Nanodegree Engenheiro de Machine Learning

Projeto final de concussão de curso

Italo Pereira Guimarães
5 de janeiro de 2019

I. Definição

I.I Visão geral do projeto

O câncer de pele é a forma mais comum da doença, nos EUA 5,4 milhões de novos casos são diagnosticados por ano, podem ser carcinomas ou melanomas, que é a forma mais mortal, estima-se que 20% dos americanos vão ter câncer de pele, felizmente na maioria dos casos é benigno, o pré-câncer afeta 58 milhões de americanos e muitas pessoas no mundo, nos EUA são registrados 76 mil casos de melanoma por ano, e aproximadamente 10 mil mortes, os EUA gastam em torno de US$ 8,1 bilhões por ano com tratamento e diagnóstico, a taxa de sobrevida em 5 anos para o melanoma de estágio 4 é de aproximadamente 15-20%, enquanto que no estágio 0 é de 99-100%, portanto detectar a doença nos estágios iniciais é primordial. Com base nessas informações, me senti motivado a desenvolver como projeto final, um algoritmo de machine learning capaz de diagnosticar o visualmente o melanoma, a forma mais mortal do câncer de pele. Especificamente, o algoritmo distinguirá este tumor maligno de dois tipos de lesões benignas (nevos e queratoses seborréicas), entre todas as opções de algoritmos disponíveis, eu optei por usar uma rede neural convolucional (CNN) pois ela tem um ótimo desempenho em tarefas de visão computacional.

Esse projeto foi inspirado no 2017 ISIC Challenge on Skin Lesion Analysis Towards Melanoma Detection, que inclusive faz parte desse programa nanodegree, para se ter uma ideia de como essa tarefa é difícil, a imagem abaixo mostra algumas lesões benignas e também malignas, tente distinguir a diferença:

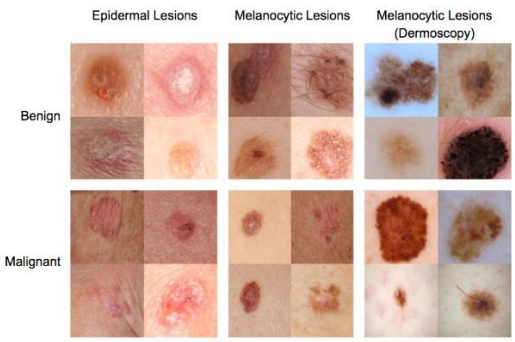

Figura 1: lesões malignas e benignas

Os dados são imagens de lesões de pele, obtidas no site oficial do desafio, com base na imagem da lesão, a rede consegue diagnosticar corretamente a doença, o objetivo é que a rede possua um desempenho que seja comparável, ou preferencialmente superior ao de um dermatologista humano.

I.II Descrição do problema

O câncer de pele é um grande problema de saúde pública, com mais de 5 milhões de casos recém diagnosticados nos Estados Unidos a cada ano. O melanoma é a forma mais letal de câncer de pele, responsável por mais de 9.000 mortes a cada ano.

Dermatoscopia

Como lesões pigmentadas que ocorrem na superfície da pele, o melanoma é passível de detecção precoce por inspeção visual especializada. Também é passível de detecção automatizada com análise de imagem. Dada a ampla disponibilidade de câmeras de alta resolução, os algoritmos que podem melhorar nossa capacidade de rastrear e detectar lesões problemáticas podem ser de grande valor. Como resultado, muitos centros iniciaram seus próprios esforços de pesquisa em análises automatizadas. No entanto, um esforço centralizado, coordenado e comparativo entre as instituições ainda precisa ser implementado. Dermatoscopia é uma técnica de imagem que elimina o reflexo da superfície da pele. Ao remover a reflexão da superfície, a visualização de níveis mais profundos da pele é aprimorada. Pesquisas anteriores mostraram que, quando usada por especialistas em dermatologia, a dermatoscopia oferece maior precisão diagnóstica, em comparação com a fotografia padrão. À medida que os acessórios de dermatoscópio baratos para smartphones estão começando a chegar ao mercado, aumenta a oportunidade de algoritmos automatizados de avaliação dermatoscópica influenciarem positivamente o atendimento ao paciente. Com base nessas informações, me senti motivado a desenvolver como projeto final de conclusão de curso, um algoritmo de machine learning para detectar o câncer de pele.

A estratégia para solucionar o problema consiste basicamente em:

1) Baixar, dividir e pré-processar os dados.
2) Definir a arquitetura da rede neural convolucional.
3) Ajustar os hiper-parâmetros.
4) Treinar o classificador usando transferência de aprendizagem.
5) Utilizar a técnica do ajuste fino, incluindo o aumento dos dados para melhorar ainda mais os resultados.
6) Na fase de testes o objetivo é utilizar métricas para avaliar o desempenho da rede.

A expectativa é que após seguir essa estratégia, seja possível que a CNN consiga diagnosticar o câncer de pele, com um desempenho que supere um dermatologista humano.

I.III Métricas de avaliação

Acurácia

A acurácia é uma métrica muito comum usada em tarefas de classificação, ela é definida como:

$$Acurácia = \frac{\text{Verdadeiros Positivos (TP)} + \text{Falsos Positivos (FP)}}{\text{Quantidade total de dados}}$$

A acurácia mede a frequência com que o modelo geralmente está correto, ela calcula a razão entre a quantidade de instâncias de dados que o modelo classificou corretamente, pela quantidade total de dados que foram usados para testar o modelo.

A acurácia nem sempre é uma métrica adequada, em casos de classes bastantes desbalanceadas, e estamos interessados em detectar o caso mais raro, ela é bastante limitada, nesses casos as métricas precisão e recall são mais adequadas.

Precisão (precision)

A precisão pode ser definida como:

$$Precisão = \frac{\text{Verdadeiros Positivos (TP)}}{\text{Verdadeiros Positivos (TP)} + \text{Falsos Positivos (FP)}}$$

A precisão mede a razão entre a quantidade de instâncias que realmente eram corretas e aquelas que o modelo classificou corretamente, nesse caso especificamente ela responde a seguinte pergunta: de todas imagens que classifiquei como sendo câncer quantas realmente eram câncer?

