



Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia Eletrônica

Aplicação de técnicas de XAI em redes neurais convolucionais na classificação de lesões de pele

Autor: João Vitor Rodrigues Baptista
Orientador: Dr. Nilton Correia da Silva

Brasília, DF
2019



João Vitor Rodrigues Baptista

**Aplicação de técnicas de XAI em redes neurais
convolucionais na classificação de lesões de pele**

Monografia submetida ao curso de graduação
em Engenharia Eletrônica da Universidade
de Brasília, como requisito parcial para ob-
tenção do Título de Bacharel em Engenharia
Eletrônica.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Nilton Correia da Silva

Brasília, DF

2019

João Vitor Rodrigues Baptista

Aplicação de técnicas de XAI em redes neurais convolucionais na classificação
de lesões de pele/ João Vitor Rodrigues Baptista. – Brasília, DF, 2019-

91 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Nilton Correia da Silva

Monografia submetida ao curso de graduação em – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2019.

1. Redes Neurais. 2. Explicabilidade. I. Dr. Nilton Correia da Silva. II.
Universidade de Brasília. III. Faculdade UnB Gama. IV. Aplicação de técnicas de
XAI em redes neurais convolucionais na classificação de lesões de pele

CDU 02:141:005.6

João Vitor Rodrigues Baptista

Aplicação de técnicas de XAI em redes neurais convolucionais na classificação de lesões de pele

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Dr. Nilton Correia da Silva
Orientador

Titulação e Nome do Professor
Convidado 01
Convidado 1

Titulação e Nome do Professor
Convidado 02
Convidado 2

Brasília, DF
2019

*Dedico esta, assim como todas as demais conquistas,
para a flor da minha vida Rosangela Rodrigues da Silva,
ao meu amado pai, João dos Santos Baptista e a irmã que tanto amo.*

Agradecimentos

Agradeço primeiramente a Deus, por ser essencial na minha vida. Por me guiar e me levar à lugares que nunca imaginei chegar e aprender coisas que nunca imaginei ser possível.

A minha incrível família por todo suporte e carinho que possuem por mim. Principalmente a mulher que é luz da minha vida, minha querida mãe, Rosangela Rodrigues da Silva e meu amado pai, João dos Santos Baptista que me aturam todos os dias. À minha super irmã Maria Victória Rodrigues Baptista.

Agradeço ao meu orientador, Prof. Dr. Nilton Correia da Silva, por propor esse desafio sensacional.

“(. . .) as pessoas precisam de capacidade para extrair um sentido da informação, perceber a diferença entre o que é importante e o que não é, e acima de tudo combinar os muitos fragmentos de informação num amplo quadro do mundo.

Yuval Noah Harari.

Resumo

Modelos de *machine learning* estão cada vez mais presente no dia a dia. Com o crescimento do poder computacional verificou-se um aumento na complexidade desses modelos. Devido a alta complexidade, principalmente, em redes neurais profundas conhecidas como “*caixas pretas*“ pois é difícil entender como esses modelos lidam como os dados de entrada. O presente trabalho tem como finalidade aplicar conceitos de XAI para entender quais são as bases de decisões de modelos de redes neurais convolucionais. Para tanto, foi desenvolvido um modelo que classifica 9 tipos de lesões de pele, sendo 4 malignas. Essas doenças afetam mais de 14,1 milhões de pacientes e sem sido a cause de mais de 8,2 milhões de mortes no mundo. Para auxiliar os diagnósticos clínicos é necessário avaliar o processo de decisão do modelo. Por tratar-se de decisões sensíveis é necessário confiabilidade baseado na interpretabilidade do modelo. Com o auxilio do modelo e os *insight* gerados a partir da explicabilidade dos padrões aprendidos, pode-se criar novas metodologias de classificação de lesões por profissionais da saúde.

Palavras-chaves: Redes neurais artificiais, explicabilidade, lesões de pele, interpretabilidade.

Abstract

Machine learning models has become more present in every-day life. Computing power has grown exponentially in the last few years as well as models complexity. Due to the high models complexity, especially, in deep neural network and they are now known as "black boxes", because it's too difficult to understand how they deal with input data. This work focus on applying XAI concepts in order to understand what are the base decision process done by a deep neural network. Therefore, a skin's lesion classification model was developed to classify 9 classes, being 4 of these malignant, including Malignant Melanoma and Basal Cell Carcinoma. Those diseases threaten more than 14.1 million patients and had been the cause of more than 8.2 million deaths, worldwide. In order to assist clinic diagnosis is necessary to evaluate model's decision process. As the model deals with sensitivity decisions is required reliability based on model's interpretability. As model's insights have grown they will be useful in further classification's methodologies in health industry.

Key-words: Neural network. explainability. skin lesions, interpretability.

Listas de ilustrações

| | |
|---|----|
| Figura 1 – Comparação entre o neurônio e o perceptron. | 18 |
| Figura 2 – Operação de <i>max-pooling</i> usando um filtro 2 x 2 e stide 2. | 22 |
| Figura 3 – Comparativo entre as topologias pela acurácia em função da operação em função do tamanho da base de dados | 25 |
| Figura 4 – Blocos de construção da ResNet | 26 |
| Figura 5 – Diagrama das lesões separadas por categorias | 30 |
| Figura 6 – Representação da pele humana sem lesão | 33 |
| Figura 7 – Lesões de interesse neste trabalho | 35 |
| Figura 8 – Imagem da base do ISIC da Basal cell carcinoma | 36 |
| Figura 9 – Diagrama de processos utilizados para a preparação da base de dados | 40 |
| Figura 10 – Detalhes técnicos da infraestrutura | 42 |
| Figura 11 – Escopo da área XAI | 47 |
| Figura 12 – O panorama geral de interpretabilidade para modelos de machine learning | 49 |
| Figura 13 – Explicação usando LIME para as top 3 classes na classificação de uma imagem feita pelo modelo da Google Inception. | 51 |
| Figura 14 – Exemplos de visualização do gradiente das imagens | 52 |
| Figura 15 – Multiplicação da imagem de entrada com o gradiente de ativação res- petivo | 53 |
| Figura 16 – Representação dos filtros de cada camada convolucional de uma rede neural convolucional. | 54 |
| Figura 17 – Exemplo de mapa de calor usando a técnica do GradCAM | 55 |
| Figura 18 – Matriz de confusão do modelo para as 9 lesões | 89 |
| Figura 19 – Curva ROC para o restante das lesões. Continuação 1/2. | 90 |
| Figura 20 – Curva ROC para o restante das lesões. Continuação 2/2. | 91 |

Lista de tabelas

| | |
|---|----|
| Tabela 1 – Numero de amostras da base do MED-NODE | 27 |
| Tabela 2 – Numero de amostras da base do Edinburgh | 27 |
| Tabela 3 – Numero de amostras da base do ISIC | 28 |
| Tabela 4 – Numero de amostras da base do Atlas | 29 |
| Tabela 5 – Transformações aplicadas no processo de <i>data augmentation</i> | 39 |
| Tabela 6 – Numero de amostras da base agregada | 41 |
| Tabela 7 – Numero de amostras finais | 41 |
| Tabela 8 – Comparativo entre trabalhos correlatos as AUC | 45 |
| Tabela 9 – Cronograma de atividades para o desenvolvimento das proximas etapas | 56 |
| Tabela 10 – Reporte das métricas para cada lesão | 88 |

Listas de abreviaturas e siglas

| | |
|------|--|
| BCC | Basal Cell Carcinoma |
| SCC | Squamous Cell Carcinoma |
| IA | Inteligência Artificial |
| ML | Machine Learning |
| XAI | Explainable Artificial Intelligence |
| DNN | Deep Neural Network |
| CNN | Convolutional Neural Network |
| ROC | Receiver Operating Characteristic |
| AUC | Area Under the Curve |
| UMCG | Department of Dermatology at the University Medical Center Groningen |
| ISIC | International Skin Imaging Collaboration |
| INCA | Instituto Nacional de Câncer |
| INC | National Cancer Institute |
| SBD | Sociedade Brasileira de Dermatologia |
| GPU | Unidade de processamento gráfico |
| SGD | Stochastic gradient descent |
| ReLU | Rectifier Linear Unit |

Lista de símbolos

| | |
|-------------|--------------------------------|
| W | Volume da entrada |
| F | Dimensões do campo receptivo |
| P | tamanho do <i>zero-padding</i> |
| S | Valor do <i>stride</i> |
| H' | Dimensão da altura de saída |
| W' | Dimensão da largura de saída |
| t_p | Verdadeiros Positivos |
| t_n | Verdadeiros Negativos |
| f_n | Falsos negativos |
| f_p | Falsos positivos |
| s | Total de Amostras |
| \in | Pertence |
| $\Omega(g)$ | Complexidade do modelo |

Sumário

| | | |
|------------|--|-----------|
| 1 | INTRODUÇÃO | 16 |
| 1.1 | Contexto | 16 |
| 1.2 | Organização do trabalho | 17 |
| 2 | REFERENCIAL TEÓRICO | 18 |
| 2.1 | Redes Neurais | 18 |
| 2.1.0.1 | Breve histórico | 19 |
| 2.1.0.2 | Rede Neural Convolucional | 19 |
| 2.1.0.2.1 | Camada convolucional | 20 |
| 2.1.0.2.2 | Camada de <i>pooling</i> | 21 |
| 2.1.0.2.3 | Camada <i>fully-connected</i> | 22 |
| 2.1.0.2.4 | Funções de ativação | 22 |
| 2.2 | Treinamento da rede neural | 23 |
| 2.3 | Avaliação da rede neural | 23 |
| 2.3.1 | Métricas de treino e validação | 24 |
| 2.3.2 | Métricas para teste | 24 |
| 2.4 | Arquitetura de rede neural artificial | 25 |
| 2.4.1 | ResNet | 25 |
| 2.5 | Base de dados | 26 |
| 2.5.1 | MED-NODE | 26 |
| 2.5.2 | Edinburgh | 27 |
| 2.5.3 | ISIC | 27 |
| 2.5.4 | Atlas | 28 |
| 2.6 | Lesões de interesse | 28 |
| 2.6.1 | Actinic Keratosis | 29 |
| 2.6.2 | Basal cell carcinoma | 29 |
| 2.6.3 | Dermatofibroma | 31 |
| 2.6.4 | Hemangioma | 31 |
| 2.6.5 | Intraepithelial carcinoma | 32 |
| 2.6.6 | Malignant melanoma | 32 |
| 2.6.7 | Melanocytic nevus | 33 |
| 2.6.8 | Pyogenic Granuloma | 33 |
| 2.6.9 | Squamous cell carcinoma | 33 |
| 2.7 | Natureza da base de dados | 34 |
| 2.7.1 | Dificuldades | 34 |

| | | |
|--|---|-----------|
| 2.7.2 | Amostra de dados | 37 |
| 2.7.3 | Rótulos | 37 |
| 2.8 | Preparação das amostras | 37 |
| 2.8.1 | Transfer Learning | 37 |
| 2.8.2 | Data augmentation | 38 |
| 2.8.2.1 | Metodos de Augmentation | 38 |
| 2.8.2.1.1 | Transformações | 39 |
| 2.9 | Preparação da base de dados | 39 |
| 3 | RESULTADOS PRÉVIOS | 42 |
| 3.1 | Infraestrutura | 42 |
| 3.2 | Classificação | 43 |
| 3.2.1 | Processo de treinamento | 43 |
| 3.2.2 | Parâmetros livres | 43 |
| 3.2.3 | Resultados | 44 |
| 4 | PRÓXIMOS PASSOS | 46 |
| 4.1 | Interpretabilidade e Explicabilidade | 46 |
| 4.1.1 | Conceitos | 47 |
| 4.2 | Métodos de interpretabilidade | 48 |
| 4.2.1 | Método Model-agnostic | 48 |
| 4.2.1.1 | Perturbation-based | 50 |
| 4.2.1.2 | LIME | 50 |
| 4.2.1.3 | Gradient-based | 52 |
| 4.2.1.4 | GradCAM | 53 |
| 4.3 | Proposta e cronograma de atividades | 54 |
| 4.3.1 | Cronograma de Atividades | 56 |
| 4.3.2 | Resultados esperados | 56 |
| REFERÊNCIAS | | 57 |
| APÊNDICES | | 62 |
| APÊNDICE A – WEB SCRAPING | | 63 |
| APÊNDICE B – CONVERSÃO DAS IMAGENS E REDIMENSIONAMENTO PRÉVIO | | 65 |
| APÊNDICE C – SEPARAÇÃO DA BASE DE FORMA ESTRATIFICADA | | 66 |

| | |
|--|----|
| APÊNDICE D – DATA AUGMENTATION | 67 |
| APÊNDICE E – TREINAMENTO E VALIDAÇÃO DA REDE | 70 |
| APÊNDICE F – TESTE E CRIAÇÃO DAS MÉTRICAS | 78 |
| APÊNDICE G – MÉTRICAS DE AVALIAÇÃO PARA O MELHOR EXPERIMENTO | 88 |

1 Introdução

1.1 Contexto

Nos dias atuais, com a invenção e a evolução das rede neurais (HAYKIN, 1998), os passos para procurar características em imagens vem reduzindo-se a definir operações in camadas da redes neurais. Então a área de classificação de imagens e a visão humana ganhou um grande aliado. Portanto, diversas pesquisas então usando essa abordagem para classificar diversos tipos de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; MATSUGU et al., 2003). Contudo, esses trabalhos trouxeram diversos desafios que serão enfrentado no futuro.

Recentemente, trabalhos relacionados a classificações de lesões de pele tem feito grandes avanços por meio das redes neurais profundas. Com esses resultados muitas barreiras tem sido quebradas. Contudo, o campo de dermatologia será revolucionado no futuro.

O presente trabalho faz o uso de algorítimos de redes neurais profundas, especificamente, redes neurais convolucionais para abordar o problema de classificação de lesões de pele.

Contudo, esses modelos de redes neurais profundas são complexos e o processo de tomada de decisão desses modelos são pouco conhecidos pelos seres humanos e até para os próprios engenheiros projetistas. Por essa razão esses modelos são apelidados como "caixas pretas" por não se entender o processo de predição internamente. O projetista treina com diversos amostras rotulados por um tempo longo, em seguida usa o modelo para predizer novas amostras, porem não preocupa-se como funciona o processo interno do modelo.

Na área de classificação de imagens clinicas as amostras são sensíveis pois o objeto de classificação é a saúde do paciente, para tanto o processo de tomada de decisão deve ser baseada em fatos concretos. Nesse sentido, um modelo que recebe como entrada uma amostra e responde na saída puramente as probabilidades pode ser subjetivo e pouco concreto pois não existe uma explicação detalhada do processo que levou as decisões.

Por isso é necessário métodos para avaliação dos processos internos de decisão de modelos complexos. O presente trabalho tem como finalidade interpretar um modelo de classificação de lesão de peles baseado no processo interno de decisão do modelo para explicar quais foram os fatores ou características que determinada lesão possui para determinar as probabilidades de saída.

1.2 Organização do trabalho

Esse trabalho esta divido em quatro capítulos, incluindo o capítulo de introdução que compreende a contextualização, o problema, os objetivos e a metodologia selecionada.

O capítulo dois referencial teórico faz uma descrição das técnicas e recursos, com mais enfase nos que foram utilizado para viabilizar o treino da rede neural convolucional. Esse capítulo esta descrito como foi a sequencia de passos para a preparação da base de dados final para o treinamento do modelo e de onde foi retirados os amostras utilizados, bem como todas as operações feitas nas amostras

No capítulo três esta a especificação e métricas de avaliação e resultados obtidos na fase de treinamento, validação e teste do modelo proposto. Bem como o ambiente onde foram conduzidos os experimentos e os parâmetros livres utilizados. Para finalizar esse capítulo faz uma comparação com trabalhos correlatos usados como base para desenvolvido do presente trabalho.

No capítulo quatro e ultimo, foi feita uma breve introdução das metodologias que serão utilizadas na sequencia do trabalho. Essas metodologias estão organizadas pelo nível de complexidade aparente, da mais simples para as mais complexas. Com respectivas referências e repositórios de códigos fontes que serão utilizados extensivamente durante o desenvolver das atividades. Para finalizar esse capítulo foi feito um cronograma pessimista de atividades que será utilizado para iniciar as atividades da próxima etapa. Finalmente, um texto breve sobre os possíveis resultados apos a implementação das técnicas descrita no respectivo capítulo.

Nos apêndices estão todos os recursos extras de desenvolvimento como, imagens das métricas do capítulo 3 e códigos fonte de todo o processo de desenvolvimento.

2 Referencial

2.1 Redes Neurais

São algorítimos no campo de aprendizado de maquina que emergiu de estudos biológicos do neurônio do córtex cerebral (ROSENBLATT, 1958). De maneira semelhante ao conjunto de neurônios biológicos contidos no cérebro humano. As redes neurais são representações computacionais da sinapse humana que possibilitam executar tarefas extremamente complexas.

