam divididos entre validação e teste).

As redes convolucionais foram implementadas a partir da biblioteca de alto nível Keras, utilizando-se em baixo nível a biblioteca TensorFlow. O código da implementação é mostrado abaixo.

Figura 5: Criação da arquitetura de uma rede convolucional.

```
# Define o tamanho das imagens de input
img_widht = 50
img_height = 50

# Cria o modelo
base_model = applications.VGG16(weights='imagenet',
                                include_top=False,
                                input_shape = (img_widht, img_height, 3)
                               )

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(512, activation='relu')(x)
# x = Dropout(0.25)(x)
# x = Dense(512, activation='relu')(x)

# Add a binary classification layer (sigmoid)
predictions = Dense(1, activation='sigmoid')(x)
model = Model(inputs=base_model.input, outputs=predictions)
```

Fonte: Acervo pessoal.

Para o treinamento, foram importadas duas funções para o call-back do treinamento da rede. A primeira, ModelCheckpoint, para visualização das métricas de avaliação do treino, enquanto este acontece, e uma segunda chamada EarlyStopping que interrompe o treinamento caso a métrica de avaliação do modelo não melhore em épocas seguidas.

Figura 6: funções utilizadas no treinamento da rede

```
from keras.callbacks import ModelCheckpoint
from time import time

experimento = 'VGG16-randomico-01'

# Define os Callbacks de treinamento
callback = ModelCheckpoint( filepath='saved_models/weights.'+ experimento +'.hdf5',
                            verbose=1,
                            save_best_only=True
                          )

early = EarlyStopping(monitor='val_acc',
                      min_delta=0,
                      patience=7,
                      verbose=1,
                      mode='auto'
                     )

start = time()

# Treina o modelo
training = model.fit( train_tensors,
                      y_train,
                      epochs=30,
                      validation_data=(valid_tensors, y_val),
                      batch_size=10,
                      callbacks=[callback, early]
                    )

end = time()

print (' ')
print ("O modelo foi treinado em {:.1f} segundos".format(end - start))
```

Fonte: Acervo Pessoal.

Na camada de ensemble, foi utilizado uma rede neural simples, perceptron multicamada, com os seguintes parâmetros:

Figura 7: parâmetros da rede neural perceptron multicamada

```
MLPClassifier(activation='relu', alpha=10, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='adaptive',
       learning_rate_init=0.001, max_iter=200, momentum=1,
       nesterovs_momentum=True, power_t=0.5, random_state=2017,
       shuffle=True, solver='sgd', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
```

Fonte: Acervo Pessoal.

3.3. Refinamento

Depois das redes VGG 01, VGG 02, VGG 03, VGG 04 serem treinadas em 10.000 imagens, variando-se as camadas congeladas, foi treinada a rede VGG 05 em 50 mil imagens, validada em 10 mil imagens e testada em 18 mil imagens. Essa rede teve os parâmetros copiados da melhor rede encontrada nos treinamentos prévios. Era pressuposto que essa rede apresentasse melhores resultados que as anteriores, pois ao ser treinada com um maior número de imagens, em tese é possível se extrair mais características de classificação.

4. Resultados

4.1. Modelo de avaliação e validação

Os resultados das redes treinadas estão dispostos na tabela a seguir.

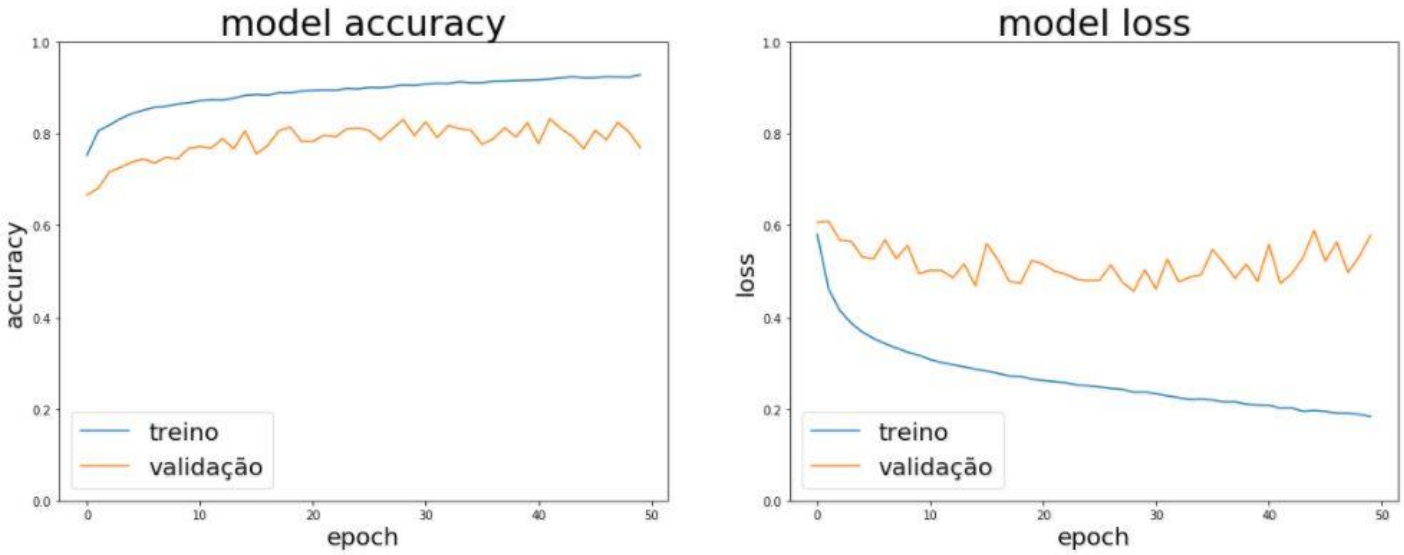
Tabela 3: Resultados do treinamento das redes convolucionais