Recall (revocação)

Nesse caso em particular o Recall é uma métrica mais adequada, porque queremos detectar o máximo possível de pessoas com câncer, é definida como:

$$Recall = \frac{\text{Verdadeiros Positivos (TP)}}{\text{Verdadeiros Positivos (TP)} + \text{Falsos Negativos (FN)}}$$

O recall mede a frequência com que o classificador encontra exemplos de uma classe, ou seja é a razão entre quantas instâncias o modelo classificou como sendo de uma classe, e quantas instâncias realmente eram daquela classe, nesse caso o recall responde a seguinte pergunta: entre todas as pessoas que tem melanoma, quantas o nosso modelo conseguiu identificar corretamente como tendo melanoma?

FBeta Score

Essa métrica é na verdade uma combinação da precisão e do recall, é definida como:

$$F\beta eta\ Score = (1 + \beta^2)\frac{\text{precisão} * \text{recall}}{(\beta^2 * \text{precisão}) + \text{recall}}$$

F Beta Score avalia a qualidade geral do modelo, trabalha bem até com conjuntos de dados desbalanceados, quanto maior o beta mais próximo o valor se torna do recall, nesse caso em particular usei beta igual a 4 pois queria que o racall tivesse um peso maior.

Matriz de Confusão

A matriz de confusão permite avaliar o desempenho do modelo graficamente, mostrando os valores reais e previstos pelo classificador, ela é definida como:

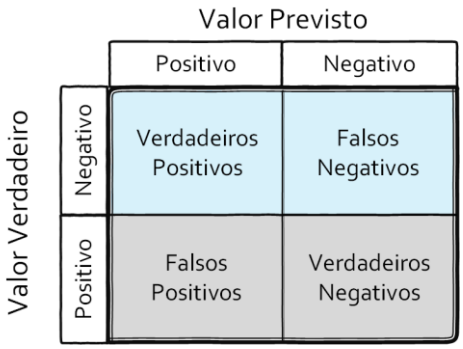

Figura 2: matriz de confusão

Quanto mais próximos de 1 estiverem os elementos da diagonal principal, melhor será o desempenho do nosso modelo.

Curva ROC

A análise ROC tem sido utilizada em medicina, radiologia, psicologia e outras áreas por muitas décadas e, mais recentemente, foi introduzida à áreas como aprendizado de máquina e mineração de dados.

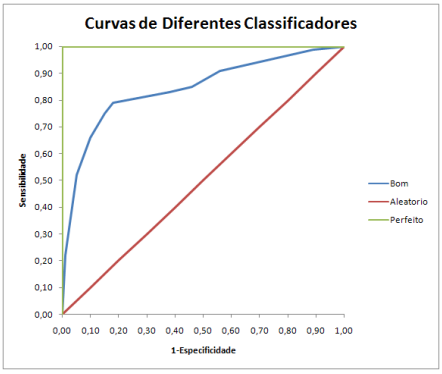

Figura 3: curva roc

Como o resultado de sistemas de classificação em classes geralmente são contínuos, é necessário definir um ponto de corte, ou um limiar de decisão, para se classificar e contabilizar o número de predições positivas e negativas (como diagnósticos verdadeiros e falsos no caso de ocorrência de uma patologia). Como este limiar pode ser selecionado arbitrariamente, a melhor prática para se comparar o desempenho de diversos sistemas é estudar o efeito de seleção de diversos limiares sobre a saída dos dados.

Para cada ponto de corte são calculados valores de sensibilidade e especificidade, que podem então serem dispostos em um gráfico denominado curva ROC, que apresenta no eixo das ordenadas os valores de sensibilidade e nas abscissas o complemento da especificidade,

ou seja, o valor (1-especificidade). Quanto maior a área sob a curva do gráfico, melhor é o classificador, um classificador aleatório teria área igual a 0.5, e um ideal teria área igual a 1.

Inspirado pelo desafio ISIC, o algoritmo será classificado segundo três categorias separadas:

Categoria 1: ROC AUC para Classificação de Melanoma Na primeira categoria, avaliaremos a habilidade da sua CNN distinguir entre melanoma maligno e lesões de pele benignas (nevos, queratose seborréica) calculando a área sob a curva característica de operação do receptor (ROC AUC) correspondente a esta tarefa de classificação binária.

Categoria 2: ROC AUC para Classificação Melanocítica Todas as lesões de pele que examinaremos são causadas pelo crescimento anormal de [melanócitos] ou [queratinócitos], que são dois tipos diferentes de células epidérmicas da pele. Melanomas e nevos são derivados de melanócitos, enquanto queratoses seborréicas são derivadas de queratinócitos. Na segunda categoria, avaliaremos a habilidade da sua CNN distinguir entre lesões de pele melanocíticas e queratinocíticas, calculando a área sob a curva característica de operação do receptor (ROC AUC) correspondente a esta tarefa de classificação binária.

Categoria 3: ROC AUC média: Na terceira categoria, tiraremos a média dos valores ROC AUC das duas primeiras categorias.

Quando o script get_results.py é executado, ele exibe a curva roc e a matriz de confusão, que foram incluídas em IA_dermatologist.ipynb.

1 Blog César Souza. Análise de Poder Discriminativo Através de Curvas ROC. Disponível em: <http://crsouza.com/2009/07/13/analise-de-poder-discriminativo-atraves-de-curvas-roc/> Acesso em: 2 de Janeiro de 2019

2 Medium. Métricas Comuns em Machine Learning, Disponível em: <https://medium.com/as-m%C3%A1quinas-que-pensam/m%C3%A9tricas-comuns-em-machine-learning-como-analisar-a-qualidade-de-chat-bots-inteligentes-m%C3%A9tricas-1ba580d7cc96 > Acesso em: 3 de Janeiro de 2019

3 Mineirando Dados. Café com Código #20:Matriz de confusão, você sabe como utilizar**?**

. Disponível em: <http://minerandodados.com.br/index.php/2018/01/16/matriz-de-confusao/ > Acesso em: 2 de Janeiro de 2019

II. Análise

II.I Exploração dos dados

Os dados estão disponíveis publicamente na seção Get Started, do repositório Dermatologist AI: https://github.com/udacity/dermatologist-ai, e também foram estendidos, a partir do 2017 ISIC Challenge on Skin Lesion Analysis Towards Melanoma Detection.