Analogamente com os neurônios biológicos, o perceptron é a abstração computacional que usa a saída do neurônio anterior para propagar o sinal para os próximos neurônios. Esse processo é realizado através da conexão entre a dendritos com o terminal *Axon* do neurônio anterior (Fig. 1), semelhante, o perceptron usa a saída do perceptron anterior como entrada, processa o sinal e passa para os perceptrons restantes.

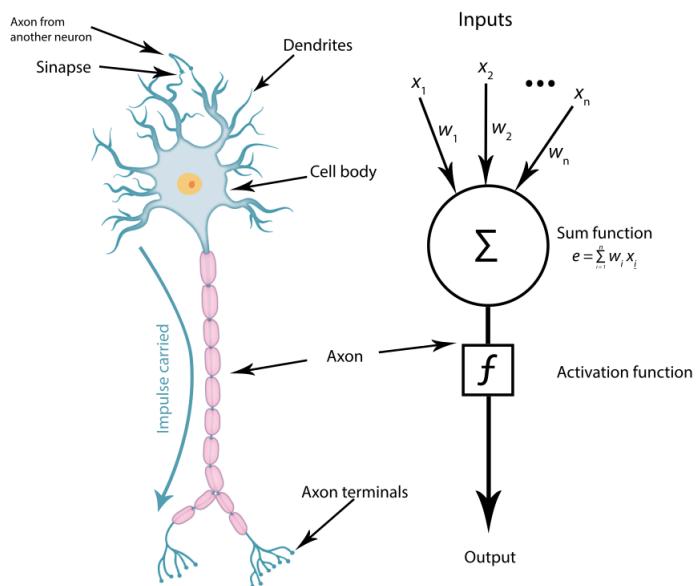


Figura 1 – Comparaçāo entre o neurônio e o perceptron.
(MENDES; SILVA, 2018)

O neurônio tem a finalidade de processar o sinal somando todas as entradas recebidas, caso o resultado do sinal recebido for forte o suficiente para ativar o neurônio, o sinal é passado para os próximos neurônios. O sinal é processado no *Axon* do neurônio que é comparado com a função de ativação no perceptron. Apenas quando os requisitos são alcançados, o neurônio é capaz de produzir o sinal que será repassado como entrada dos próximos neurônios da rede.

Por mais simples que pareça essa estrutura organizacional é responsável para conduzir e executar tarefas extremamente complexas. No entanto, as redes neurais artificiais ainda não conseguem reproduzir a capacidade cognitiva total do cérebro humano.

2.1.0.1 Breve histórico

Redes simples modeladas em circuitos elétricos datadas da década de 40 são reconhecidas na literatura (MCCULLOCH; PITTS, 1988). Contudo, sua popularidade deu-se, em parte, na década de 80 com a reinvenção de *loops* de realimentação dinâmicos conectando os neurônios usando linhas bidirecionais (WERBOS, 1981).

Backpropagation, conhecido também como *backprop*, é um algorítimo de optimização que tem como objetivo o reconhecimento de padrões baseados em correção de erros. Isso ocorre devido à retro-propagação dos erros de saída de uma rede neural artificial através das camadas precedentes para que dessa maneira, sejam balanceados os pesos de entrada da rede. Contudo, esse algorítimo era computacionalmente custoso para a época. Então, até os recentes avanços no desenvolvimento de hardware, computação paralela e unidades de processamento gráficas que o uso das redes neurais artificiais tornou-se mais proeminente.

Existem vários tipos de redes neurais artificiais e topologias, no presente trabalho será utilizada uma Rede Neural Convolucional, que é uma variação das perceptron de múltiplas camadas. As redes convolucionais também são inspiradas nos processos biológicos (MATSUGU et al., 2003).

2.1.0.2 Rede Neural Convolucional

É uma topologia proposta por (HUBEL; WIESEL, 1968) baseada no modo que as conexões dos neurônios de animais estão dispostas de forma dispersa. Portanto, essa característica permite que a rede neural artificial tenha maior capacidade de reconhecer características individuais. Sendo assim muito aplicada na classificação de imagens, contudo outras áreas tem-se beneficiado dessa topologia.

Quando aplicada em imagens, essa rede tem os neurônios arranjados em 3 dimensões $A \times L \times P$, representando a altura, largura e profundidade respectivamente, então são feitas operações lineares chamadas convolução. Diferente de multiplicações entre matrizes como é feita em redes *fully-connected* essa topologia faz convoluções entre a imagem e um filtro.

Portanto, uma rede neural convolucional, tipicamente, possui um conjunto de camadas denominadas camadas convolucionais, camadas de *pooling* e a camadas *fully-connected*.

2.1.0.2.1 Camada convolucional

É a camada mais importante, por isso da-se o nome à topologia o nome. Essa camada tem por finalidade calcular e detectar características específicas de cada ponto da entrada. Em linhas gerais, a convolução é uma operação matemática amplamente utilizada na área processamento de sinais. Quando discreta e bidimensional, é o somatório dos produtos de duas matrizes ao longo da região subentendida pela superposição delas em função do deslocamento existente entre elas.

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n) \quad (2.1)$$

Onde na Equação 2.1 I representa a matriz da imagem de entrada, K é a matriz de *kernel* ou filtro, m e n são a altura e a largura da matriz do *kernel*, geralmente é uma matriz quadrada ($m = n$), i e j são os eixos horizontal e vertical, respectivamente.

O filtro em geral é uma matriz quadrada com uma série de pesos que são responsáveis pela formação dos mapas de características, diferentes filtros detectam diferentes padrões. Esse processo é feito ao deslizar *kernel* por toda a matriz de entrada. Essa operação é feita na fase de passagem da rede, ou seja, quando as entradas passam através de todas as camadas até alcançar a saída.

A saída da rede neural convolucional depende de alguns parâmetros livres que controlam a quantidade de neurônios existentes no volume de saída e como eles estão dispostos. A seguir serão apresentados:

- **tamanho do campo receptivo local:** O campo é usado para reconhecimento de padrões locais de forma a ignorar ruídos ou evitar padrões que poderiam influenciar a habilidade da rede de reconhecer padrões. É denominada esta região, onde cada conexão possui determinado peso e cada neurônio reconhece um viés global (NIELSEN; MICHAEL, 2018). O tamanho ($m \times n$) é o mesmo para todos os neurônios na mesma camada, normalmente 3×3 ou 5×5 , dependendo do tamanho da matriz de entrada.
- **Profundidade:** A profundidade do volume de saída está atrelada ao número de filtros aplicados na matriz de entrada. Portanto, é o número de neurônios que serão estimulados pelo mesmo campo receptivo. No entanto, cada campo receptivo aprende uma característica diferente da entrada.
- **Stride:** Esse parâmetro livre é responsável por indicar o deslocamento, tanto horizontal quanto vertical, do movimento que o filtro deve fazer antes de fazer a convolução com a matriz de entrada.

- **Padding:** Algumas vezes a o filtro não encaixa perfeitamente nas dimensões da matriz de entrada. Pode-se então, complementar as bordas com zeros (*zero-padding*) ou recortar a parte da imagem que o filtro não encaixa (*valid padding*)

Baseado nos parâmetros livres é possível calcular as dimensões da saída baseado nas dimensões de entradas e nos valores de cada parâmetro utilizado usando a Equação 2.2.

$$\text{saída} = \frac{W - F + 2P}{S} + 1 \quad (2.2)$$

Onde a entrada tem volume W , o campo receptivo tem tamanho F tamanho do *zero-padding* P e o valor do *stride* S .

As redes convolucionais possuem os chamados pesos compartilhado. Estes pesos são aprendidos e compartilhados entre os neurônios de mesma profundidade, reduzindo a ordem de magnitude do numero de parametros. Ou seja, os neurônios da primeira camada oculta detectam um padrão que é o mesmo das outras regiões da imagem. Esta é uma característica que torna a CNN adaptativa em relação a diferentes representações que um padrão possa ter (NIELSEN; MICHAEL, 2018).

2.1.0.2.2 Camada de *pooling*

É uma operação usada em intervalos regulares entre as camadas de convolução. É responsável por generalizar a posição dos padrões encontrados pela camada de convolução, isso é alcançado devido a diminuição da dimensão espacial sem de fator descartar informação.

A operação mais comum é conhecida como *max-pooling* (Fig. 2) que retira o elemento de maior valor em um determinado intervalo de análise. Existe o *avarage pooling* que faz a média entre os elementos de um determinada região.

Baseado em parâmetros livres, é possível comparar a matriz de entrada com a saída. Assim como as camadas convolucionais a camada de *pooling* possui *stride* e tamanho do campo receptivo ao qual define o comportamento das operações. Com os parâmetros livres definidos é possível através das Equações 2.3 e 2.4 é possível analisar a dimensão de saída.

$$H' = \frac{H - H_p}{S} + 1 \quad (2.3)$$

$$W' = \frac{W - W_p}{S} + 1 \quad (2.4)$$

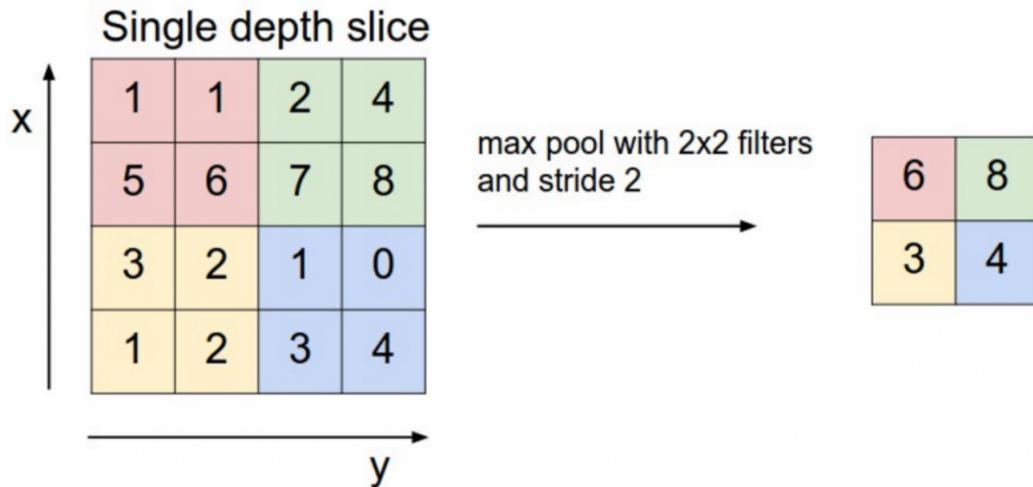


Figura 2 – Operação de *max-pooling* usando um filtro 2 x 2 e stide 2.
(LI; KARPATHY; JOHNSON,)

Na Equação 2.3 H' e na Equação 2.4 W' representam, respectivamente, altura e largura da saída, H e W são a altura e largura de entrada, respectivamente, H_p e W_p são as dimensões do campo receptivo do filtro aplicados na entrada.

2.1.0.2.3 Camada *fully-connected*

A camada totalmente conectada é uma camada que também está presente nas redes neurais artificiais em geral (as não necessariamente convolucionais), ela é responsável por conectar as camadas sem utilizar pesos compartilhados (HAFEMANN; SABOURIN; OLIVEIRA, 2016). Cada neurônio da camada anterior está conectado a algum neurônio da camada a vir, e adiciona uma camada saída com o número de neurônios em relação ao número de classes presentes no estudo em questão. por isso o termo totalmente conectado. É uma forma barata de aprender combinações não-lineares desses recursos.

2.1.0.2.4 Funções de ativação

As funções de ativação desempenham papel de adicionar não-linearidade nas sobre entrada. Existem diversas funções, contudo a mais utilizada é a *ReLU* (*Rectifier Linear Unit*) representada na Equação 2.5. Devido ao fator de acelerar a convergência de otimizadores como *SGD* (*stochastic gradient descent*), comprador com outras funções.

$$\text{ReLU}(x) = \max(0, x) \quad (2.5)$$

2.2 Treinamento da rede neural

Para obter uma boa generalização as redes neurais convolucionais necessitam de grandes quantidades de dados. A quantidade de dados, topologia da rede e a escolha dos parâmetros livres são essenciais para a potencializar o desempenho da rede.

O processo de treinamento de uma rede pode ser simplificado em duas partes: *forward pass* e *backpropagation*.

- **forward pass:** É quando as entradas são passadas pelas camadas até alcançar a saída. Nesse momento, a entrada sofre diversas operações e a aprendizagem ou os pesos são considerações nessas operações (HAYKIN, 1998).
- **backpropagation:** Por outro lado, ao fazer a passagem das entradas por toda a rede deve-se atualizar os pesos para corrigir a saída dado uma entrada. Por tanto, essa operação é um tipo de realimentação usada para atualizar os parâmetros da rede (HAYKIN, 1998).

2.3 Avaliação da rede neural

Apos a etapa de treinamento da rede é necessário avaliar os resultados. Por essa razão existe métricas que podem ser utilizadas para avaliar e determinar o quanto satisfatório é um modelo.

A etapa de avaliação não necessariamente é feita ao final do treinamento, por esse motivo, existem abordagens que fazem o treinamento com validação usando um subset derivado do dataset principal, ou seja, dados que a rede não teve contato durante o treinamento. Esse procedimento é referenciado como “Validação” que é muito importante para observar casos de *underfitting* ou *overfitting* e ajudar nos ajustes dos parâmetros livres da rede.

- **underfitting:** Normalmente acontece quando o modelo não se adapta aos dados de entrada, sendo incapaz de aprender ou predizer de forma efetiva.
- **overfitting:** É quando o modelo performa muito bem no dado de treinamento, porem é incapaz de predizer de forma efetiva dados não conhecidos. Em termos gerais, o modelo apenas memoriza os dados de treinamento, não oferecendo uma boa generalização.

A seguir esta definido as métricas utilizadas para treinar, avaliar e testar o modelo. Como foram feitos experimentos para ajustar o melhor conjunto de parâmetros livres foi necessário definir métricas de comparação entre os modelos.

2.3.1 Métricas de treino e validação

Como foi utilizado treino e validação, as métricas utilizadas foi principalmente a acurácia e a função de perda *Cross-Entropy*. Essas métricas tem como principal objetivo fornecer o quanto bom o modelo está indo nas fases de treino e validação e a com a função de perda é possível observar casos de *underfitting* ou *overfitting*

A formula para calcular a acurácia esta descrita na Equação 2.6 Onde t_p são os verdadeiros positivos, t_n são os verdadeiros negativos e s é total de amostras utilizadas

$$\text{Acurácia} = \frac{\sum t_p + \sum t_n}{s} \quad (2.6)$$

2.3.2 Métricas para teste

Apos o treinamento do modelo, é feita uma etapa final que visa testar o modelo com dados que não foram utilizados no momento do teste e da validação. Com as predições que o modelo fornece e os verdadeiros rótulos dos dados é possível criar avaliações mais refinadas, como a matriz de confusão do modelo. A partir da matriz de confusão do modelo é possível derivar diversas outras métricas auxiliares para o modelo, como a precisão, recall (*sensitivity*) e a acurácia.

Na Equação. 2.7 e Equação 2.8 pode-se observado as equações para calculo da precisão e recall respectivamente. Onde t_p são os verdadeiros positivos, f_n são os falsos negativos e f_p são os falsos positivos.

$$\text{Precisão} = \frac{t_p}{t_p + f_p} \quad (2.7)$$

$$\text{Recall} = \frac{t_p}{t_p + f_n} \quad (2.8)$$

Outra métrica importante para testar o modelo foi a UAC (Area Under the Curve), junto com a curva ROC (Receiver Operating Characteristic). A curva de ROC mapeia a *sensitivity* (probabilidade de detecção) em função da $1 - specificity$ (probabilidade de falsos positivos). Tipicamente, essa métrica é implementada em sistemas que analisa o quanto exato é o diagnóstico do estado de um paciente em termos de doenças (SWETS, 1986).

Essas métricas são de suma importância para avaliar de forma sólida como o modelo vai performar com dados novos. Com o recall é possível calcular a probabilidade de detecção da lesão, a precisão possibilita o calculo do grau de certeza que a lesão será classificada como verdadeiro positivo e a UAC calcula o quanto exato o é diagnóstico.

2.4 Arquitetura de rede neural artificial

Como as redes neurais convolucionais são algorítimos muito populares existe uma grande diversidade de pesquisas relacionadas. Contudo, é necessário usar algum tipo de topologia para esse algorítimo. A topologia deve levar em conta como estão dispostas as camadas da rede.