Rede	F1 score Validação	F1 score Teste
VGG 01	76,2 %	81,1 %
VGG 02	70,5 %	79,5 %
VGG 03	72,0 %	79,9 %
VGG 04	74,6 %	80,1 %
VGG 05	84,9 %	70,7 %

Fonte: Acervo pessoal.

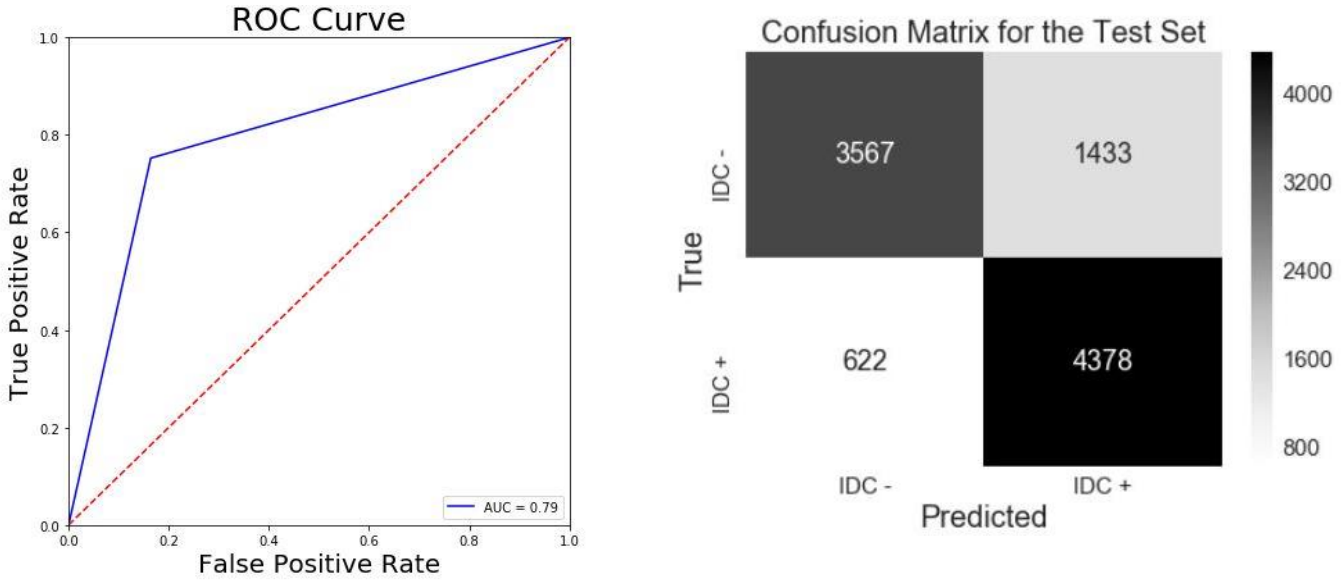
A partir dos resultados, foi contatado que a melhor rede treinada em 10 mil imagens foi a VGG 01, na qual 3 blocos foram congelados. Essa rede apresentou um F1 Score no conjunto de teste de 80,1%. As outras redes treinadas com o mesmo número de imagens tiveram um desempenho quase semelhante, sendo a pior delas, a VGG 02, com um score de 79,5%, uma diferença de 1,6%. Os resultados do treino da rede VGG 01 estão dispostos a seguir.

Figuras 7 e 8: Acurácia e loss da rede VGG 01 durante o treinamento.