O International Skin Imaging Collaboration (ISIC) é um esforço internacional para melhorar o diagnóstico do melanoma, patrocinado pela Sociedade Internacional de Imagem Digital da Pele (ISDIS). O Arquivo ISIC contém a maior coleção publicamente disponível de imagens dermatoscópicas controladas de qualidade de lesões de pele. Os inputs são imagens de lesões:

melanoma, nevos e queratose sebórréica obtidas do ISIC Archive, o ISIC Archive contém mais de 13.000 imagens dermatoscópicas, que foram coletadas dos principais centros clínicos internacionalmente e adquiridas a partir de uma variedade de dispositivos dentro de cada centro.

A participação ampla e internacional na contribuição da imagem é projetada para garantir uma amostra representativa clinicamente relevante. Os conjuntos de dados iniciais derivam principalmente dos Estados Unidos, o ISIC tem compromissos e contribuições contínuos de conjuntos de dados adicionais de colaboradores internacionais. Todas as imagens recebidas no ISIC Archive são rastreadas para garantia de privacidade e qualidade. A maioria das imagens tem metadados clínicos associados, que foram avaliados por especialistas reconhecidos em melanoma. Um subconjunto das imagens foi submetido a anotações e marcações por especialistas reconhecidos em câncer de pele. Essas marcações incluem características dermatoscópicas (isto é, elementos morfológicos globais e focais na imagem conhecidos por discriminar entre os tipos de lesões da pele).

Esse dataset é uma fonte rica para esse projeto, pois contém imagens suficientes com todas as características necessárias para que ela possa reconhecer os 3 tipos de doenças, para esse projeto em particular, as imagens de treinamento, teste e validação, foram adquiridos combinando as duas fontes citadas acima, usei dados de teste e validação diferentes dos dados oficiais do desafio, comecei reunindo as imagens para cada categoria de doença, como o objetivo é que a rede fosse boa em detectar o melanoma, eu usei o máximo de imagens de melanoma que consegui, eu quis que o dataset fosse desbalanceado propositalmente, tendo muito mais imagens de melanoma do que das outras doenças, pois queria que a rede tivesse uma maior tendência a classificar qualquer imagem como sendo melanoma, ainda sim o dataset ficou menos desbalanceado que o do desafio original, também estendi o máximo que pude as imagens de seborrheic keratosis, pois a parte 2 do desafio avalia a capacidade da rede de detectar essa doença, mas como o dataset era muito pobre consegui poucas imagens, e tive que usar todas que consegui, incluindo as imagens ruins, eu reduzi significativamente as imagens da classe nevus, pois das minhas experiencias anteriores, a rede tinha uma maior tendência a classificar qualquer doença como pertencendo a essa classe, felizmente as imagens da classe nevus eram de boa qualidade, então pude selecionar as melhores imagens.

O dataset tinha 1500 imagens de melanoma, 500 de nevus e 419 de seborrheic keratosis, usei 70% dos dados para treinamento, 20% para teste e 10% para validação, após a divisão os dados ficaram distribuídos da seguinte forma:

MELANOMA: 1050 imagens para treinamento, 300 para teste e 150 para validação.

NEVUS: 350 imagens para treinamento, 100 para teste e 50 para validação.

SEBOREIC KRATOSIS: 293 imagens para treinamento, 84 para teste e 42 para validação.

As imagens abaixo representam uma amostra do dataset:

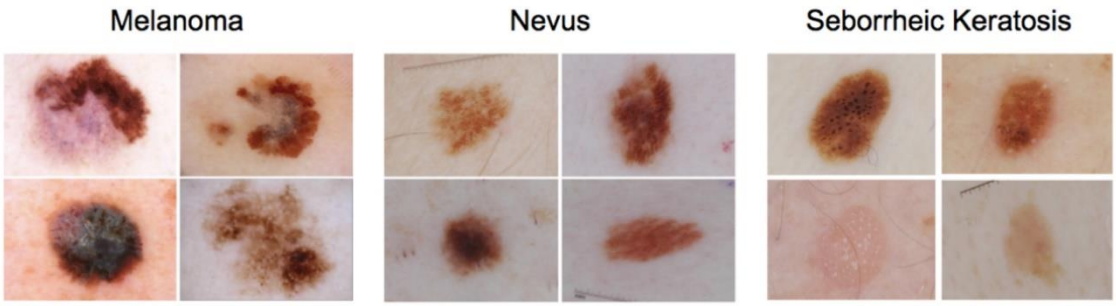

Figura 4: imagens de lesões causadas por melanoma, nevus e seborrheic keratosis

II.II Visualização exploratória

Como se tratam de imagens, a forma utilizada para visualizar os dados consiste basicamente em carregar as imagens, e exibi-las usando uma função para plotar imagens, nesse caso o pyplot do módulo matplotlib, entretanto, as imagens foram plotadas apenas na etapa de depuração, e não foram incluídas no projeto depois de concluído.

II.III Algoritmos e técnicas

Rede Neural

Optei por usar uma rede neural convolucional (CNN) usando keras, pois apresentam um ótimo desempenho em tarefas de classificação de imagens, portanto ela é adequada para essa tarefa.

Os seguintes parâmetros foram ajustados para otimizar o modelo:

Parâmetros de treinamento:

- Taxa de aprendizado
- Decaimento da Taxa de aprendizado
- Tamanho do lote
- Número de épocas
- Otimizador
- Função de perda

Arquitetura da rede neural:

- Número de camadas
- Tipos de camadas
- Funções de ativação
- Probabilidade de dropout

Transferência de Aprendizado

A transferência de aprendizado é uma técnica muito utilizada, inclusive na indústria, pois é muito caro computacionalmente treinar uma CNN do zero, por esse motivo optei por usa-la, ela consiste em usar uma CNN já treinada em outro conjunto de dados, preferencialmente o

mais parecido possível com o que se pretende utilizar, remover a ultima camada densamente conectada, responsável por realizar a classificação, e depois passar todo o conjunto de imagens pela rede, esses dados são conhecidos como recursos de gargalo, posteriormente é necessário implementar a arquitetura da rede para a tarefa especifica que se pretende solucionar, e finalmente usar os dados obtidos anteriormente para treinar a nova rede.