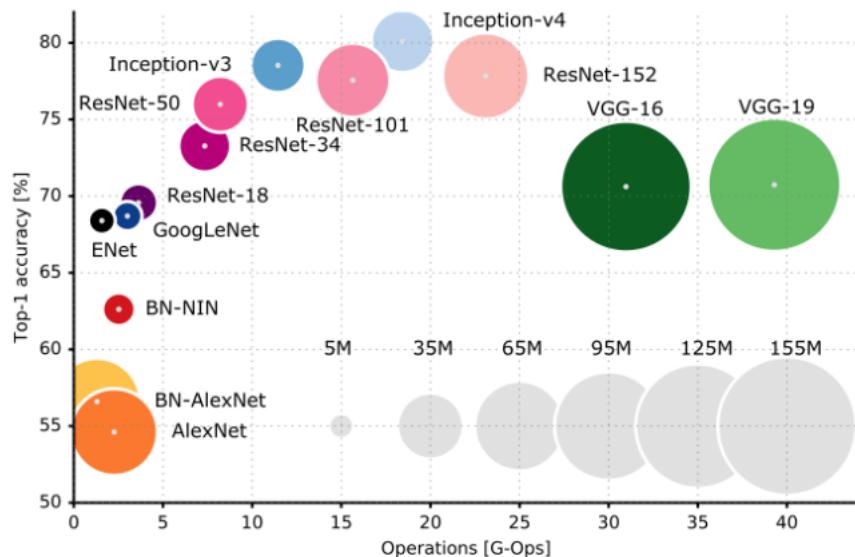


Figura 3 – Comparativo entre as topologias pela acurácia em função da operação em função do tamanho da base de dados
 (CANZIANI; PASZKE; CULURCIELLO, 2017)

Existe diversas abordagens. Um exemplo, pode-se criar a topologia customizadas do princípio, ao passo que, pode-se escolher topologias que já foram criadas e testadas por outros pesquisadores e que se provaram muito boa para determinada tarefa (Fig. 3) ou até customizar redes já consolidadas.

Para simplificar a criação da topologia o presente trabalho é baseado em uma topologia específica que provou-se ser o estado da arte em classificação de imagens em competições de classificação de imagens.

2.4.1 ResNet

Foi uma topologia proposta por pesquisadores da Microsoft em 2015, onde ganhou uma o desafio do ImageNet (RUSSAKOVSKY et al., 2014). Desde introdução dos seus conceitos, essa topologia tem sido usado de forma extensiva pela comunidade. Devido ao problema de degradação da acurácia que está enraizado nas redes neurais profundas, pois a medida que são adicionados novas camadas o erro de treinamento aumenta (SRIVASTAVA; GREFF; SCHMIDHUBER, 2015).

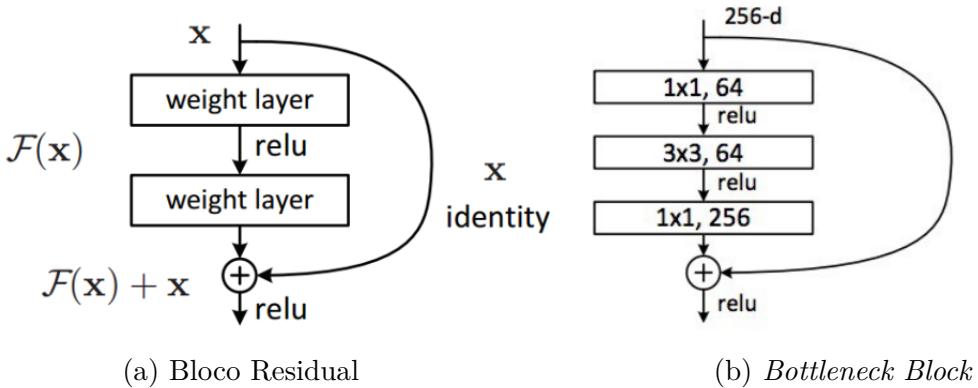


Figura 4 – Blocos de construção da ResNet
(HE et al., 2015)

A topologia introduz um novo conceito de treinamento baseado em blocos residuais. O bloco residual passa a entrada direto para o próximo bloco ou camada, para então somar a entrada atual com a antiga entrada (Fig. 4a). Mais ainda, ao passar a entrada por duas camadas, observou-se um comportamento parecido com um pequeno classificador.

Contudo, foi observado que passar a entrada entre duas camadas gerava um grande custo computacional pois adicionava mais parâmetros para serem calculados o que dificultava a implementação. Para resolver o problema foi adicionado *bottleneck layers* (Fig. 4b) para reduzir o numero de parâmetros em cada operação (HE et al., 2015).

Portanto, baseado na extensiva utilização dessa topologia e a utilização em trabalhos correlatos como (HAN et al., 2018) treinando aproximadamente 855,370 imagens de lesões de pele e conseguindo alcançar bons resultados.

2.5 Base de dados

Durante a criação da base de dados utilizados para o presente trabalho, foi observado que não existe muitos recursos disponíveis para conseguir imagens medicas de lesões de pele. Por essa razão a única restrição imposta ao montar a base é que devem ser imagens clínicas.

Obedecendo esse critério, três bases principais foram utilizadas e a utilização de *web scraping* em sites confiáveis para fazer um complemento na base de dados. A seguir será detalhado essas bases de dados e as quantidades de imagens e o processo de utilização do *web scraping* para a criação de uma base própria para suplementar as bases principais.

2.5.1 MED-NODE

A primeira base de dados utilizado foi fornecida pelo Department of Dermatology at the University Medical Center Groningen (UMCG) (GIOTIS et al., 2015). As lesões e

numero de amostras estão apresentadas na Tabela 1. Essa base de dados pode ser usadas mediante a citação explícita e pode ser encontrada facilmente.¹

Tabela 1 – Número de amostras da base do MED-NODE

| Tipo de lesão | Amostras |
|----------------------|-----------------|
| Melanoma | 70 |
| Melanocytic Nevus | 100 |
| Total | 170 |

2.5.2 Edinburgh

Essa é uma base de dados disponível para ser comprada mediante a uma licença de utilização². É a base mais completa encontrada na internet. São imagens de diagnósticos baseada na avaliação de profissionais coletadas em condições padronizadas. Divida em 10 tipos de lesões, não balanceadas, ou seja, algumas lesões possuem mais imagens do que outras, imagens são de diferentes dimensões, totalizando 1300 amostras. Um resumo do numero de amostras com suas respectivas classes esta apresentada na Tabela 2.

Tabela 2 – Número de amostras da base do Edinburgh

| Tipo de lesão | Amostras |
|---------------------------|-----------------|
| Actinic Keratosis | 45 |
| Basal Cell Carcinoma | 239 |
| Melanocytic Nevus | 331 |
| Seborrhoeic Keratosis | 257 |
| Squamous Cell Carcinoma | 88 |
| Intraepithelial Carcinoma | 78 |
| Pyogenic Granuloma | 24 |
| Haemangioma | 97 |
| Dermatofibroma | 65 |
| Malignant Melanoma | 76 |
| Total | 1300 |

2.5.3 ISIC

É um repositório de imagens de lesões de pele público chamado International Skin Imaging Collaboration (ISIC)³. É um projeto que possui um repositório com acesso a imagens clínicas de lesões. Esse projeto é uma parceria entre a indústria e as universidades com o objetivo de reduzir mortes por câncer e biopsias desnecessárias (ISDIS, 2018).

¹ Disponível em: <http://www.cs.rug.nl/~imaging/databases/melanoma_naevi/>

² Disponível em: <<https://licensing.eri.ed.ac.uk/i/software/dermofit-image-library.html>>

³ Disponível em: <<https://www.isic-archive.com>>

Esse repositório de imagens é organizado, imagens possuem *metadata* que contem o diagnóstico, idade aproximada do paciente e o gênero. Em trabalhos futuros, esse tipo de dado pode ser utilizado para a criação de modelos mais sofisticados. A Tabela 3 mostra os tipos de lesões e a quantidade de amostras.

Tabela 3 – Número de amostras da base do ISIC

| Tipo de lesão | Amostras |
|-------------------------|-----------------|
| Actinic Keratosis | 132 |
| Basal Cell Carcinoma | 480 |
| Seborrhoeic Keratosis | 339 |
| Squamous Cell Carcinoma | 226 |
| Dermatofibroma | 122 |
| Total | 1299 |

2.5.4 Atlas

Essa base é um conjunto de imagens com o objetivo de suplementar as bases mais padronizadas. Foi utilizado *scripts* de extração de imagens em diferentes sites dermatológicos que pode ser visto no Apêndice A.⁴

A principal diferença dessa base é que as imagens não foram coletadas sobre condições padrões, assim sendo, as imagens possuem qualidades diferentes, luminosidade, posição, dimensões diferentes. Portanto, é uma base mais heterogênea comparada com as outras utilizadas.

A procedência de como foram diagnosticadas as doenças não é conhecida, contudo são referências utilizadas pela literatura correlata, portanto foi utilizados no trabalho. Porem, antes de seguir com as transformações para serem utilizadas para o treinamento da rede, essa base foi inspecionada, empiricamente, de forma a descartar imagens que possivelmente não corresponde-se as classes estudadas.

2.6 Lesões de interesse

As lesões de interesse podem ser divididas em dois grandes grupos, o primeiro é composto de lesões que efetivamente são tipos de câncer e são extremamente perigosos para a saúde do paciente, por outro lado, o segundo grupo é composto de lesões que não oferecem riscos mais imediatos a o paciência.

⁴ Sites utilizados para a base: <<http://www.dermatlas.net/>>, <<http://www.dermatlas.net/>>, <<http://www.dermis.net/dermisroot/en/home/index.htm>>, <<http://www.meddean.luc.edu/lumen/MedEd/medicine/dermatology/melton/atlas.htm>>, <<http://www.dermatoweb.net/>>, <<http://www.atlasdermatologico.com.br/>>, <<http://www.dandermpdv.is.kkh.dk/atlas/index.html>>, <<http://www.hellenicdermatlas.com/en/?params=en>>.

Tabela 4 – Numero de amostras da base do Atlas

| Tipo de lesão | Amostras |
|---------------------------|-----------------|
| Actinic Keratosis | 24 |
| Basal Cell Carcinoma | 151 |
| Melanocytic Nevus | 71 |
| Seborrhoeic Keratosis | 107 |
| Squamous Cell Carcinoma | 103 |
| Intraepithelial Carcinoma | 81 |
| Pyogenic Granuloma | 74 |
| Haemangioma | 76 |
| Dermatofibroma | 19 |
| Malignant Melanoma | 65 |
| Total | 771 |

Foi coletado, inicialmente, imagens de 12 tipos de lesões diferentes, contudo a medida que foram realizado os experimentos, foi descartado 3 tipos de lesões, essa questão será abordada mais detalhadamente nas próximas secções deste trabalho. A seguir será apresentado um breve resumo de cada tipo de lesão.

2.6.1 Actinic Keratosis

É um tipo de lesão mais frequente em pessoas com tons de pele mais claros, visto que possuem menos proteção dos pigmentos da pele e portanto mais expostos a luz solar. Conhecida como solar keratosis é um tipo de lesão mais frequente nas pessoas que são expostas ao sol mais constantemente (MOY, 2000).

Portanto, lugares perto da linha do equador que sofrem com maior incidência da luz solar possuem mais casos desse tipo de lesão. Esse tipo de lesão apresenta-se como uma área dura, escamosa e pode-se notar pigmentação ao redor da pele, apresentando descolorações amareladas, aprestando danos provocados pela luz solar (Fig. 7a). É frequentemente detectada na cabeça, pescoço, mãos e antebraços (MOY, 2000).

Apesar desta lesão ser benigna (Fig. 5), caso não seja feito os devidos tratamentos tem 20 % de risco de progredir para squamous cell carcinoma e lesões malignas, portanto o tratamento deve ser iniciado o mais breve possível (PATTERSON, 2014).

2.6.2 Basal cell carcinoma

Essa lesão é um tipo de câncer não melanoma (Fig. 7b) que é originado de um crescimento inesperado e não controlado de celular basais. Essas celulares são localizadas na parte de baixo da epiderme, sobre a camada de células basais (SOCIETY, 2017).

Essa é o tipo de câncer de pele mais comum, onde 80 % dos cânceres de pele são

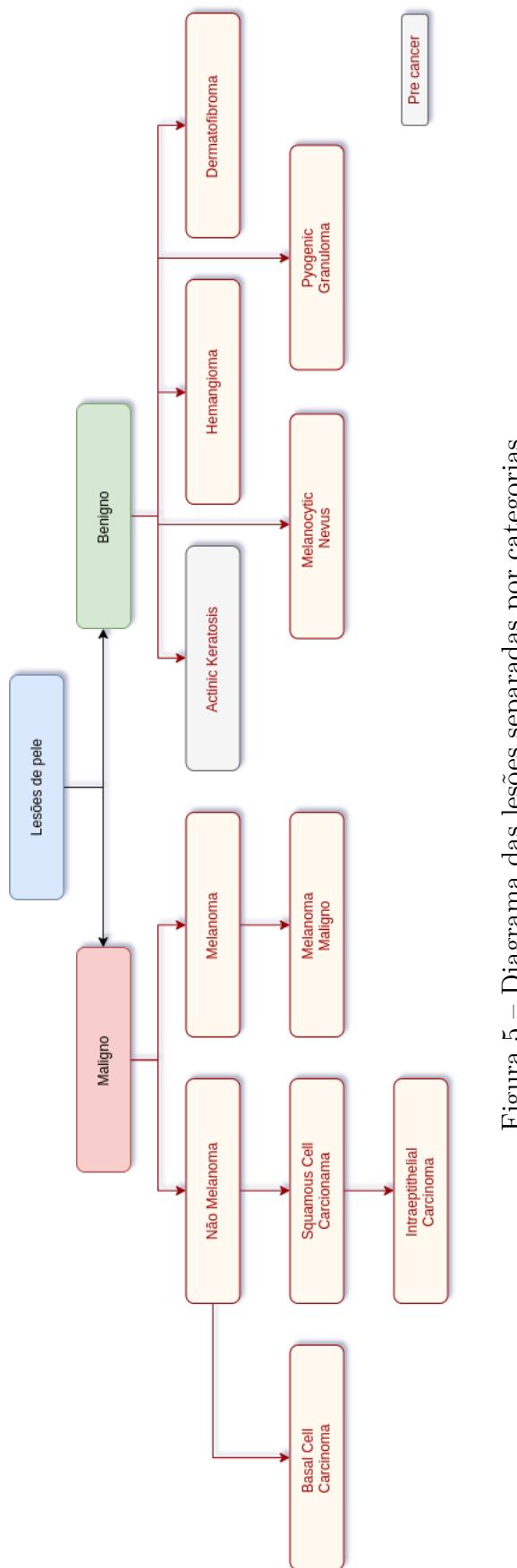


Figura 5 – Diagrama das lesões separadas por categorias
Autor

diagnosticados como sendo basal cell carcinomas. De acordo com a sociedade de câncer americana a cada ano é diagnosticado aproximadamente 4.32 milhões de casos desse tipo de lesão (SOCIETY, 2017).

Caso os tratamento não for efetivo ou apropriado esse tipo de lesão pode ser ocorrer na mesma região de pele (SOCIETY, 2017). Estima-se que no Brasil em 2015 foram aproximante 1958 mortes por canceres não melanoma e 1794 mortes por casos de melanoma (SAUDE, 2017).

- **Melanoma:** Tem origem nos melanócitos (células produtoras de melanina, substância que determina a cor da pele) e é mais frequente em adultos brancos. O melanoma pode aparecer em qualquer parte do corpo, na pele ou mucosas, na forma de manchas, pintas ou sinais. Nos indivíduos de pele negra, ele é mais comum nas áreas claras, como palmas das mãos e plantas dos pés (INCA, 2018).

Embora o câncer de pele seja o mais frequente no Brasil e corresponda a cerca de 30% de todos os tumores malignos registrados no país, o melanoma representa apenas 3% das neoplasias malignas do órgão. É o tipo mais grave, devido à sua alta possibilidade de provocar metástase (disseminação do câncer para outros órgãos)(INCA, 2018).

- **Não melanoma:** É o mais frequente no Brasil e corresponde a cerca de 30% de todos os tumores malignos registrados no país. Apresenta altos percentuais de cura, se for detectado e tratado precocemente. Entre os tumores de pele, é o mais frequente e de menor mortalidade, porém, se não tratado adequadamente pode deixar mutilações bastante expressivas (INCA, 2018).

2.6.3 Dermatofibroma

Classificada como benigno Pode apresentar-se de diferentes cores, mas comumente acastanhado ou bronzeado, com pontos elevados na pele (Fig. 7c). Localizados principalmente nas extremidades do corpo, braços, pernas e antebraços. São assintomáticas e quando apresentam sintoma, o mais comum é dor (SBD, 2018). É descrita como sendo a lesão de pele mais dolorosa (NAVERSEN et al., 1993).