Fonte: Acervo pessoal

Figuras 9 e 10: Curva ROC e matriz de confusão da rede VGG 01

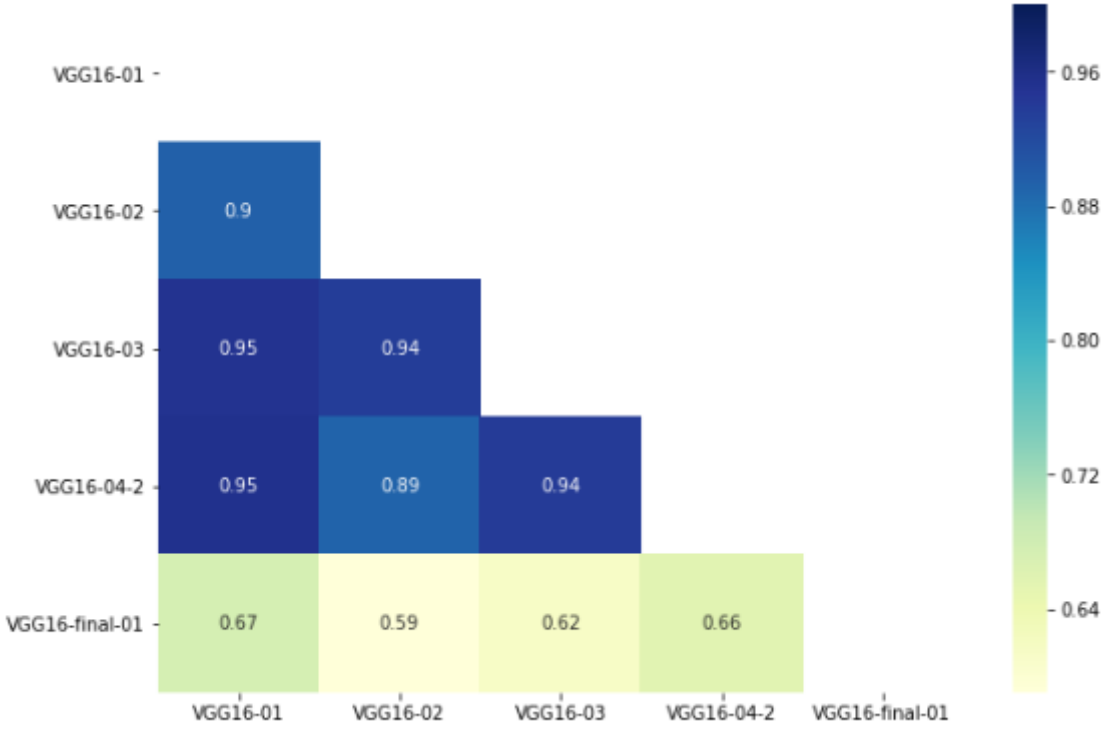


Fonte: Acervo pessoal

Diferentemente do esperado, a rede VGG 05, treinada com 50 mil imagens, teve um desempenho pior se comparado a sua versão treinada em 10 mil imagens. Esse fato pode ser devido à rede VGG 05 ter encontrado mínimos locais e não ter conseguido sair dos mesmos. Foi observado também que para todas as redes, a partir da época de numero 15, não se obtinha melhoras na validação, esta que tendia a aumentar, indicando *overfitting*, uma vez que a acurácia do treino só aumentava.

Após treinadas as redes, foi realizado o ensemble. Abaixo é apresentada a matriz de correlação entre os modelos.

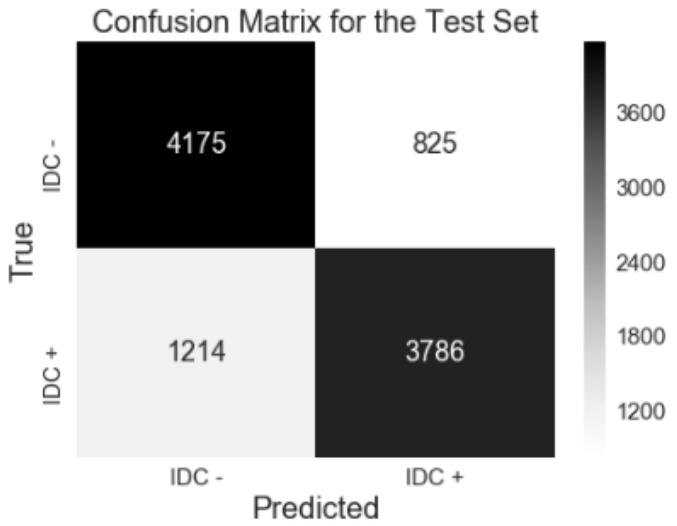
Figura 11: Matriz de correlação entre os modelos



Fonte: Acervo pessoal.