Para esse projeto foi utilizada uma rede treinada no dataset da imagenet, a rede utilizada foi a NASNetLarge que é a rede com o maior desempenho entre todas disponíveis no keras.

Aumento dos Dados

Essa técnica consiste em alterar os dados de entrada, para que a rede possa generalizar melhor, as alterações podem ser deslocamentos verticais e horizontais, inverter a imagem, aumentar o tamanho da imagem, distorcer a imagem, etc, essa técnica foi utilizada pois o objetivo é evitar o sobre ajuste, e que a rede pudesse reconhecer melhor o melanoma.

Ajuste Fino

O Ajuste Fino é normalmente utilizado com o aumento dos dados, é usada para aumentar ainda mais a precisão do modelo, por esse motivo optei por utiliza-la, ela consiste em combinar duas redes, uma é a mesma CNN usada para obter os dados na etapa da transferência de aprendizado, outra rede é a camada final densamente conectada, responsável por fazer a classificação, já treinada anteriormente, essa rede é conectada no final da CNN, finalmente a rede resultante é treinada, mas dessa vez não são usados os recursos de gargalo, e sim as imagens originais.

II.IV Modelos de Benchmark

O modelo de referência é o trabalho publicado na Nature, pela equipe de Sebastian Thrun, no qual treinam uma CNN para diagnosticar câncer de pele, usando a base de dados DSA internacional, e surpreendentemente obteve uma precisão de 71%, um resultado consideravelmente melhor que o obtido pala média dos médicos dermatologistas humanos, que gira em torno de 65%.

1 Nature. Dermatologist-level classification of skin cancer with deep neural networks. Disponível em: <https://www.nature.com/articles/nature21056.epdf?author_access_token=8oxIcYWf5UNrNpHsUHd2StRgN0jAjWel9jnR3ZoTv0NXpMHRAJy8Qn10ys2O4tuPakXos4UhQAFZ750CsBNMMsISFHIKinKDMKj ShCpHIlYPYUHhNzkn6pSnOCt0Ftf6> Acesso em: 4 de Dezembro de 2018

III. Metodologia

III.I Pré-processamento de dados

Iniciei a etapa de pré-processamento realizando uma limpeza profunda nos dados, foi um processo lento e tedioso, mas que deu ótimos resultados, o objetivo é garantir que todos os

dados tenham um mesmo padrão, se por exemplo existiam 3 imagens semelhantes no dataset, eu enviei cada uma delas para o conjunto de treinamento, testes e validação, além de eliminar imagens com marcadores, com bordas escuras, muito poluídas, imagens repetidas, etc.

Utilizei também outra técnica de pré-processamento bastante simples, tipicamente usada em processamento de imagens, como cada pixel da imagem pode ter qualquer valor entre 0 e 255, todos os pixels devem ser divididos por 255 para que cada um deles seja um número entre 0 e 1, para que sejam normalizados.

Os rótulos foram codificados em one-hot, usando a função to_categorical disponível em keras.utils, as imagens foram redimensionadas para o formato (331,331), que é o formato as imagens do imagenet, para que a transferência de aprendizado pudesse ser usada.

III.II Implementação

A primeira parte da implementação é extrair os recursos de gargalo, necessários para a transferência de aprendizado, o script extract_bottleneck_feature.py é responsável por essa tarefa, na qual estão implementadas as seguintes funções:

A função load_dataset é responsável por carregar os dados, ela recebe como argumento o caminho relativo dos dados, e os retorna na forma de arrays numpy, além de converter os rótulos em codificação one-hot.

A função paths_to_tensor utiliza a função auxiliar path_to_tensor, ela é responsável por converter o caminho relativo dos dados em um tensor, ela recebe como argumento os dados retornados pela função load_dataset, redimensiona, e os retorna convertidos em uma lista de tensores.

Após os dados serem carregados com a função load_dataset, eles são convertidos em tensores pela função paths_to_tensor, e para a etapa de pré-processamento, os valores armazenados nos tensores são divididos por 255, então os dados de treinamento, teste e validação, são passados por uma instância da NASNetLarge, disponível no keras, através da função predict e salvos para serem utilizados futuramente.

O projeto principal está implementado em IA_dermatologist.ipynb, inicialmente os dados são carregados e pré-processados exatamente como foi feito anteriormente no script extract_bottleneck_feature.py, a diferença é que dessa vez são exibidas algumas estatísticas referentes ao dataset, e os dados depois de serem pré-processados são salvos para que não seja necessário realizar essa etapa toda vez que for necessário executar o notebook, pois essa etapa de pré-processamento é muito demorada.

Para executar a transferência de aprendizagem, primeiramente os dados salvos na etapa anterior, incluindo os recursos de gargalo são carregados, optei por utilizar a NASNetLarge, pois é o modelo de melhor desempenho segundo o artigo do Gooble IA Blog. **AutoML for large scale image classification and object detection**.

A próxima etapa é implementar a arquitetura da rede neural densamente conectada, responsável pelas classificações, que posteriormente será conectada no final da rede, na etapa de ajuste fino:

```
from keras.models import Sequential
```

```
from keras.layers import GlobalAveragePooling2D
from keras.layers import Dropout,Dense,Activation,BatchNormalization
n_classes=len(disease_names) # número de classes
p_dropout = 0.50 # probabilidade de dropout
n_hidden = 8064 # número de neurônios na camada primeira oculta

NASNet_top = Sequential()
NASNet_top.add(GlobalAveragePooling2D(input_shape=train_NASNet.shape[1:]))
NASNet_top.add(Dense(n_hidden))
NASNet_top.add(Activation('relu'))
NASNet_top.add(BatchNormalization())
NASNet_top.add(Dropout(p_dropout))
NASNet_top.add(Dense(n_classes, activation='sigmoid'))
NASNet_top.summary()
```