De origem diferenciada, os dermatofibromas são, na verdade, depósitos de fibrinas, ou seja, cicatrizes causadas por pequenos traumatismos, geralmente como picadas de insetos ou espinhos, por isso são típicos de ocorrerem nas regiões nas quais a vestimenta não cobre a pele (SBD, 2018).

2.6.4 Hemangioma

Mais frequente na infância, é a lesão de proliferação cutânea vascular mais comum entre as lesões de interesse nesse trabalho. O hemangioma é decorrente de uma atividade

anormal das células dos vasos sanguíneos, que resulta na formação de um tumor de pele

Esse tumor é formado pelo excesso de vasos sanguíneos ou pela proliferação de pequenas veias dilatadas. Como pode ser visto na Figura 7d, possuem uma cor avermelhada e é perceptível uma grande quantidade de sangue, pode variar de pequenos pontos vermelhos até grandes lesões (ODOM, 2000).

2.6.5 Intraepithelial carcinoma

Foi uma lesão descrita por John T. Bowen in 1912 (BOWEN, 1983), também conhecida pelo nome de doença de Bowen. É uma sub divisão da squamous cell carcinoma com um potencial significativo de progressão lateral. Significa que caso não tratada de forma adequada, existem a possibilidade de progredir para camadas mais profundas da pele. Pessoas com tipos de peles mais sensíveis são mais suscetíveis a esse tipo de lesão como observado por (GUPTA et al., 2009).

Essa lesão é localizada, normalmente, nas camadas mais externas da pele, resultado em manchas avermelhadas, com probabilidade de elevação de pequenas regiões na pele como é observado na Figura 7e.

Frequentemente, observada em pacientes mais velhos, entre 60 anos de idade. O prognóstico é favorável nos estágios iniciais da lesão, porém existe a possibilidade de progressão para lesões mais invasivas como squamous cell carcinoma (MORTON; BIRNIE; EEDY, 2014). Devido o fato de ser uma lesão assintomática, é possível notar uma demora na busca por assistência pelo fato de não causar desconforto nos pacientes. Nos estágios iniciais pode ser confundida com seborrheic keratosis.

2.6.6 Malignant melanoma

É um crescimento anormal das células melanocytes ou células relacionadas com a pigmentação da pele (NCI, 2018c). Essas células tem a finalidade de atribuir colorações características na pele humana. São localizadas na *basement membrane*, na divisão da epiderme com a derme como é mostrado na Figura 6 . É uma lesão de grande risco para os pacientes devido a grande probabilidade de metástase para camadas mais profundas da pele.

Embora ser considerada uma lesão incomum, as taxas de mortes tem crescido nos últimos 30 anos (SOCIETY, 2017). No Brasil estima-se que 6360 novos casos em 2018 (SILVA., 2018b)

A causa pode ter varias contribuições como exposição ao sol até histórico familiar. Essa lesão pode surgir em qualquer parte do corpo, incluindo partes mucosas e apresentam coloração principalmente escura como pode ser visto na Figura 7f.

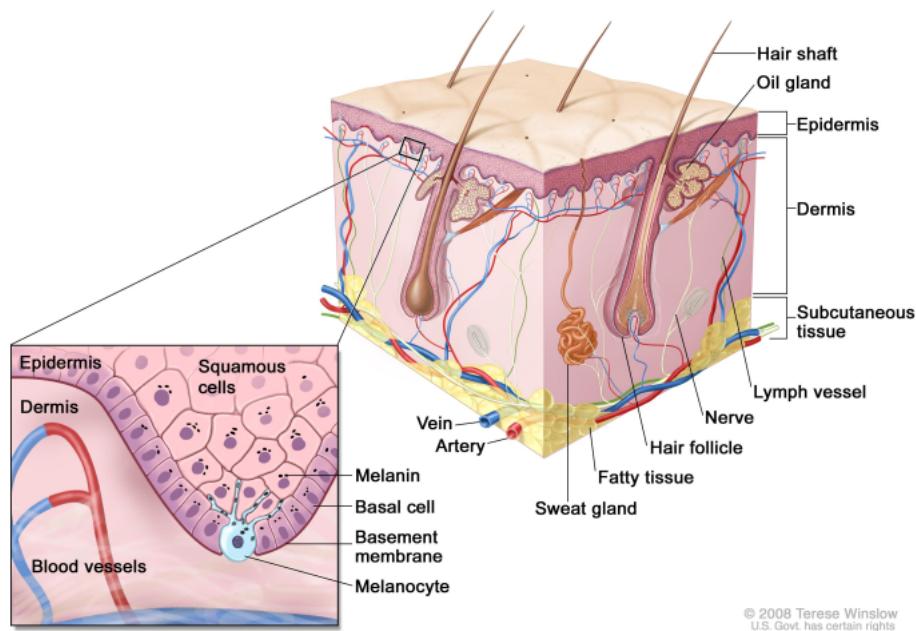


Figura 6 – Representação da pele humana sem lesão
(NCI, 2018c)

2.6.7 Melanocytic nevus

É uma lesão que é observada em todos os mamíferos principalmente em seres humanos (Fig. 7g) , cachorros e cavalos. Lesão benigna nas células que se relacionam com a pigmentação da pele humana. Pessoas normais podem possuir entre 10 a 40 espalhados pelo corpo. São formados no começo da infância até a velhice. Normalmente, por volta dos 40 anos de idade é possível notar que a mancha tente a desvanecer (NCI, 2018b).

2.6.8 Pyogenic Granuloma

Tumor benigno de lesão vascular relativamente comum na pele na mucosa, porém sua causa é desconhecida (MILLS; COOPER; FECHNER, 1980). Essa lesão não é uma forma de *pyogenic*, que forma pus, e nem *granuloma*, lesão inflamatória. O maior problema com essa patologia é propensão de sangramentos e criação de úlceras.

Comumente, localizadas na cabeça e pescoço, como pode ser visto na Figura 7h a lesão apresenta uma bolsa de sangue brilhante. Possui uma evolução rápida com o passar das primeiras semanas, mas normalmente não apresenta riscos como outras lesões e se não tratada tende a secar e regredir vagamente.

2.6.9 Squamous cell carcinoma

Esse câncer desenvolve-se do *neoplasma* na celulas squamous, que é perto da parte mais externa da epiderme, como mostrado na Figura 6. É possível notar uma pequena

êucera avermelhada como na Figura 7i. Normalmente esta é encontrada nas partes do corpo que são expostas ao sol, principalmente na cabeça e mãos ([SOCIETY, 2017](#)).

Normalmente é recomendado para olhar sinais em pacientes que possuem histórico de actinic keratosis. Caso o paciente tenha múltiplos casos de keratosis é um indicativo que possivelmente o câncer irá se desenvolver ([JY RAMSEY ML. CANCER, 2019](#)).

2.7 Natureza da base de dados

A natureza das imagens medicas em todas as áreas propõem diversos desafios, com imagens de lesões não são diferentes. A seguir será listado alguns problemas observados nas base de dados.

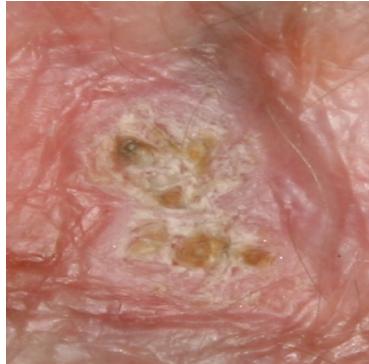
2.7.1 Dificuldades

O campo de imagens medicas trás desafios enraizados na natureza das imagens. Por essa razão, é necessário avaliar para traçar as melhores estrategias para trabalhar com essas imagens. Outra desafio é a diferença entre as bases de dados usado no presente trabalho, isso adiciona mais variabilidade para a base e pode acarretar em modelos que não conseguem convergir, não importa o quanto complexo é o modelo, pois a natureza da base é tão diversa que nem especialistas podem classificar as imagens.

A base do Edinburgh é a mais padronizada, organizada de todas, é possível notar que as imagens foram retiradas em meio a condições de iluminação, resolução e distância semelhantes. A base do MED-NODE foram retiradas em condições padronizadas mas quando comparadas com as imagens da base Edinburgh é possível notar algumas diferenças de iluminação e qualidade das imagens. A base atlas é a base que adiciona mais variabilidade pois são imagens coletadas em diferentes condições de iluminação, resolução e distância, essa base foi adicionada para suplementar as demais. A base do ISIC trás uma proposta padronizada e coletadas em condições parecidas, porém adiciona mais variabilidade pelo fato das imagens serem coletadas por dermatoscópio o que apresenta a pele de forma diferente das imagens das outras bases, um exemplo pode ser visto na Figura 8.

- **Etnia:** As diversas etnia na área de lesões de pele é importante pois os síntomas e as formas de manifestação das lesões varia de etnias, pois geralmente existem tons de pele diferentes para cada etnia, essa questão foi exposta no trabalho do ([HAN et al., 2018](#)).

Portanto, para se obter uma boa generalização do problema é necessário balancear a base de dados com imagens de diferentes etnias com o mesmo tipo de lesão. Essa falta de diversidade pode gerar métricas ruins para testar o modelo em novas imagens de diferentes etnias.



(a) Actinic Keratosis



(b) Basal Cell Carcinoma



(c) Dermatofibroma



(d) Hemangioma



(e) Intrapithelial Carcinoma



(f) Malignant Melanoma



(g) Melanocytic nevus



(h) Pyogenic Granuloma



(i) Squamous Carcinoma

Figura 7 – Lesões de interesse neste trabalho

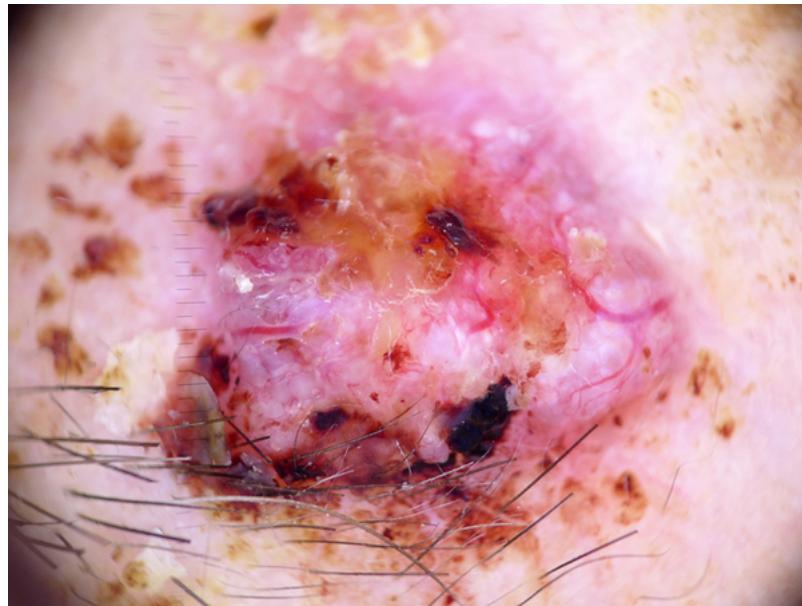


Figura 8 – Imagem da base do ISIC da Basal cell carcinoma
Base do ISIC

- **Idade:** A idade pode influenciar na aparência que a lesão se apresenta. Pelos trabalho do (HAN et al., 2018) chegou-se a conclusão que a maioria das imagens da bases eram de pessoas mais velhas, o que resultou em baixas métricas para pessoas mais jovens com as mesmas lesões.
- **Diferentes equipamentos para coleta das imagens:** Isso adiciona mais variabilidade nas imagens, pois duas câmeras diferentes podem trazer apresentar diferentes pontos de vista da mesma lesão, o que leva a problemas de classificação pelo modelo.
- **Posição:** Esse problema esta atrelado principalmente a habilidade para coletar a imagem. Por outro lado, a localização da lesão obriga a capturar outras partes do corpo, como por exemplos unhas. Além disso, algumas fotos capturaram outras partes do corpo que não deveriam aparecer isso trás dificuldades para o modelo generalizar o problema.
- **Cabelos:** Muitas imagens possuem cabelos que cobrem a lesão. Isso pode ser um problema para a generalização do problema, pois o cabelo adiciona um *ruido*. É comum observar em imagens que a localização da lesão é na cabeça do paciente.
- **Tipos de pele:** As características intrínsecas de cada lesão de pele sob diferentes peles. Os desafios observados pela elasticidade e a reflexividade. A elasticidade significa as formas de distorções sobre a lesão. Outro fato que pode ser levado em consideração é a mudança de elasticidade da pele humana a medida que o este envelhece (CUA; WILHELM; MAIBACH, 1990). A reflexividade é a propriedade da pele de refletir raios focais. Dependendo da parte do corpo a pele pode ter diferentes propriedades de reflexividade (DENGEL et al., 2015).

2.7.2 Amostra de dados

Uma base ideal tem a mesma quantidade amostras para todas as classes e dentro das classes existe diversas formas da mesma lesão assim o modelo é balanceado e fiel a realidade. Contudo, a base final não possui o mesmo numero de imagens, algumas classes tem menos amostras isso pode ser observado nas tabelas da Seção 2.5. Portanto as classes com menos imagens estão sujeitas a possíveis *underfitting* pois o modelo não terá muitos exemplos do que aprender.

2.7.3 Rótulos

As bases consolidados são mais confiáveis quanto ao processo de rotulagem das imagens. A base de dados do Edinburgh foi rotulada por especialistas. O MED-NODE não divulgou como foi o processo de rotulagem dos dados e do Atlas varia de website para website. Porem, possivelmente nem uma das imagens foi rotulada baseada em resultados de biopsias, o que seria uma informação extremamente importante pois garantira realmente que os rótulos estão corretos.

2.8 Preparação das amostras

Para o treinamento de modelos de redes neurais artificiais é necessário uma grande quantidade de amostras, contudo devido a falta de amostras de imagens medicas foi necessário a utilização de algumas técnicas para contornar esse problema.

Para essa finalidade deve-se fazer transformações não invasivas nas imagens original com o intuito de replicar-las de forma a preservar os principais padrões encontrados na imagem original.

Outro problema encontrado é a criação um novo modelo convolucional para ser usado com a base. Para abordar esse problema, foi utilizado uma técnica que parte do principio que existe um modelo que performa muito bem para problemas de classificação e que a partir desse modelo pode-se construir novos conceitos.

Para abordar o problema da falta de imagens foi utilizado técnicas de *Data Augmentation* e para o problema da criação de um novo modelo foi utilizado técnicas de *Transfer Learning*.

2.8.1 Transfer Learning

Como é observado o problema de ter poucas amostras é recorrente tanto na industria quanto nas pesquisas científicas. Isso impõe um grande obstaculo para o treinamento de redes neurais convolucionais, por conta das amostras disponíveis não representar de

forma efetiva o comportamento observado no mundo. Por esse razão é comum a utilização de pesos pre treinados em topologias treinadas, para tanto existe duas abordagens:

- **Fixed feature extractor:** Consiste em usar CNN com extratores de características fixos e então ajustar os parâmetros livres. Comumente essa estratégia é mais rápido pois existe menos parâmetros para serem atualizados, porem se a natureza das amostras forem muito diferentes das amostras que foram usados para treino o modelo terá muitas dificuldades de convergir.
- **Continuar o treino mudando a camada final:** Essa estratégia utiliza o modelo pre treinado, porem é alterado a ultima camada da rede, geralmente a camada *fully-connected*, para o numero de classes do nova base de dados e então atualiza os parâmetros via *backpropagation*. Normalmente é mais demorado porem resulta em melhores métricas de teste.

Para o presente trabalho foi usado a arquitetura da ResNet (Sec. 2.4.1) pre treinada com a base de dados do ImageNet. Um ponto a ser observado é que dependendo da natureza das amostras deve-se considerar uma taxa de aprendizado da rede baixa, pois partindo principio que os pesos são bons na classificação e se possível deve-se evitar distorcer-los de forma abrupta (YOSINSKI et al., 2014).

2.8.2 Data augmentation

Essa técnica é usada quando não possui uma quantidade infinita de amostras para treinar o modelo. Isso pode ser feito criando dados sintéticos que estão relacionado com as amostras originais da base de dados através de aplicação de transformações. Em classificação de imagens isso é feito aplicando rotação, cor tanto, alterando a luminosidade e qualquer outra transformação que não seja invasiva e degrade a natureza da imagem original. Essas perturbações adiciona mais variabilidade como entrada, portanto is pode levar a uma redução na probabilidade de *overfitting* (Krizhevsky; Sutskever; Hinton, 2012; Perez; Wang, 2017; Cubuk et al., 2018).

Para o presente trabalho foi adicionado aleatoriamente transformações de luminosidade para imitador o comportamento de diferentes luminosidades impostas por diferentes ambientes de coletas.