É possível perceber que a rede VGG 05 se correlacionou pouco com as outras redes treinadas, que entre si apresentaram grande correlação. Em uma primeira análise, para um ensemble isso pode ser um bom resultado, pois a rede pode ter encontrado outros mínimos locais se comparada as outras redes e com isso desenviesar as previsões feitas pelos outros modelos. O ensemble dos modelos apresentou a seguinte matriz de confusão.

Figura 12: Matriz de confusão do ensemble.



Fonte: Acervo pessoal.

Comparando com os modelos de benchmark, temos que, para as referências [7] e [8], os resultados de F1 score foram de 0,82 e 0,8213 respectivamente. O modelo deste trabalho apresentou um F1 score de 79%, muito similar se comparado aos outros dois, que possuem a arquitetura muito parecida com a arquitetura utilizada neste trabalho. Isso pode indicar que redes com essa arquitetura possuem uma limitação em melhorar suas precisões, sendo esse valor de F1 score uma barreira invisível inerente à essa arquitetura.

5. Conclusão

Este trabalho buscou utilizar redes neurais convolucionais com pesos pré-treinados em outros datasets para detectar a presença de câncer em imagens de histologias mamárias. É possível concluir que o modelo apresentou bons resultados, prevendo corretamente 79% das imagens. A rede apresentou um enviesamento para imagens sem a presença de câncer, tendo previsto corretamente 83,5% dessas imagens contra 75,7% das imagens que apresentavam a presença de câncer. Na comparação com outros modelos, pode-se concluir que o método de transfer learning foi satisfatório, dado que apresentou resultados muito próximos em relação à redes treinadas do zero [7] [8]. Isso mostra uma grande vantagem para analises de imagens realizadas por redes neurais convolucionais, uma vez que a técnica de transfer learning paresenta tempo de treinamento mais curto se comparado ao treinamento de redes virgens, pois somente uma porcentagem dos pesos da rede tem seus valores otimizados. Devido a limitação da capacidade de processamento, não foi possível treinar a rede em um número maior de imagens (idealmente utilizando

todo o dataset). Pressupõe-se que ao se treinar a rede com mais imagens, esta poderá ver mais exemplos, aferir melhor os rótulos e ser mais assertiva. Provavelmente existe uma limitação das redes convoucionais desta arquitetura de superar a barreira de aproximadamente 80% de acurácia.

Outras arquiteturas não puderam ser utilizadas por não aceitarem imagens de tamanho 50x50. A biblioteca keras possue diversas arquiteturas de redes convolucionais e uma possível melhoria poderia acontecer ao adaptar e treinar essas outras arquiteturas neste dataset e realizar um ensemble dos resultados gerados por elas.

Apesar de uma previsão razoável, ainda não é possível aplicar essas redes treinadas em aplicações reais, pois como comentado anteriormente, falsos positivos e falsos negativos podem acarretar complicações psicológicas nos pacientes submetidos aos exames. Redes neurais profundas tem se mostrado impressionantemente precisas nas mais diversas aplicações e provavelmente ainda serão muito utilizadas na área médica para ajudar médicos a realizarem diagnósticos mais precisos, direcionando melhores tratamentos a pessoas que precisam.

6. Referências

[1] Esteva, A. et al. Dermatologist-level classification of skin cancer with deep neural networks, 2017. Nature, v. 542, p. 115.

[2] Porto, M. A. T. et al. Aspectos Históricos do Controle do Câncer de Mama no Brasil; Revista Brasileira de Cancerologia, 2013; v. 59(3), p. 331-339

[3] http://www2.inca.gov.br/wps/wcm/connect/tiposdecancer/site/home/mama/deteccao_precoce+, acessado em abril de 2018.

[4]https://www.kaggle.com/paultimothymooney/breast-histopathology-images/data, acessado em abril de 2018.

[5] http://blog.kaggle.com/2017/10/17/planet-understanding-the-amazon-from-space-1st-place-winners-interview/, acessado em abril de 2018.

[6] Silva, T. C. et al. Detecção automática de tumores em mamografias utilizando técnicas de processamento digital de imagem; XXIV Congresso Brasileiro de Engenharia Biomédica, 2014.

[7] https://www.kaggle.com/paultimothymooney/predict-idc-in-breast-cancer-histology-images/code, acessado em abril de 2018.

[8] https://www.kaggle.com/raoulma/cancer-image-tensorflow-cnn-80-valid-acc/code, acessado em abril de 2018.

[9] https://github.com/fabioperez/udacity-live-presentations, acessado em abril de 2018.