Utilizei dropout para evitar o sobre ajuste, e normalização em lote para resolver o problema das distribuições dos recursos que variam nos dados de treinamento e teste, melhorando o desempenho do modelo, e relu como função de ativação para evitar o problema da fuga do gradiente, e como função de perda utilizei a função Dice Coeficiente:

```
from keras import backend as K

def dice_coef(y_true, y_pred, smooth=1):
   y_true_f = K.flatten(y_true)
   y_pred_f = K.flatten(y_pred)
   intersection = K.sum(y_true_f * y_pred_f)
   return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)

def dice_coef_loss(y_true, y_pred):
   return -dice_coef(y_true, y_pred)
```

Após modelo ser compilado, usando Adam como otimizador:

```
decay=1e-6

opt = optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=decay,
amsgrad=False, clipvalue=0.5)
```

utilizei taxa de aprendizado de 0.001, taxa de decaimento de 1e-6, e gradient clipping, para evitar o problema da explosão do gradiente.

a rede é treinada, usando os seguintes parâmetros:

```
batch_size = 32 #tamanho do lote
epochs = 10 #numero de épocas
```

e o modelo com o melhor acurácia no conjunto de testes é salvo.

Para testar o modelo será utilizada a acurácia como principal métrica de avaliação, pois é uma boa forma de testar o desempenho geral do modelo, sua implementação segue abaixo:

```
# obtain the predicted disease class index for each image in the test set
NASNet_predictions = [np.argmax(NASNet_top.predict(np.expand_dims(feature, axis=0))) for
feature in test_NASNet]
```

```
# report test accuracy
test_accuracy = 100*np.sum(np.array(NASNet_predictions)==np.argmax(test_targets,
axis=1))/len(NASNet_predictions)
print('Acurácia no Conjunto de Testes: %.4f%%' % test_accuracy)
```

Após treinar e obter um modelo com bom desempenho, suas previsões são salvas em um arquivo sample_predictions.csv, para serem usadas para calcular a curva roc e a matriz de confusão.

As métricas precisão, recall e fbeta score estão disponíveis em sklearn.metrics respectivamente como precision_score, recall_score e fbeta_score.

O recall é uma métrica muito importante, pois queremos detectar o máximo possível de pessoas com câncer, o fbeta score foi implementado usando beta=4, pois o objetivo é que o recall tenha maior peso, pois é mais importante.

As demais métricas curva roc e matriz de confusão estão implementadas no script get_results.py respectivamente como plot_roc_auc, plot_confusion_matrix, não foi necessário implementa-las pois esse script é fornecido nos arquivos do projeto do desafio original da Udacity.

a matriz de confusão é uma boa forma de visualizar o desempenho do modelo, e a curva roc é muito utilizada em aplicações médicas, sendo que a área da curva roc é uma boa métrica para avaliar o desempenho do modelo, para plotar a curva roc e a matriz de confusão utilizei um threshold igual a 0.4, pois a intenção é que o modelo consiga detectar o máximo possível de lesões malignas.

A próxima etapa é executar o ajuste fino, para isso é necessário criar uma instância do modelo NASNetLarge do Keras, pré-inicializada com os pesos da imagenet, cuja função é ser usado como a CNN base:

```
from keras.applications.nasnet import NASNetLarge

NASNet_base=NASNetLarge(weights='imagenet', include_top=False,
input_shape=train_tensors.shape[1:])
```

Agora é necessário combinar os dois modelos criados anteriormente, NASNet_base e NASNet_top, em um único modelo:

```
NASNet_model = Sequential()
NASNet_model.add(NASNet_base)
NASNet_model.add(NASNet_top)
```

Para treinar o modelo utilizei os seguintes hiper-parâmetros:

```
batch_size = 32 #tamanho do lote
epochs=50 #número de épocas
```

Defini as últimas 94 camadas do modelo como treináveis, que são compostas pelos 2 últimos blocos convolucionais da CNN NASNet, incluindo a última camada densamente conectada, não é recomendado treinar a rede toda, pois como a CNN tem uma capacidade entrópica muito alta, ela tem grande chance de sofrer sobre-ajuste, após compilar o modelo utilizando adam

como otimizador, com uma taxa de aprendizado de 0.0001, para usar o ajuste fino a taxa de aprendizado deve ser muito baixa, taxa de decaimento igual a 1e-7, os demais parâmetros são iguais aos usados anteriormente, para treinar o modelo, utilizei a classe ImageDataGenerator disponível em keras.preprocessing.image, para implementar o aumento dos dados.

Começamos definindo o gerador:

```
datagen = ImageDataGenerator(
    rotation_range=270, #faixa de rotação em graus
    width_shift_range=0.15, #deslocamento horizontal em porcentagem da largura da imagem
    height_shift_range=0.15, #deslocamento vertical em porcentagem da altura da imagem
    shear_range=0.15, #ângulo da tensão de cisalhamento
    zoom_range=0.2, #faixa do zoom em porcentagem do tamanho da imagem
    horizontal_flip=True, # inverter hotizontalmente
    vertical_flip=True, #inverter verticalmente
    fill_mode='nearest') #modo de preenchimento
```

Depois ajustamos o gerador aos dados:

```
datagen.fit(train_tensors)
```

então utilizamos a função fit_generator, passando os dados de treinamento originais, após serem ajustados pelo gerador, o modelo final foi treinado por mais de 50 épocas, usando tamanho do lote igual a 32.

Finalmente o modelo final foi testado da mesma forma que antoriormente.

O processo de implementação não teve maiores problemas, a grande dificuldade que enfrentei inicialmente, foi treinar a rede, pois a rede não aprendia, ela tinha um péssimo desempenho no conjunto de testes, após realizar uma limpeza profunda nos dados e algumas mudanças na implementação que serão discutidas detalhadamente na sessão posterior, a rede teve um desempenho satisfatório.