2.8.2.1 Metodos de Augmentation

Existe uma biblioteca no *python* que auxilia essa tarefa (BLOICE; STOCKER; HOLZINGER, 2017). A biblioteca possui transformações básicas pre definidas como rotação, translação e etc, e fornece as ferramentas necessárias para criação de novas trans-

formações que foram utilizadas para a implementação da variação de luminosidade nas imagens.

2.8.2.1.1 Transformações

Cada escolha de transformação aplicada é baseada no guia de data augmentation (PEREZ; WANG, 2017; CUBUK et al., 2018) ou na natureza das amostras. As transformações foram organizadas em forma de pipeline para que cada transformação tenha uma probabilidade de ser aplicada na imagem e então salvar a amostra no destino especificado. A Tabela 5 mostra as transformações as probabilidade do pipeline.

Tabela 5 – Transformações aplicadas no processo de *data augmentation*

| Transformações | Probabilidades |
|---------------------------|----------------|
| Rotação | 0.5 |
| Zoom | 0.4 |
| Flip Horizontal | 0.7 |
| Flip Vertical | 0.5 |
| Distorção | 0.8 |
| Variância na Luminosidade | 0.5 |

2.9 Preparação da base de dados

A preparação da base de dados é um dos passos mais importantes para qualquer projeto que envolve machine learning. Esse passo leva em consideração principalmente o modo que é dividido a base de dados em três partes treino, validação e teste.

Como observado na Figura 9 foram agregados quatro bases de dados diferentes em uma única base. E foi feita uma verificação nas imagens para descartar qualquer amostra que não parecia uma imagem de lesão. Isso foi necessário pois como visto na seção 2.5.4 foi utilizado *scripts* de *web scraping* para formar a base Atlas e durante o processo algumas imagens de esquemáticos e diagramas relacionados a lesão foram baixados na base. Formando uma base de amostras apresentadas na Tabela 6.

Em seguida foi feito um pre processamento nas imagens para converter todas para o mesmo formato, renomear de forma crescente e redimensionar as imagens pois nem todas eram do mesmo tamanho, foi escolhido um tamanho de 448x448 que é o dobro da dimensão das imagens usadas no modelo. Um cuidado especial foi feito nessa etapa, pois para não distorcer as imagens que não são quadradas foi feito um complemento de pixels tanto horizontal quanto vertical em imagens de diferentes dimensões o código fonte para essa operação pode ser encontrado no Apêndice B.

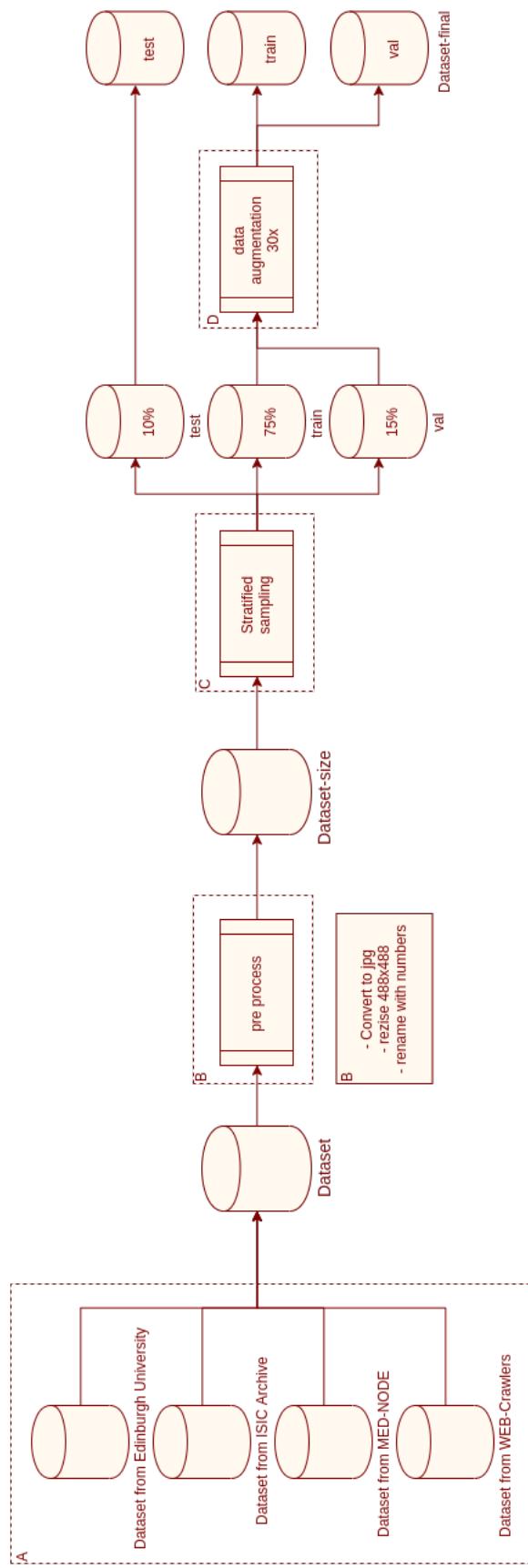


Figura 9 – Diagrama de processos utilizados para a preparação da base de dados
Autor

Tabela 6 – Numero de amostras da base agregada

| Tipo de lesão | Amostras |
|---------------------------|-----------------|
| Actinic Keratosis | 185 |
| Basal Cell Carcinoma | 832 |
| Melanocytic Nevus | 502 |
| Seborrhoeic Keratosis | 107 |
| Squamous Cell Carcinoma | 417 |
| Intraepithelial Carcinoma | 148 |
| Pyogenic Granuloma | 98 |
| Haemangioma | 173 |
| Dermatofibroma | 206 |
| Malignant Melanoma | 687 |
| Total | 3355 |

Em seguida como foi feito a separação da base de dados em três sub bases, para treinamento, validação e teste. Nessa etapa foi utilizado uma biblioteca do *python* onde esta implementado a separação de imagens de forma estratificada para não misturar amostras entre as sub bases. A proporção escolhida da seguinte forma 75%, 15% e 10%, para respectivamente, treinamento, validação e teste. Essa biblioteca implementa um parâmetro de *seed*, caso queria reproduzir esse mesmo experimento com a base de dados agregada basta apenas usar o mesmo código para o *seed* que a subdivisão será exatamente igual ao usado no presente trabalho. Esse *script* esta apresentando no Apêndice C

Com as bases separadas foi aplicada a augmentation na base de treinamento e validação usando as transformações citadas na seção 2.8.2.1.1 com um fator de multiplicação de 30x. Na Tabela 7 fica resumido o numero de amostras utilizadas em cada etapa dos experimentos e o condigo fonte pode ser encontrado no Apêndice D.

Tabela 7 – Numero de amostras finais

| Tipo de lesão | treino | validação | teste |
|---------------------------|---------------|------------------|--------------|
| Actinic Keratosis | 4278 | 837 | 20 |
| Basal Cell Carcinoma | 18720 | 3844 | 84 |
| Melanocytic Nevus | 11656 | 2325 | 51 |
| Squamous Cell Carcinoma | 9672 | 1922 | 43 |
| Intraepithelial Carcinoma | 3441 | 682 | 15 |
| Pyogenic Granuloma | 2263 | 434 | 11 |
| Haemangioma | 3999 | 775 | 19 |
| Dermatofibroma | 4774 | 930 | 22 |
| Malignant Melanoma | 15965 | 3193 | 69 |
| Total | 74768 | 14942 | 334 |

Em seguida com essa base pronta foi iniciado os experimento de ajuste de parâmetros livres.

3 Resultados prévios

Esse espaço esta reservado discussão dos resultados prévios obtidos através das técnicas e referencias teóricas apresentada nas seções anteriores.

3.1 Infraestrutura

Todos os experimentos quem envolveram treinamento, validação e teste foram realizados na plataforma *open source* da Google que é uma variação do serviço *datalab* que fornece uma infraestrutura que atende as necessidades de processamento do presente trabalho.

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality { }

incarnation: 10575003407439260273,
name: "/"
device:XLA_CPU:0"
device_type: "XLA_CPU"
memory_limit: 17179869184
locality { }

incarnation: 12607010922076994446
physical_device_desc: "
device: XLA_CPU device",
name: "/"
device:XLA_GPU:0"
device_type: "XLA_GPU"
memory_limit: 17179869184
locality { }