III.III Refinamento

Foram experimentadas várias combinações diferentes de arquiteturas, otimizadores e hyper-parâmetros, todas por tentativa e erro, tentei otimiza-los usando hyperas, e gridsearchcv mas não foi possível por estar usando a transferência de aprendizagem, como não é possível mostrar todas as diferentes combinações, apenas as principais serão apresentadas:

Inicialmente foram testadas 3 arquiteturas:

1)

```
n_classes=len(disease_names) # número de classes
p_dropout = 0.40 # probabilidade de dropout
n_hidden = 5000 # número de neurônios na camada oculta

NASNet_top = Sequential()
NASNet_top.add(GlobalAveragePooling2D(input_shape=train_NASNet.shape[1:]))
NASNet_top.add(Dense(n_hidden))
NASNet_top.add(Activation('elu'))
NASNet_top.add(BatchNormalization())
```

```
NASNet_top.add(Dropout(p_dropout))
NASNet_top.add(Dense(n_classes, activation='softmax'))
```

2)

```
n_classes=len(disease_names) # número de classes
p_dropout = 0.40 # probabilidade de dropout
n_hidden_1 = 5000 # número de neurônios na camada primeira oculta
n_hidden_2 = 2500 # número de neurônios na camada segunda oculta

NASNet_top = Sequential()
NASNet_top.add(GlobalAveragePooling2D(input_shape=train_NASNet.shape[1:]))
NASNet_top.add(Dense(n_hidden_1))
NASNet_top.add(Activation('elu'))
NASNet_top.add(BatchNormalization())
NASNet_top.add(Dropout(p_dropout))
NASNet_top.add(Dense(n_hidden_2))
NASNet_top.add(Activation('elu'))
NASNet_top.add(BatchNormalization())
NASNet_top.add(Dropout(p_dropout))
NASNet_top.add(Dense(n_classes, activation='softmax'))
```

3)

```
n_classes=len(disease_names) # número de classes
p_dropout = 0.40 # probabilidade de dropout
n_hidden_1 = 8064 # número de neurônios na camada primeira oculta

NASNet_top = Sequential()
NASNet_top.add(GlobalAveragePooling2D(input_shape=train_NASNet.shape[1:]))
NASNet_top.add(Dense(n_hidden_1))
NASNet_top.add(Activation('elu'))
NASNet_top.add(BatchNormalization())
NASNet_top.add(Dropout(p_dropout))
NASNet_top.add(Dense(n_classes, activation='softmax'))
```

Durante o processo de desenvolvimento, foram testados diferentes combinações entre otimizadores: RMSprop, Adam, Adagrad, Adadelta e sgd, número de épocas: 10, 20, 30, 50 e 100 e tamanho do lote: 32, 64, 128 e 256, sem obter um resultado satisfatório, após fazer uma limpeza profunda dos dados, alterar a arquitetura, e a função de perda para dice coeficiente, houve uma melhora significativa no desempenho do modelo. arquitetura que apresentou o melhor desempenho foi baseada no artigo:

Menegola A, Tavares J, Fornaciali M, Li LT, Avila S, Valle E. “RECOD Titans at ISIC Challenge 2017”. International Skin Imaging Collaboration (ISIC) 2017 Challenge at the International Symposium on Biomedical Imaging (ISBI) (**github**)

Esse artigo, serviu como principal referência para todo o projeto, pois ao fazer as mudanças nele sugeridas que consegui melhores resultados no conjunto de testes, a implementação da arquitetura final segue abaixo:

4)

```
n_classes=len(disease_names) # número de classes
p_dropout = 0.50 # probabilidade de dropout
```

```
n_hidden = 8064 # número de neurônios na camada primeira oculta

NASNet_top = Sequential()
NASNet_top.add(GlobalAveragePooling2D(input_shape=train_NASNet.shape[1:]))
NASNet_top.add(Dense(n_hidden))
NASNet_top.add(Activation('relu'))
NASNet_top.add(BatchNormalization())
NASNet_top.add(Dropout(p_dropout))
NASNet_top.add(Dense(n_classes, activation='sigmoid'))
```

Ela é basicamente a arquitetura 3, apenas as funções de ativação foram alteradas.

IV. Resultados

IV.I Modelo de avaliação e validação

Discutindo de forma mais profunda os resultados obtidos na sessão anterior, as arquiteturas 1,2 e 3 obtiveram uma acurácia no conjunto de testes respectivamente de 61.2173%, 57.1405% e 65.3736%, todos os testes foram feitos utilizando o tamanho do lote igual a 32, número de ép
ocas igual a 10, e adam como otimizador.

Para otimizar os demais hyper-parâmetros, Todos os demais testes foram realizados usando a arquitetura 3, taxa de decaimento de 1e-6 e categorical_crossentropy como função de perda.

A tabela abaixo mostra a acurácia obtida através de diferentes combinações de otimizadores, com e sem decaimento, usando número de épocas igual a 10 e tamanho do lote igual a 32:

	Sem decaimento	**Com decaimento**
RMSprop	52.500%	63.667%
Adam	33.333%	65.333%
Adagrad	55.833%	33.020%
Adadelta	42.833%	31.500%
sgd	46.333%	45.500%

A tabela abaixo mostra a acurácia obtida através de diferentes combinações de números de épocas, usando tamanho do lote igual a 32:

Número de épocas	**Acurácia**
10	65.500%
20	65.1600%
50	64.000%
100	58.600%

A tabela abaixo mostra a acurácia obtida através de diferentes combinações de tamanho do lote, usando número de épocas igual a 10:

Tamanho do lote	**Acurácia**
32	65.500%

64	64.1600%
128	62.000%
256	59.600%

Após realizar as mudanças discutidas na sessão anterior, sugeridas pelo artigo, o modelo conseguiu obter uma acurácia no conjunto de testes igual a 77.0661%, precisão de 0.7311%, recall de 0.7707% e fbeta score igual a 0.7637%, utilizando adam como otimizador, tamanho do lote igual a 32, e número de épocas igual a 10, uma melhora significativa em relação aos resultados anteriores.

A próxima etapa é executar o ajuste fino e aumento dos dados, o modelo foi compilado utilizando adam como otimizador, com uma taxa de aprendizado de 0.0001, para usar o ajuste fino a taxa de aprendizado deve ser muito baixa, taxa de decaimento igual a 1e-7, os demais parâmetros são iguais aos usados anteriormente, após treinar o modelo por mais de 50 épocas e usando tamanho do lote igual a 32, o modelo conseguiu obter uma acurácia no conjunto de testes de apenas 73.5537%, confesso que fiquei bastante decepcionado, pois levou meses para treinar o modelo final e a expectativa é que a acurácia fosse aumentar, como leva muito tempo para treinar o modelo utilizando o ajuste fino e o aumento dos dados em uma CPU, aproximadamente 7,5 horas para treinar uma única época, não pude treinar o modelo por 50 épocas de uma única vez, e como o modelo final teve um desempenho inferior ao modelo anterior, não achei necessário calcular as demais métricas de avaliação.

IV.II Justificativa

O modelo conseguiu obter uma acurácia no conjunto de testes de 77.0661% enquanto que o modelo de referência conseguiu obter uma acurácia de 71%, apesar da última etapa não melh orar o desempenho do modelo final, o modelo conseguiu um resultado significativamente sup erior ao modelo de referência, claro que existem outros fatores que devem ser levados em con sideração, mas apesar disso, o modelo final atendeu as expectativas, pois soluciona o problem a proposto pelo projeto.

V. Conclusão

V.I Forma livre de visualização

A imagem abaixo representa a distribuição dos dados de treinamento, teste e validação, usados para implementar o modelo:

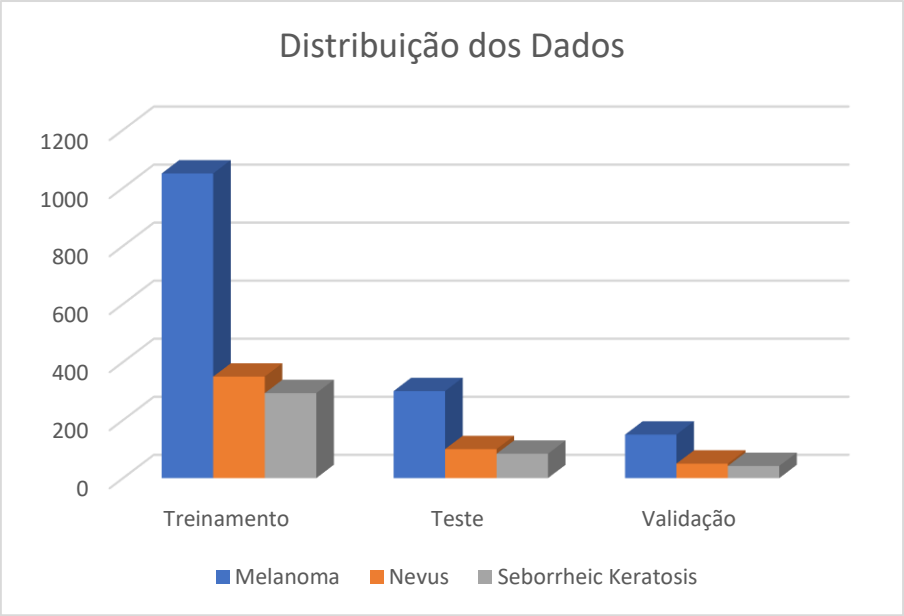

Figura 5: distribuição dos dados pelo modelo

Como foi discutido anteriormente apesar dos dados estarem bastante desbalanceados, ainda estão menos desbalanceados que os dados fornecidos pelo desafio original.

A área da curva ROC é uma métrica muito importante para avaliar o desempenho do modelo, abaixo temos a curva ROC obtida durante o projeto:

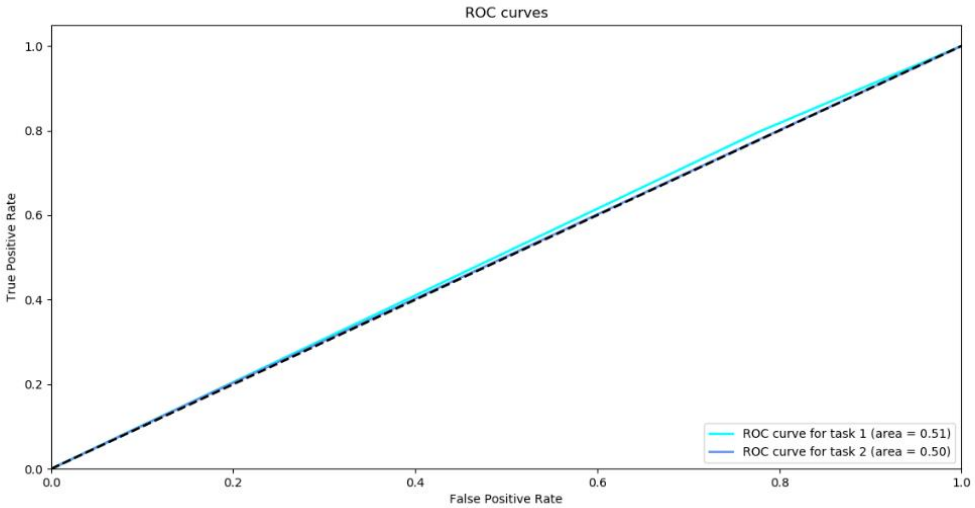

Figura 6: curva roc obtida pelo modelo

As pontuações obtidas nas categorias 1,2 e 3 do desafio, foram respectivamente iguais a 0.510, 0.502 e 0.506.

A figura abaixo mostra a matriz de confusão obtida pelo modelo:

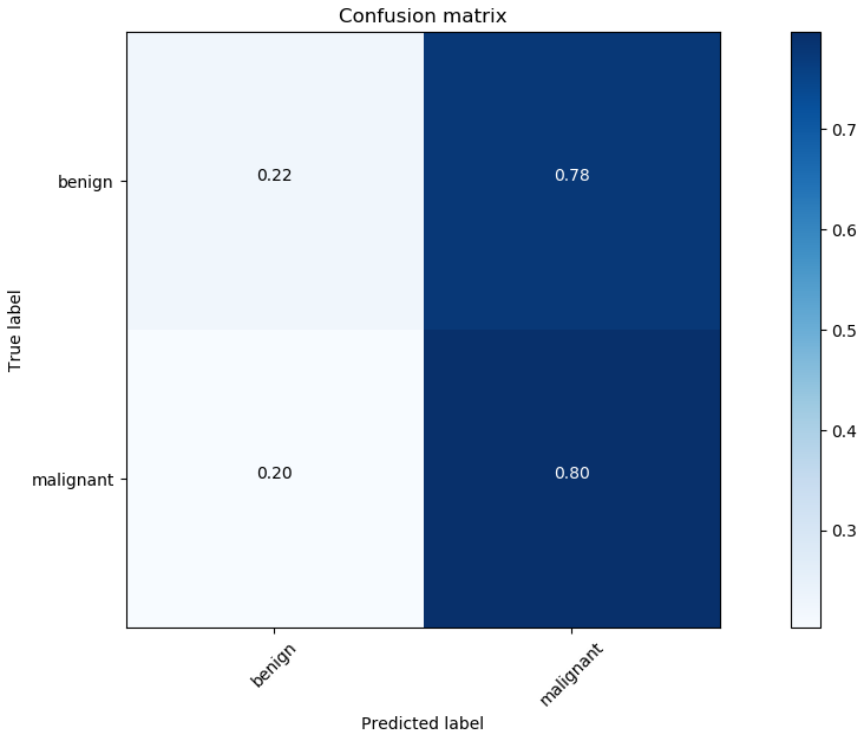

Figura 7: matriz de confusão obtida pelo modelo

Esses resultados não são muito satisfatórios, pois como eu usei mais imagens de melanoma, pois queria que a rede tivesse maior facilidade de detectar câncer e tivesse uma tendência maior de classificar outras doenças como melanoma, e como já era esperado, a rede tem uma tendência maior de classificar outras doenças como melanoma, como mostra a matriz de confusão, a rede conseguiu detectar 80% das doenças malignas, mas também classificou muitas doenças benignas como melanoma, ela também teve bastante dificuldade de distinguir a doença seborrheic keratosis, por esse motivo ela não teve um desempenho muito bom nas 3 categorias do desafio.

V.II Reflexão

Todo o processo pode ser resumido nas seguintes etapas:

1. Estender o conjunto de dados usando outras fontes
2. Limpar e pré-processar os dados
3. Extrair os recursos de gargalo
4. Carregar e pré-processar todos os dados
5. Testar várias arquiteturas de rede, otimizadores e hyper-parâmetros diferentes, para obter o modelo otimizado.
6. Executar a transferência de aprendizado
7. Treinar o modelo
8. Utilizar as métricas de avaliação para e testar o modelo
9. Executar o ajuste fino e aumento dos dados
10. Treinar o novo modelo
11. Utilizar novamente as métricas de avaliação, para verificar se houve melhorias no modelo final

Como já foi dito anteriormente, o grande desafio enfrentado inicialmente foi treinar a rede, pois a rede não aprendia, tinha um péssimo desempenho no conjunto de testes, notei que ao fazer uma limpeza profunda nos dados a rede apresentou uma melhora significativa, mesmo tendo reduzido o dataset, em relação ao que eu usei no início, com isso aprendi uma importante lição na ciência de dados: a qualidade dos dados é muito mais importante do que a quantidade, a após implementar as mudanças sugeridas pelo artigo o desempenho do modelo melhorou ainda mais.

Outra coisa que me chamou muito a atenção é que mesmo após executar o ajuste fino e o aumento dos dados, e treinar por mais de 50 épocas, o desempenho da rede piorou, fiquei bastante surpreso com esse resultado, pois a expectativa é que ele melhorasse, confesso que não sei explicar o motivo disso ter acontecido, talvez foi uma falha na implementação ou a rede sofreu sobre-ajuste.

Apesar do modelo não ter obtido um bom desempenho nas métricas curva ROC e matriz de confusão, o modelo conseguiu um desempenho significativamente superior ao projeto de referência, o modelo conseguiu obter um bom desempenho em algumas métricas de avaliação, portanto acredito que o projeto atendeu as expectativas, e conseguiu solucionar o problema proposto.

V.III Melhorias

Um aspecto que acredito que possa ser melhorado é aumentar ainda mais os dados a partir de fontes externas, apesar de ter lido no artigo que equipe obteve um desempenho inferior ao estender os dados, acredito que reduzir o desequilíbrio entre as diferentes classes, e conseguir mais dados para treinar a rede melhoraria o desempenho no conjunto de testes.

Outro aspecto que acredito que possa ser melhorado, é treinar a rede por mais épocas durante a fase de ajuste fino, a equipe da UNICAMP, durante essa etapa, treinou a rede por aproximadamente 300 épocas, como eu não tinha tanto tempo disponível, não pude treinar a rede durante tanto tempo, talvez a partir de um certo ponto a precisão comece a melhorar, e acredito que talvez se tivesse treinado mais a rede, tivesse obtido um desempenho melhor.

Analisando vários artigos, encontrei algumas equipes que conseguiram um desempenho muito superior ao meu, mas eles tinham muita experiência na área de processamento de imagens, diagnósticos médicos de doenças, conheciam técnicas e usavam tecnologias nas quais eu não tinha acesso, levando em consideração a complexidade da tarefa, e as limitações que eu tinha em termos de hardware, pois tive que executar todo o projeto em uma simples CPU.

Referências:

1 Menegola A, Tavares J, Fornaciali M, Li LT, Avila S, Valle E. RECOD Titans at ISIC Challenge 2017. Disponível em: < https://arxiv.org/pdf/1703.04819.pdf> Acesso em: 2 de Janeiro de 2019

2 Matsunaga K, Hamada A, Minagawa A, Koga H. Image Classification of Melanoma, Nevus and Seborrheic Keratosis by Deep Neural Network Ensemble. Disponível em: <https://arxiv.org/ftp/arxiv/papers/1703/1703.03108.pdf> Acesso em: 9 de Dezembro de 2018

3 González-Díaz I. Incorporating the Knowledge of Dermatologists to Convolutional Neural Networks for the Diagnosis of Skin Lesions. Disponível em: <https://arxiv.org/pdf/1703.01976.pdf > Acesso em: 9 de Dezembro de 2018

4 Codella Noel C. F, Gutman D, Celebi M. Emre, Helba B. SKIN LESION ANALYSIS TOWARD MELANOMA DETECTION: A CHALLENGE AT THE 2017 INTERNATIONAL SYMPOSIUM ON BIOMEDICAL IMAGING (ISBI), HOSTED BY THE INTERNATIONAL SKIN IMAGING COLLABORATION (ISIC). Disponível em: <https://arxiv.org/pdf/1710.05006.pdf> Acesso em: 16 de Novembro de 2018