incarnation: 15083069251257245813
physical_device_desc: "
device: XLA_GPU device",
name: "/"
device:GPU:0"
device_type: "GPU"
memory_limit: 15956161332
locality { bus_id: 1 links { } }
incarnation: 7574273019028501897
physical_device_desc: "
device: 0,
name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0"]
```

Figura 10 – Detalhes tecnicos da infraestrutura
Autor

A Figura 10 mostra os detalhes técnicos do ambiente onde foram conduzidos os experimentos. Com Linux version 4.14.137+ (chrome-bot@chromeos-legacy-release-us-central1-b-x32-44-v3dn) (gcc version 4.9.x 20150123 (prerelease) rodando o framework PyTorch - 1.3.1 e TourcehVision - 0.4.2.

3.2 Classificação

Existem muitas topologias para aplicar a técnica de *transfer learning*, porém como mencionado em seções anteriores a escolha foi a ResNet-152 treinada usando a base de dados processada mencionada na seção 2.9.

Então foi modificada a ultima camada da topologia mencionada de 1000 classes para 9 classes, correspondendo ao numero de lesões de interesse.

3.2.1 Processo de treinamento

Foram conduzido 8 experimentos com diferentes combinações de parâmetros livres com o objetivo de conseguir o melhor modelo possível. Cada experimento teve um tempo de duração de aproximadamente 12 horas ininterruptas. Durante o processo de treinamento foram utilizados as métricas mencionadas na seção 2.3 para verificar possíveis *overfitting* e *underfitting* durante o treinamento do modelo.

3.2.2 Parâmetros livres

Como foi utilizado um *framework* diferente e bases diferentes das referencias como ([HAN et al., 2018](#)). O processo de escolha dos parâmetros livres foi na tentativa e erro.

Portanto como pode ser visto no Apêndice E os parâmetros livres foram definidos da seguinte forma:

- **batch size:** Esse parâmetro livre define o numero de amostras que será salva em memoria para ser processada na passagem pela rede neural. O maior limite encontrado para esse parâmetro é a limitação de memoria na GPU, para esse trabalho no melhor experimento foi definido como 32 no momento do treino e 6 nos testes.
- **step size:** Refece-se a frequência com que a taxa de aprendizagem vai cair durante as *epoch* a partir de outro fator chamado de *weight_decay*. Como cada epoch demorava bastante para completar esse parâmetro foi definido como 1.
- **numero de epoch:** Cada *epoch* defina uma passagem e *backpropagation* completa da base de dados no modelo. Esse parâmetro foi definido como sendo um valor alto mas que nunca seria alcançado pois cada experimento durava 12 horas. Em média foram 12 - 20 *epoch* para cada experimento.
- **taxa de aprendizagem:** O *learning rate* é o parâmetro livre que controla o quanto os pesos são atualizados com relação a função de perda. Foi definido como 0.01.
- **weight_decay:** Define a intensidade que a taxa de aprendizagem vai cair durante cada *step size*. Foi definido sendo 0.00001.

Os outros parâmetros livres foram definidos como sendo $momentum = 0.9$ e $gamma = 0.1$. Foi escolhido uma taxa de aprendizagem alta comparado com os trabalhos correlatos devido a dois fatores. Nos experimentos foi observado que as métricas atingiam um plato muito cedo, mostrando que o modelo não tinha poder para aprender as características da lesões. Aumentar a taxa de aprendizado por levar a redução de *underfitting* (SMITH, 2018).

3.2.3 Resultados

Foi salvo o modelo com as melhores métricas durante o treinamento e a validação. Com o modelo salvo, foi criado um *script* para aplicar a base de dados de teste para então avaliar o quanto bom o modelo está performando para amostras completamente novas para o modelo.

Foi avaliado todas as métricas apresentadas na Seção 2.3. Contudo, a acurácia total do modelo foi de 78.44% para a base de teste. Contudo, a acurácia não é a única métrica para avaliar o modelo, pois possuem o viés da base não estando balanceada o *script* que gera essas avaliações está no Apêndice F.

Com a matriz de confusão é possível notar a dificuldade do modelo em predizer algumas classes. Como pode ser visto na Figura 18 no Apêndice G, a lesão Basal Cell Carcinoma é confundida com Squamous Cell Carcinoma, bem como a dificuldade de predizer Intraepithelial Cell Carcinoma com relação ao Basal Cell Carcinoma. Essas previsões são de se esperar visto a natureza maligna da lesão e a semelhança superficial.

Além disso, foi calculado um relatório de classificação que fornece o recall, precisão e F1 score para as classes individuais assim como as médias. Pode-se avaliar esses valores na Tabela 10 no Apêndice G.

Por último foi plotado a ROC curve para cada lesão e sua respectiva UAC as curvas podem ser vistas no Apêndice G. Essa métrica é uma das mais populares para a avaliação de modelos de machine learning a Tabela 8 mostra uma comparação entre diferentes trabalhos relacionados.

Tabela 8 – Comparativo entre trabalhos correlatos as AUC

| Tipo de lesão | (ESTEVA et al., 2017) | (HAN et al., 2018) | (MENDES; SILVA, 2018) | Atual |
|---------------------------|-----------------------|--------------------|-----------------------|-------|
| Actinic Keratosis | - | 0.83 | 0.96 | 0.94 |
| Basal Cell Carcinoma | - | 0.90 | 0.91 | 0.95 |
| Melanocytic Nevus | - | 0.94 | 0.95 | 0.98 |
| Squamous Cell Carcinoma | - | 0.91 | 0.95 | 0.84 |
| Intraepithelial Carcinoma | - | 0.83 | 0.99 | 0.93 |
| Pyogenic Granuloma | - | 0.97 | 0.99 | 0.99 |
| Haemangioma | - | 0.83 | 0.99 | 0.92 |
| Dermatofibroma | - | 0.90 | 0.90 | 0.94 |
| Malignant Melanoma | 0.96 | 0.88 | 0.96 | 0.96 |

4 Próximos passos

As seções anteriores refere-se a construção de um modelo que irá ser utilizado para a aplicação de conceitos e técnicas de *Explainable artificial intelligence*. Na sigla XAI que refere-se a técnicas aplicadas a modelos de inteligencia artificial com o objetivo de tornar-los comprehensíveis do ponto de vista das predições. Essa seção esta reservada para discutir os próximos passos de implementação e pesquisa deste trabalho.

4.1 Interpretabilidade e Explicabilidade

Atualmente, modelos de inteligência artificial tem a fama de serem "*Caixas Pretas*", devido a complexidade dos modelos tem-se pouca visibilidade em como as decisões são feitas. Ademas, a necessidade de explicações surge, com mais notoriedade, em sistemas de IA que são utilizados em ambientes sensíveis e que afetam diferente as pessoas, como nas áreas financeiras, educacionais, contratações de trabalho e campos médicos ([CARUANA et al., 2015](#)). A habilidade de explicar determinadas decisões é um aspecto desejado em softwares de decisão-assistido ([TEACH; SHORTLIFFE, 1981](#)).

Visibilidade e transparência do modo que os modelos predizem são necessários para verificar possíveis anomalias, que pode no final das contas auxiliar os designers do modelo a verificar o que esta acontecendo de errado.

Com o crescimento do poder computacional e das teóricas acerca de redes neurais profundas a complexidade cresce na mesma proporção e a explicabilidade das decisões tornam-se cada vez mais difíceis.

Ainda nessa questão existem níveis de necessidade de explicação. Existe menos impacto em decisões como distinguir cachorro quente ou não em relação a diagnosticar um tipo de câncer de um paciente que terá que fazer uma cirurgia de emergência. Um fator importante é fornecer explicações durante o processo de predição, pois será possível revisar as predições feitas e verificar o que ocorreu de errado ou o que contribuiu para fornecer a predição correta.

A busca das respostas do "*Por quê ?*" não é recente, pela literatura existem pesquisas datada da década de 80 ([CLANCEY; SHORTLIFFE, 1984](#); [CLANCEY, 1981](#); [CHANDRASEKARAN; TANNER; JOSEPHSON, 1989](#)). A ideia é basicamente a mesma dos dias atuais, sistemas que lidam com decisões sensíveis devem ser capaz de responder o "*Por quê ?*".

4.1.1 Conceitos

O trabalho de (BIRAN; COTTON, 2017) define que um sistema é interpretável se o ser humano pode entender suas operações, tanto por inspeção ou uma explicação produzida pelo modelo. Nesse sentido as explicações produzidas pode não ser consoante com o estilo de decisões humanas. Eles definem justificativa como a explicação pela qual uma decisão é boa ou ruim sem informar sobre o processo pelo qual a decisão foi feita (BIRAN; COTTON, 2017).

XAI se refere ao conceito que um agente é responsável para explicar o processo de explicação ou a decisão feita por outro agente. Esse área envolve diversos campos de conhecimento como filosofia, psicologia, ciências cognitivas e áreas de interação entre homem e computador. Cada campo contribui com os métodos para definir como as explicações devem ser abordadas. Na Figura 11 refente ao trabalho do (MILLER, 2017) fornece um escopo de definição dos campos para XAI.

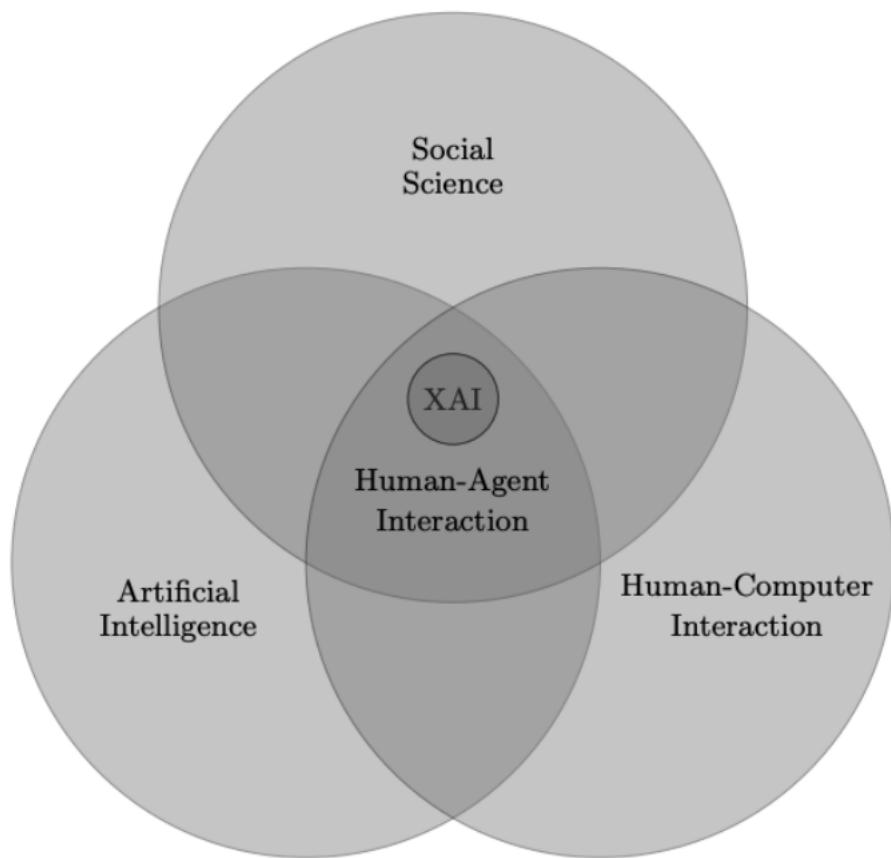


Figura 11 – Escopo da área XAI
(MILLER, 2017)

4.2 Métodos de interpretabilidade

Visto uma breve introdução do escopo do trabalho nessa seção será apresentado alguns métodos que serão implementados no modelo criado para o presente trabalho e fazer as avaliações e julgamentos pertinentes.

Interpretabilidade divide-se em dois níveis, global e local. As implementações futuras estarão mais focadas em métodos de avaliação locais.

Dentro das avaliações locais pode ser verificado grupos de precisões únicas e previsões em grupos. As previsões únicas levam em consideração uma única entrada e explica quais foram os fatores que levaram o modelo a determinada decisão baseado na entrada. Espera-se um comportamento linear nessas previsões pois para entradas de mesma classes espera-se explicações parecidas para as respectivas decisões.

Avaliações globais são mais complexas pelo fato de ser necessário conhecer o modelo como um todo de uma única vez (LIPTON, 2016). Esse conhecimento está relacionado com o modo que o modelo foi treinado, os pesos, parâmetros livres, características e estruturas.

4.2.1 Método Model-agnostic

É um método separado do modelo. Sua grande vantagem é a flexibilidade pois os desenvolvedores podem usar qualquer modelo de machine learning que o presente método pode ser aplicado. Qualquer recurso que foi construído para a interpretação do modelo e não está depende diretamente do modelo como gráficos e interfaces do usuário são representações desse modelo.

Aspectos desejáveis desse sistema de explicação (RIBEIRO; SINGH; GUESTRIN, 2016). (2016a).

- **Flexibilidade do modelo:** O método de interpretação pode funcionar com diversos modelos, desde *random forest* até redes neurais profundas.
- **Flexibilidade da explicação:** Não possui limitação quanto ao modo de explicação. Tem situações que é necessário uma formula linear outros casos gráficos com características importantes.
- **Flexibilidade da representação:** O sistema de explicação deve ser capaz de usar diferentes características de representação em quanto o modelo está sendo explicado.

Pode-se representar o caminho geral que a informa até alcançar os humanos. Na camada mais baixa, a informação é capturada do mundo. Pode ser qualquer coisa que

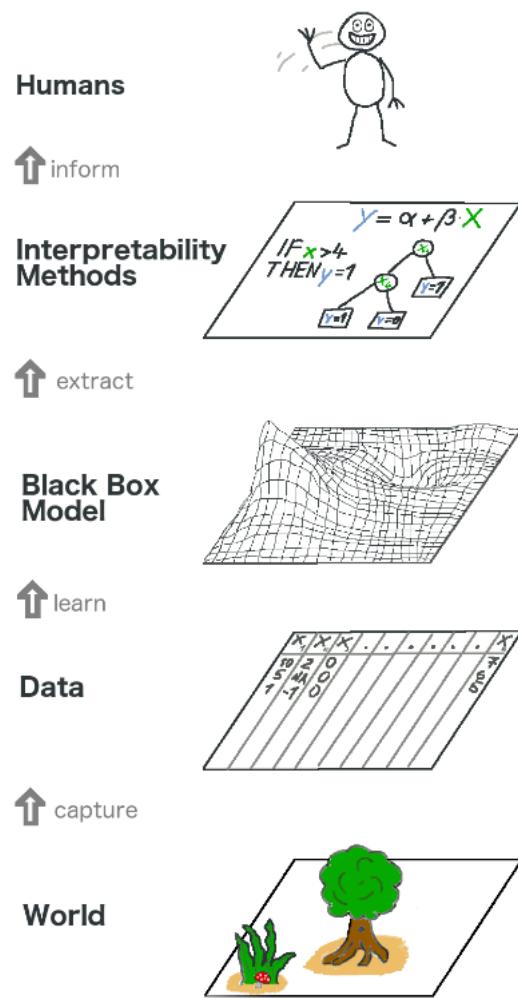


Figura 12 – O panorama geral de interpretabilidade para modelos de machine learning ([MOLNAR, 2019](#))

existe algum tipo de interação. Essa informação é transportada para computadores e então processada por modelos que não é possível entender completamente. O modelo vai abstrair os dados e aprender os padrões. A partir desse modelo pode-se aplicar métodos de interpretabilidade para entender melhor o modelo. Então finalmente essas explicações podem ser processadas e representadas em diferentes formatos para utilização dos humanos. Esse processo é representado por meio da Figura 12

Claro que nesse caminho pode existir diversas variações e mudanças no que esta representado. Os dados podem vir de simulações, os resultados do modelo alimentar outros caminhos e assim por diante.

O modelo de explicação agnostic pode se subdividir em duas categorias, métodos baseados em abordagens pelo gradiente, esses métodos são mais focados para redes neurais, a segunda abordagem usa *input-perturbations* para gerar explicações ([ROBNIK-SIKONJA; BOHANEC, 2018](#)).

Brevemente, a abordagem baseadas em nos gradientes usa o calculo dos gradientes de saída do neurônio em relação a entrada. Por outro lado, como o nome sugere, a abordagem baseada em *input-perturbations* usa entradas com uma pequena perturbação para testar se a decisão final do modelo muda em relação a entrada sem perturbação.

4.2.1.1 Perturbation-based

Como mencionado brevemente, a abordagem adiciona uma perturbação na entrada e avalia as consequências dessa perturbação na decisão final do modelo. Essa perturbação consiste na remoção de pequenas porções específicas de informação da entrada aplicando ruído. Comparado com a abordagem baseada na computação dos gradientes esse método é computacionalmente mais custoso.

Outro problema enfrentado nessa abordagem é a dificuldade de escolha da perturbação que era aplicada na amostra. Pois a intenção não é mudar a natureza da amostra portanto a remoção de pedaços da imagem é feito apenas substituindo regiões de interesse por pixels cinza. Porem a coloração do pixel pode ser um problema dependendo do classificador, para classes com coloração predominantemente cinza o modelo pode predizer erroneamente com grande confiança.

Quando a adição da perturbação é muito baixa, existe a possibilidade do modelo não interpretar. Para evitar esse problema esse método é utilizado em grandes regiões de imagens, em detrimento de torna-se menos preciso. ([FONG; VEDALDI, 2017](#)).

4.2.1.2 LIME

É a abri viação de (*Local Interpretable Model-agnostic Explanations*) é uma metodologia apresentada por ([RIBEIRO; SINGH; GUESTRIN, 2016](#))que implementa um modelo local que explica predições individuais. LIME usa o conceito de perturbação apresentado na subseção anterior para treinar um modelo interpretável. Esse modelo treinado deve possuir boa interpretabilidade local.

$$\text{explicação}(x) = \underset{g \in G}{\operatorname{argmin}} L(f, g, \pi x) + \Omega(g) \quad (4.1)$$

LIME pode se expresso matematicamente pela Equação 4.1. onde x é a instancia que será explicada, g é o modelo interpretável (e.g. Regressão Linear), L é a função de perda (e.g. erro quadrático médio), que computa o quão perto a explicação esta da predição do modelo original f , G é a família de possíveis explicações e a complexidade do modelo é representada por $\Omega(g)$.

Para as imagens faz muito mais sentido perturbar grupos de pixels já que mais de um pixel contribui para a determinação das características de uma classe. A Figura

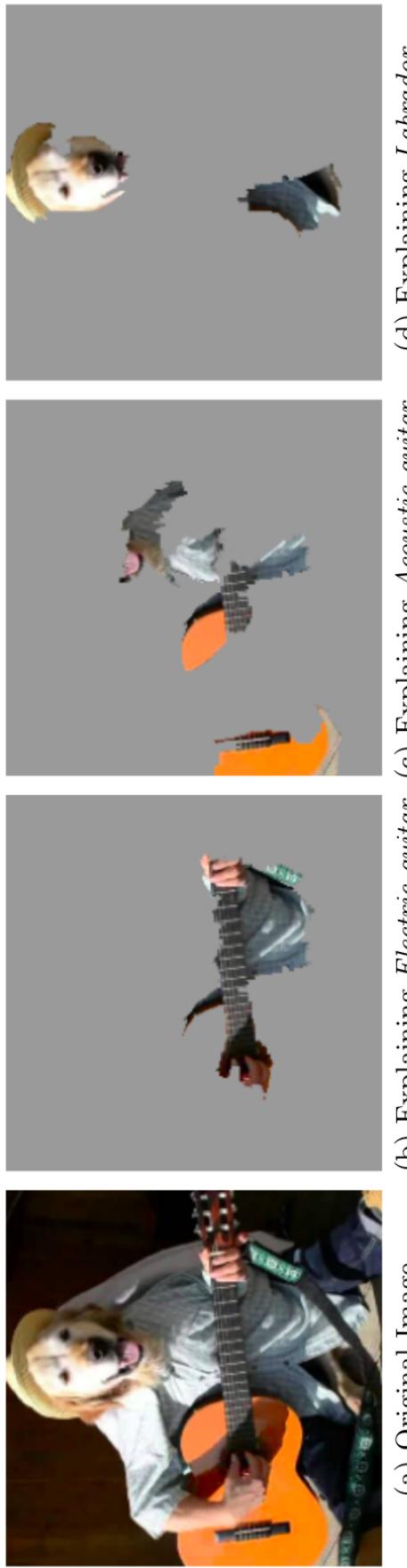


Figura 13 – Explicação usando LIME para as top 3 classes na classificação de uma imagem feita pelo modelo da Google Inception.
(RIBEIRO; SINGH; GUESTRIN, 2016)

13 mostra a segmentação de pixels conhecidas como *superpixels* que são utilizados para explicar o modelo.

4.2.1.3 Gradient-based

É a abordagem mais popular dos métodos de explicação local para classificação de imagens (ERHAN et al., 2009; SMILKOV et al., 2017; SUNDARARAJAN; TALY; YAN, 2017). Essa metodologia é baseada na importância de ativação de cada pixel da imagem de entrada em relação a predição de saída. Esses métodos são bons para explicar amostras únicas porem não performa tão bom para obter entendimento geral da classe observada.

Existem varias sub técnicas que serão abordadas nesse trabalho as Figuras 14 e 15 mostra os resultados visuais da aplicação dessas técnicas e algumas das variações possíveis.

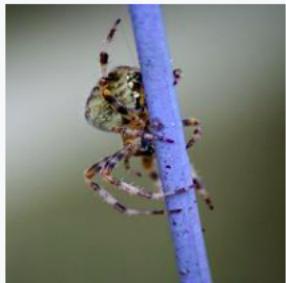
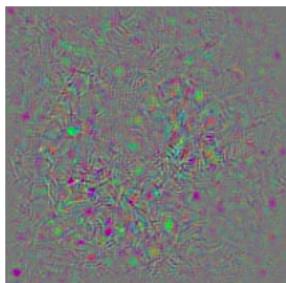
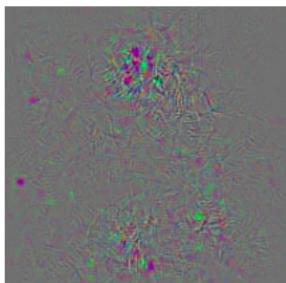
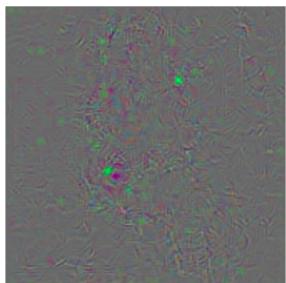
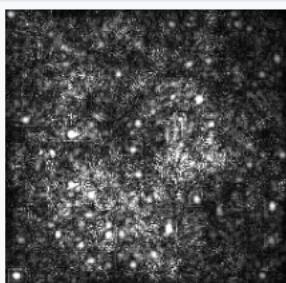
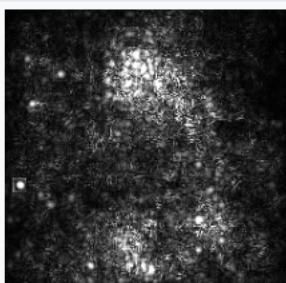
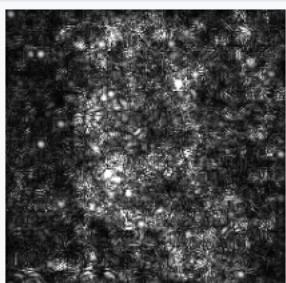
| | Target class: King Snake (56) | Target class: Mastiff (243) | Target class: Spider (72) |
|----------------------------------|---|--|---|
| Original Image |  |  |  |
| Colored Vanilla Backpropagation |  |  |  |
| Vanilla Backpropagation Saliency |  |  |  |

Figura 14 – Exemplos de visualização do gradiente das imagens (OZBULAK, 2019)

Como os filtros que são utilizados em cada camada convolucional tem um papel de extrair as características das amostras de entrada aplicar técnicas para visualização dos filtros utilizado pode ajudar na interpretabilidade da rede neural. Como a organização dos filtros numa rede neural tende a ser hierárquico, ou seja, filtros mais simples tentem

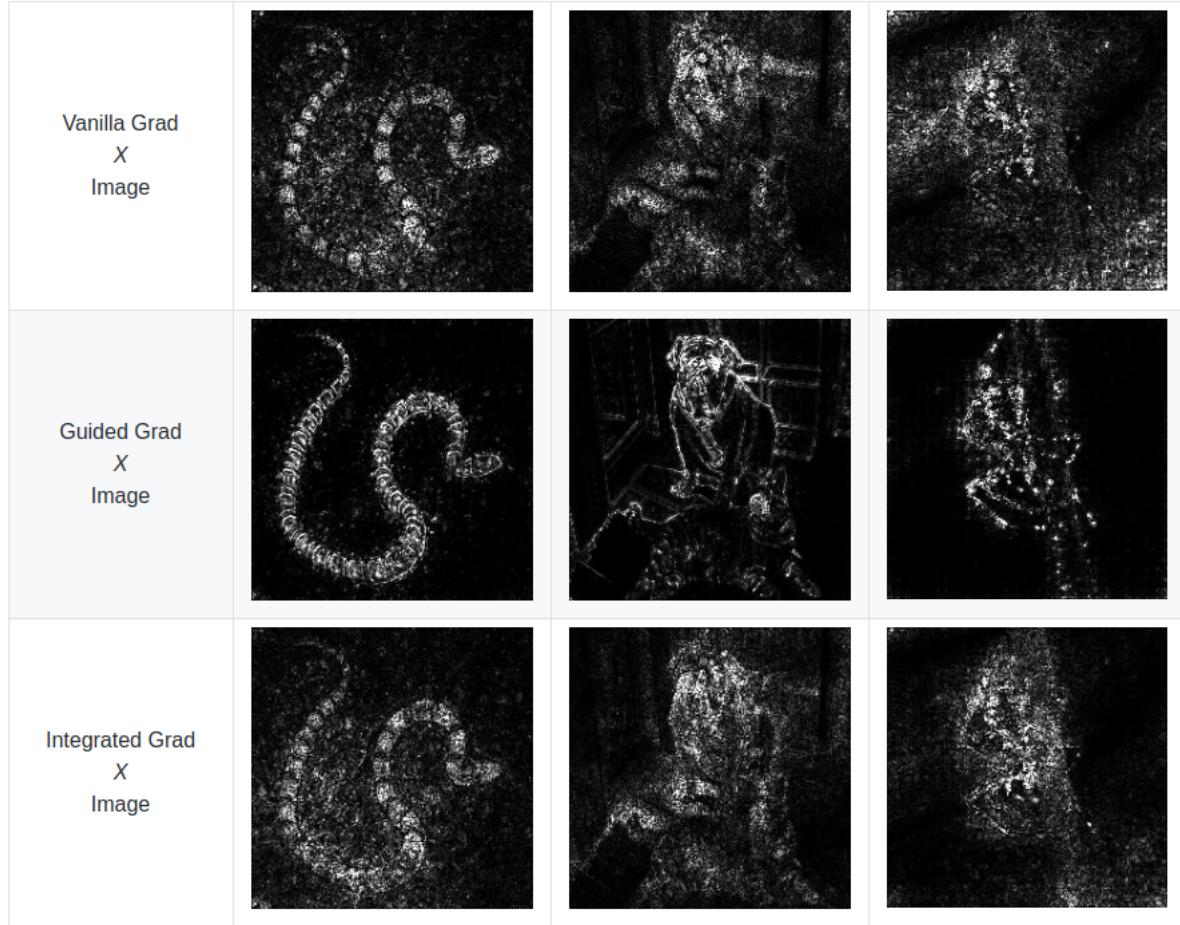


Figura 15 – Multiplicação da imagem de entrada com o gradiente de ativação respetivo ([OZBULAK, 2019](#))

a esta no começo da arquitetura pois são responsáveis por captar características gerais das imagens, no final da rede é esperado filtros mais complexos e específicos pois esses tem o papel de extrair características tão específicas que pode ser impossível humanos compreenderem o papel desses filtros. Na Figura 16 é mostrado esse comportamento hierárquico.

4.2.1.4 GradCAM

Abreviação para *Gradient-weighted Class Activation Mapping* proposta por ([SELVARAJU et al., 2016](#)) é uma generalização do CAM (class activation maps), proposta por ([ZHOU et al., 2015](#)). Com essa abordagem é possível localizar as principais áreas de interesse da cada classe no modelo. A ideia geral dessa abordagem é tentar direcionar os mapas de ativação da ultima camada para inferir a relevância dos pixels. E o resultado final é parecido com o mapa de calor parecido com o da Figura 17.

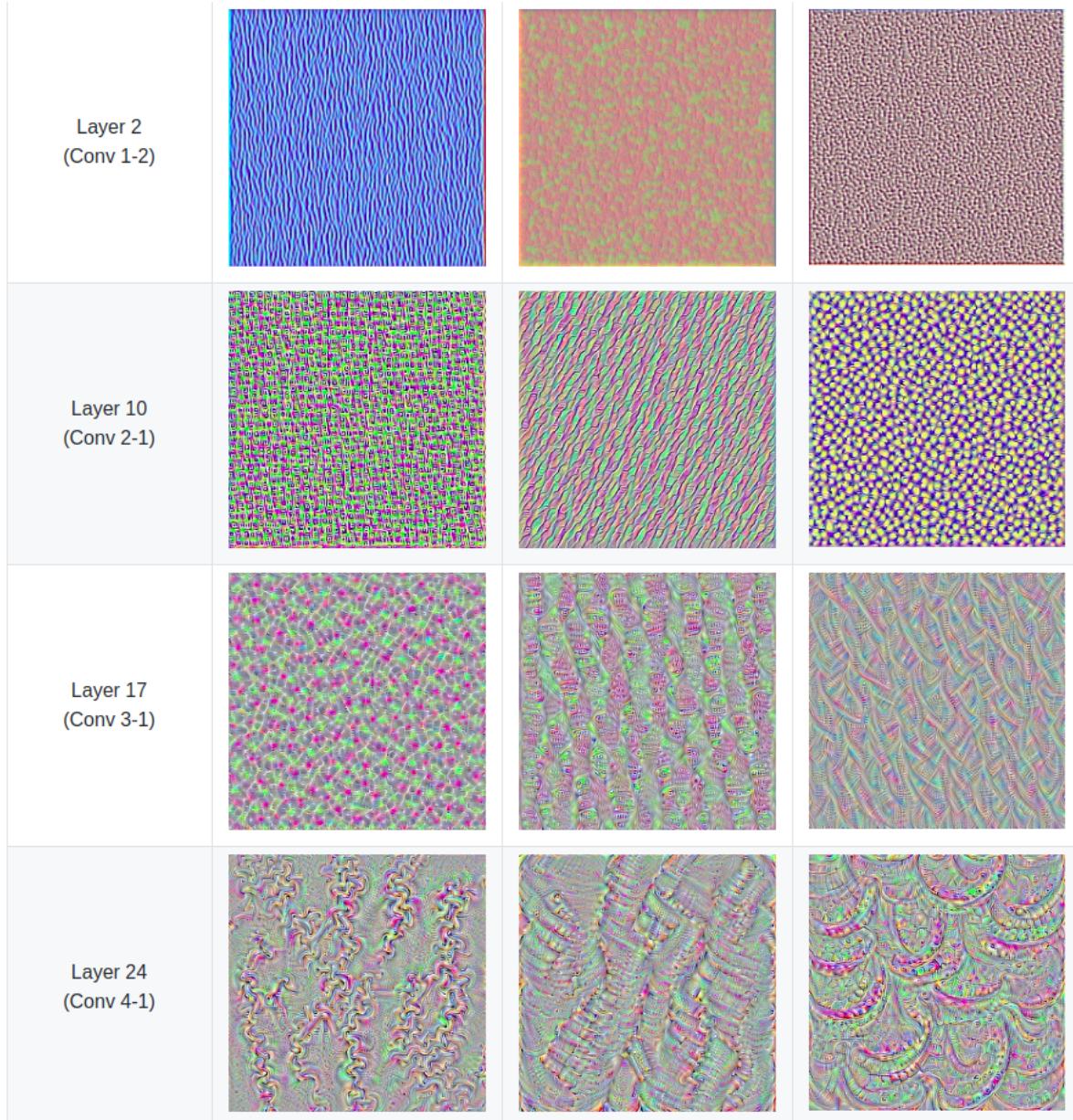


Figura 16 – Representação dos filtros de cada camada convolucional de uma rede neural convolucional.

(OZBULAK, 2019)

4.3 Proposta e cronograma de atividades

A partir da breve introdução dos conceitos básicos de XAI, a proposta é aplicar essas técnicas na rede neural desenvolvida na Seção 3, para desvendar quais são as principais características que essa "*caixa preta*" está observando para fazer as predições de cada classe.

Primeiro será feito o embasamento teórico mais profundo de conceitos subjetivos de XAI. Como a importância dessa temática na área de inteligência artificial. Para explorar esses conceitos as referências teóricas serão os trabalhos de (DOSHI-VELEZ; KIM, 2017;

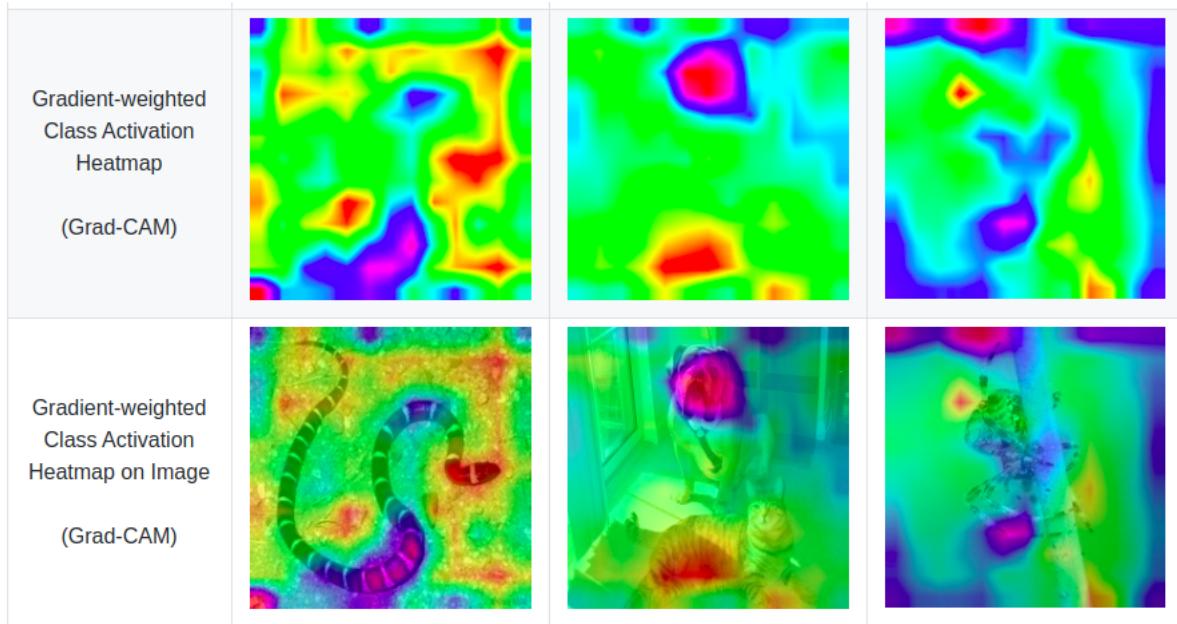


Figura 17 – Exemplo de mapa de calor usando a técnica do GradCAM
[\(OZBULAK, 2019\)](#)

DOSHI-VELEZ et al., 2017) e (TSYMBAL, 2004). Para explorar mais do papel da verdade as referências serão (HOFFMAN et al., 2013) ,(MAYER; DAVIS; SCHOORMAN, 1995) e (BRADSHAW et al., 2004). E por fim, para aferir o quão boa é uma explicação os trabalhos bases serão (MILLER; HOWE; SONENBERG, 2017) , (HILTON, 1990) e (GALE et al., 2018).

Em seguida será implementado a técnica no LIME visto brevemente na Sub Seção 4.2.1.2 que tem como referência principal o trabalho do Ribeiro, Singh and Guestrin (2016a). Portanto será feita uma avaliação mais detalhada da teoria por trás da técnica, bem como desenvolvimento matemático mais profundo dessa metodologia. Como referência principal para implementação será utilizado os repositórios feitos pela comunidade.¹

Seguindo o desenvolvimento, visto brevemente na Sub Seção 4.2.1.3, será feito um embasamento teórico mais profundo nas principais variações baseada em Gradient de acordo com o trabalho do (SELVARAJU et al., 2016). Nessa etapa a primeira variação que será implementada será a técnica do GradCam visto na Sub Seção 4.2.1.4 pois nas referencias correlatas essa técnica tem resultados visuais incríveis para a interpretabilidade do modelo. Como o modelo principal foi implementando usando PyTorch as principais referências de implementação serão repositórios criados pela comunidade.²

¹ Repositórios: <<https://github.com/marcoter/lime>>, <<https://github.com/marvinbuss/ExplainableML-Vision>>, <<https://github.com/alinlab/lime-cam-pytorch>>.

² Repositórios: <<https://github.com/ramprs/grad-cam>>, <<https://github.com/jacobgil/pytorch-grad-cam>>, <<https://github.com/utkuozbulak/pytorch-cnn-visualizations>>.

4.3.1 Cronograma de Atividades

A Tabela 9 resume as principais atividades que serão implementadas com a possível ordem cronológica de implementação com uma estimativa pessimista de implementação para cada fase. Com inicio previsto para fevereiro de 2020 e final previsto em Julho de 2020.

Tabela 9 – Cronograma de atividades para o desenvolvimento das proximas etapas

| Atividades | Meses | | | | | |
|--|-------|-----|-----|-----|-----|-----|
| | Fev | Mar | Abr | Mai | Jun | Jul |
| Revisão bibliográfica e embasamento teórico sobre o LIME | X | X | | | | |
| Implementação e Avaliação no modelo proposto usando LIME | | X | | | | |
| Escrita da seção relacionada ao LIME no TCC | X | X | | | | |
| Revisão bibliográfica e embasamento teórico sobre Gradient-based | | | | X | | |
| Implementação e Avaliação no modelo proposto usando Gradient-based | | | | X | X | X |
| Escrita da seção relacionada ao Gradient-based no TCC | | | | X | X | X |
| Finalização da escrita do TCC | | | | | | X |
| Ajustes finais no texto e correções do TCC | | | | | | X |

4.3.2 Resultados esperados

O fundamento principal de aplicação de XAI é ter uma visão melhor de como são feitas as decisões do modelo. Quais são as principais característica de determinísticas de cada classe. Para tanto, as lesões que tiveram as melhores métricas no reporte da Tabela 10 provavelmente serão as classes que irão produzir os resultados mais interpretáveis.

Portanto acredita-se que as lesões Basal Cell Carcinoma, Malignant Melanoma e Dermatofibroma produzirão os resultados mais expressivos desse trabalho, contudo será avaliado o possivel *overfitting* da classe Melanocytic Nevus pois a métrica esta muito alta em relação as outras classes. O restante das classes serão utilizados para verificar possíveis *underfitting* das decisões.

Na aplicação das técnicas de visualização é esperado resultados semelhantes as Figuras 14, 15, 17, 13 e 16.

Referências

- BIRAN, O.; COTTON, C. V. Explanation and justification in machine learning : A survey or. In: . [S.l.: s.n.], 2017. Citado na página [47](#).
- BLOICE, M.; STOCKER, C.; HOLZINGER, A. Augmentor: An image augmentation library for machine learning. *ArXiv*, abs/1708.04680, 2017. Citado na página [38](#).
- BOWEN, J. T. Classic articles in colonic and rectal surgery. john templeton bowen 1857-1940. precancerous dermatoses: a study of two cases of chronic atypical epithelial proliferation. *Diseases of the colon and rectum*, v. 28 12, p. 982–8, 1983. Citado na página [32](#).
- BRADSHAW, J. M. et al. Toward trustworthy adjustable autonomy in kaos. In: *Trusting Agents for Trusting Electronic Societies*. [S.l.: s.n.], 2004. Citado na página [55](#).
- CANZIANI, A.; PASZKE, A.; CULURCIELLO, E. An analysis of deep neural network models for practical applications. *ArXiv*, abs/1605.07678, 2017. Citado na página [25](#).
- CARUANA, R. et al. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In: *KDD*. [S.l.: s.n.], 2015. Citado na página [46](#).
- CHANDRASEKARAN, B.; TANNER, M. C.; JOSEPHSON, J. R. Explaining control strategies in problem solving. *IEEE Expert*, v. 4, p. 9–15, 1989. Citado na página [46](#).
- CLANCEY, W. J. The epistemology of a rule-based expert system - a framework for explanation. *Artif. Intell.*, v. 20, p. 215–251, 1981. Citado na página [46](#).
- CLANCEY, W. J.; SHORTLIFFE, E. H. Readings in medical artificial intelligence: the first decade. In: . [S.l.: s.n.], 1984. Citado na página [46](#).
- CUA, A. B.; WILHELM, K.; MAIBACH, H. I. Elastic properties of human skin: relation to age, sex, and anatomical region. *Archives of Dermatological Research*, v. 282, p. 283–288, 1990. Citado na página [36](#).
- CUBUK, E. D. et al. Autoaugment: Learning augmentation policies from data. *ArXiv*, abs/1805.09501, 2018. Citado 2 vezes nas páginas [38](#) e [39](#).
- DENGEL, L. T. et al. Total body photography for skin cancer screening. *International journal of dermatology*, v. 54 11, p. 1250–4, 2015. Citado na página [36](#).
- DOSHI-VELEZ, F.; KIM, B. Towards a rigorous science of interpretable machine learning. In: . [S.l.: s.n.], 2017. Citado 2 vezes nas páginas [54](#) e [55](#).
- DOSHI-VELEZ, F. et al. Accountability of ai under the law: The role of explanation. *ArXiv*, abs/1711.01134, 2017. Citado 2 vezes nas páginas [54](#) e [55](#).
- ERHAN, D. et al. Visualizing higher-layer features of a deep network. In: . [S.l.: s.n.], 2009. Citado na página [52](#).
- ESTEVA, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, v. 542, 01 2017. Citado na página [45](#).

- FONG, R. C.; VEDALDI, A. Interpretable explanations of black boxes by meaningful perturbation. *2017 IEEE International Conference on Computer Vision (ICCV)*, p. 3449–3457, 2017. Citado na página 50.
- GALE, W. et al. Producing radiologist-quality reports for interpretable artificial intelligence. *ArXiv*, abs/1806.00340, 2018. Citado na página 55.
- GIOTIS, I. et al. Med-node: A computer-assisted melanoma diagnosis system using non-dermoscopic images. *Expert Syst. Appl.*, v. 42, p. 6578–6585, 2015. Citado na página 26.
- GUPTA, S. et al. Bowen disease over photoprotected site in an indian male. *Dermatology online journal*, v. 15 10, p. 16, 2009. Citado na página 32.
- HAFEMANN, L. G.; SABOURIN, R.; OLIVEIRA, L. E. S. de. Writer-independent feature learning for offline signature verification using deep convolutional neural networks. *2016 International Joint Conference on Neural Networks (IJCNN)*, p. 2576–2583, 2016. Citado na página 22.
- HAN, S. S. et al. Classification of the clinical images for benign and malignant cutaneous tumors using a deep learning algorithm. *The Journal of investigative dermatology*, v. 138 7, p. 1529–1538, 2018. Citado 5 vezes nas páginas 26, 34, 36, 43 e 45.
- HAYKIN, S. Neural networks: A comprehensive foundation. In: . [S.l.: s.n.], 1998. Citado 2 vezes nas páginas 16 e 23.
- HE, K. et al. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. Citado na página 26.
- HILTON, D. J. Conversational processes and causal explanation. In: . [S.l.: s.n.], 1990. Citado na página 55.
- HOFFMAN, R. R. et al. Trust in automation. *IEEE Intelligent Systems*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 28, n. 1, p. 84–88, jan. 2013. ISSN 1541-1672. Disponível em: <<https://doi.org/10.1109/MIS.2013.24>>. Citado na página 55.
- HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, v. 195 1, p. 215–43, 1968. Citado na página 19.
- INCA. *Instituto Nacional de Câncer*. 2018. (Acessado em: 03-08-2019). Disponível em: <<https://www.inca.gov.br/>>. Citado na página 31.
- ISDIS. *International Skin Imaging Collaboration Project*. 2018. (Acessado em: 07-08-2019). Disponível em: <<https://isic-archive.com>>. Citado na página 27.
- JY RAMSEY ML. CANCER, S. C. o. t. S. H. [Updated 2018 Nov 15]. In: *StatPearls [Internet]*. Treasure Island (FL): StatPearls Publishing; 2019 Jan-. 2019. (Acessado em: 30-09-2019). Disponível em: <<https://www.ncbi.nlm.nih.gov/books/NBK441939/>>. Citado na página 34.

- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *NIPS*. [S.l.: s.n.], 2012. Citado 2 vezes nas páginas 16 e 38.
- LI, F.-F.; KARPATHY, A.; JOHNSON, J. Cs231n: Convolutional neural networks for visual recognition 2016. Disponível em: <<http://cs231n.stanford.edu/>>. Citado na página 22.
- LIPTON, Z. C. The mythos of model interpretability. *ArXiv*, abs/1606.03490, 2016. Citado na página 48.
- MATSUGU, M. et al. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural networks : the official journal of the International Neural Network Society*, v. 16 5-6, p. 555–9, 2003. Citado 2 vezes nas páginas 16 e 19.
- MAYER, R. C.; DAVIS, J. H.; SCHORMAN, F. D. An integrative model of organizational trust. In: . [S.l.: s.n.], 1995. Citado na página 55.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, v. 52, p. 99–115, 1988. Citado na página 19.
- MENDES, D.; SILVA, N. *Skin Lesions Classification Using Convolutional Neural Networks in Clinical Images*. 2018. Citado 2 vezes nas páginas 18 e 45.
- MILLER, T. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, v. 267, p. 1–38, 2017. Citado na página 47.
- MILLER, T.; HOWE, P.; SONENBERG, L. Explainable ai: Beware of inmates running the asylum or: How i learnt to stop worrying and love the social and behavioural sciences. *ArXiv*, abs/1712.00547, 2017. Citado na página 55.
- MILLS, S. E.; COOPER, P. H.; FECHNER, R. E. Lobular capillary hemangioma: the underlying lesion of pyogenic granuloma. a study of 73 cases from the oral and nasal mucous membranes. *The American journal of surgical pathology*, v. 4 5, p. 470–9, 1980. Citado na página 33.
- MOLNAR, C. *Interpretable Machine Learning*: A guide for making black box models explainable. [S.l.: s.n.], 2019. <<https://christophm.github.io/interpretable-ml-book/>>. Citado na página 49.
- MORTON, C. A.; BIRNIE, A. J.; EEDY, D. J. British association of dermatologists' guidelines for the management of squamous cell carcinoma in situ (bowen's disease) 2014. *The British journal of dermatology*, v. 170 2, p. 245–60, 2014. Citado na página 32.
- MOY, R. L. Clinical presentation of actinic keratoses and squamous cell carcinoma. *Journal of the American Academy of Dermatology*, v. 42 1 Pt 2, p. 8–10, 2000. Citado na página 29.
- NAVERSEN, D. N. et al. Painful tumors of the skin: "lend an egg". *Journal of the American Academy of Dermatology*, v. 28 2 Pt 2, p. 298–300, 1993. Citado na página 31.

- NCI. *Common Moles, Displastic Nevi, and Risk of Melanoma*. 2018. 2018b. (Acessado em: 15-09-2019). Disponível em: <<https://www.cancer.gov/types/skin/moles-fact-sheet>>. Citado na página 33.
- NCI. *Melanoma Treatment (PDQ®)-Health Professional Version*. 2018c. (Acessado em: 15-09-2019). Disponível em: <<https://www.cancer.gov/types/skin/hp/melanoma-treatment-pdq>>. Citado 2 vezes nas páginas 32 e 33.
- NIELSEN; MICHAEL, A. *Neural Networks and Deep Learning*. Determination Press, 2018, 2018. Disponível em: <<http://neuralnetworksanddeeplearning.com/>>. Citado 2 vezes nas páginas 20 e 21.
- ODOM, W. D. J. T. G. B. D. M. E. R. B. *Dermal and subcutaneous tumors: cherry angiomas*. 2000. (Acessado em: 03-08-2019). Citado na página 32.
- OZBULAK, U. *PyTorch CNN Visualizations*. [S.l.]: GitHub, 2019. <<https://github.com/utkuozbulak/pytorch-cnn-visualizations>>. Citado 4 vezes nas páginas 52, 53, 54 e 55.
- PATTERSON, J. *Weedon's Skin Pathology E-Book*. Elsevier Health Sciences, 2014. ISBN 9780702062001. Disponível em: <<https://books.google.com.br/books?id=Y-LTBQAAQBAJ>>. Citado na página 29.
- PEREZ, L.; WANG, J. The effectiveness of data augmentation in image classification using deep learning. *ArXiv*, abs/1712.04621, 2017. Citado 2 vezes nas páginas 38 e 39.
- RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. Model-agnostic interpretability of machine learning. *ArXiv*, abs/1606.05386, 2016. Citado 3 vezes nas páginas 48, 50 e 51.
- ROBNIK-SIKONJA, M.; BOHANEC, M. Perturbation-based explanations of prediction models. In: *Human and Machine Learning*. [S.l.: s.n.], 2018. Citado na página 49.
- ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, p. 65–386, 1958. Citado na página 18.
- RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, v. 115, p. 211–252, 2014. Citado na página 25.
- SAUDE, M. da. *Ministério da Saúde, sistema de informações sobre mortalidade*. 2017. (Acessado em: 20-10-2019). Disponível em: <<http://datasus.saude.gov.br/>>. Citado na página 31.
- SBD. *Instituto Nacional de Câncer, Dermatofibroma*. 2018. (Acessado em: 03-08-2019). Disponível em: <<https://www.sbd.org.br/dermatologia/pele/doencas-e-problemas/dermatofibroma/77/>>. Citado na página 31.
- SELVARAJU, R. R. et al. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *ArXiv*, abs/1610.02391, 2016. Citado 2 vezes nas páginas 53 e 55.
- SILVA., I. N. de Câncer José Alencar Gomes da. *Estimativa 2018 – Incidência de Câncer no Brasil*. 2018b. (Acessado em: 15-09-2019). Disponível em: <<http://www1.inca.gov.br/inca/Arquivos/estimativa-2018.pdf>>. Citado na página 32.

- SMILKOV, D. et al. Smoothgrad: removing noise by adding noise. *ArXiv*, abs/1706.03825, 2017. Citado na página 52.
- SMITH, L. N. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *ArXiv*, abs/1803.09820, 2018. Citado na página 44.
- SOCIETY, A. C. *Cancer Facts Figures 2017*. 2017. (Acessado em: 20.11.2019). Disponível em: <<https://www.cancer.org/research/cancer-facts-statistics/all-cancer-facts-figures/cancer-facts-figures-2017.html>>. Citado 4 vezes nas páginas 29, 31, 32 e 34.
- SRIVASTAVA, R. K.; GREFF, K.; SCHMIDHUBER, J. *Highway Networks*. 2015. Cite arxiv:1505.00387Comment: 6 pages, 2 figures. Presented at ICML 2015 Deep Learning workshop. Full paper is at arXiv:1507.06228. Disponível em: <<http://arxiv.org/abs/1505.00387>>. Citado na página 25.
- SUNDARARAJAN, M.; TALY, A.; YAN, Q. Axiomatic attribution for deep networks. In: *ICML*. [S.l.: s.n.], 2017. Citado na página 52.
- SWETS, J. A. Indices of discrimination or diagnostic accuracy: their rocs and implied models. *Psychological bulletin*, v. 99 1, p. 100–17, 1986. Citado na página 24.
- TEACH, R. L.; SHORTLIFFE, E. H. An analysis of physician attitudes regarding computer-based clinical consultation systems. *Computers and biomedical research, an international journal*, v. 14 6, p. 542–58, 1981. Citado na página 46.
- TSYMBAL, A. The problem of concept drift: definitions and related work. In: . [S.l.: s.n.], 2004. Citado na página 55.
- WERBOS, P. J. Applications of advances in nonlinear sensitivity analysis. In: *Proceedings of the 10th IFIP Conference, 31.8 - 4.9, NYC*. [S.l.: s.n.], 1981. p. 762–770. Citado na página 19.
- YOSINSKI, J. et al. How transferable are features in deep neural networks? In: GHAHRAMANI, Z. et al. (Ed.). *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014. p. 3320–3328. Disponível em: <<http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf>>. Citado na página 38.
- ZHOU, B. et al. Learning deep features for discriminative localization. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 2921–2929, 2015. Citado na página 53.

Apêndices

APÊNDICE A – Web Scraping

Código Fonte A.1 - Web Scraping para extrair as imagens

```
# importing the library
from google_images_download import google_images_download

# Definindo os sites de confiança
SITES_LIST = ["http://www.dermatlas.net/",
              "http://www.dermis.net/dermisroot/en/home/index.htm",
              "http://www.meddean.luc.edu/lumen/MedEd/medicine/ \
dermatology/melton/atlas.htm",
              "http://www.dermatoweb.net/",
              "http://www.atlasdermatologico.com.br/",
              "http://www.hellenicdermatlas.com/en/?params=en"]

# Palavras-chaves para pesquisar
SKIN_LESION = ["Actinic Keratosis",
                "Basal cell carcinoma",
                "Dermatofibroma",
                "Hemangioma",
                "Intraepithelial carcinoma",
                "Bowen's disease",
                "Lentigo",
                "Malignant melanoma",
                "Melanocytic nevus",
                "Pyogenic granuloma",
                "Seborrheic keratosis",
                "Squamous cell carcinoma",
                "Wart"]

for skin_lesion in SKIN_LESION:
    for site in SITE:
        print(site)
        response = google_images_download.googleimagesdownload()

        # creating list of arguments
        arguments = {
            "keywords": str(skin_lesion),
            "limit": 1000,
```

```
"specific_site": str(site),  
"output_directory": "dataset_2",  
"print_urls":True  
}  
  
# passing the arguments to the function  
paths = response.download(arguments)  
print(paths)
```

APÊNDICE B – Conversão das imagens e redimensionamento prévio

Código Fonte B.1 - Para converter para .jpg, renomear e redimensionar

```
'''  
Notebook to resize imagens in base dataset and convert all to RGB  
'''  
  
import os, sys  
from PIL import Image  
  
# diretocio/<pastas das leses>  
path=[‘PATH DO DIRETÓRIO ONDE ESTA A BASE DE DADOS’]  
output_path=[‘PATH DO DIRETÓRIO ONDE DESEJA SALVAR AS NOVAS IMAGENS’]  
dirs = os.listdir(path)  
  
def resize_convert_rename():  
    '''  
    Function to rename imagens with acceding number, convert images to rgb and  
    rezise all imagens to 448x488 to reduce process complexity in later  
    operations  
    '''  
  
    i = 0  
    for sub_dir in dirs:  
        for item in os.listdir(path + sub_dir):  
            if os.path.isfile(path + sub_dir + ‘/’ + item):  
                im = Image.open(path + sub_dir + ‘/’ + item)  
                f, e = os.path.splitext(path + sub_dir + ‘/’ + item)  
                imResize = im.resize((448,448), Image.ANTIALIAS)  
                rgb_im = imResize.convert(‘RGB’)  
                rgb_im.save(output_path + sub_dir + ‘/’ + str(i) + ‘.jpg’,  
                            ‘JPEG’, quality=90)  
                i = i + 1  
  
resize convert rename()
```

APÊNDICE C – Separação da base de forma estratificada

Código Fonte C.1 - Separação de forma estratificada

```
'''  
Slipt dataset in a stratified way  
  
train: 75%, test: 10%. val: 15%  
  
seed: 12345  
  
'''  
import split_folders  
  
# Split with a ratio.  
# To only split into training and validation set, set a tuple to 'ratio', i.e,  
# (.8, .2).  
split_folders.ratio(['PATH DO DIRETRIO DE ORIGEM'],  
                  output=['PATH DE SAIDA ONDE SER CRIADO /test /train /val'],  
                  seed=12345, ratio=(.75, .15, .1)) # default values
```

APÊNDICE D – Data augmentation

Código Fonte D.1 - Data augmentation

```

import Augmentor as aug
import glob
import os
import numpy as np
import cv2
import PIL
from Augmentor.Operations import Operation

class Lightning(Operation):
    def __init__(self, probability, intensity_low=0.7, intensity_high=1.2):
        Operation.__init__(self, probability)
        # Init classes variables with default values
        # Default values treshold intent to create a optimal range
        # Imagens cant be too dark or too brigher
        self.intensity_low = intensity_low
        self.intensity_high = intensity_high

    def perform_operation(self, images):
        for i, image in enumerate(images):
            image = np.array(image.convert('RGB'))
            row, col, _ = image.shape
            light_intensity = np.random.randint(int(self.intensity_low * 100),
                                                int(self.intensity_high * 100))

            light_intensity /= 100

            gaussian = 100 * np.random.random((row, col, 1))
            gaussian = np.array(gaussian, dtype=np.uint8)
            gaussian = np.concatenate((gaussian, gaussian, gaussian), axis=2)
            image = cv2.addWeighted(image, light_intensity, gaussian, 0.25, 0)

            image = PIL.Image.fromarray(image)
            images[i] = image

    return images

```

```
# Multiplier used to set the final augmented images number
MULTIPLIER=30

# Default dir where we can find the train dataset
TRAIN_DIRECTORY_DATASET =
    '/home/helpthx/Desktop/TCC-1/TCC-1-UnB/dataset-split_final/val/*'

folders = []
for f in glob.glob(TRAIN_DIRECTORY_DATASET):
    if os.path.isdir(f):
        folders.append(os.path.abspath(f))

print('Classes found {}'.format([os.path.split(x)[1] for x in folders]))
print('Numb: ', len([os.path.split(x)[1] for x in folders]))

# Dictionari to hold the abspath and class's name
pipelines = {}

for folder in folders:
    pipelines[os.path.split(folder)[1]] =
        (aug.Pipeline(source_directory=folder,
                      output_directory='resnet-augmented'))


classes_count = []
for p in pipelines.values():
    print("Class {} has {} samples".format(p.augmentor_images[0].class_label,
                                            len(p.augmentor_images)))

    classes_count.append(len(p.augmentor_images))

# Instantiating Lighthing Class with 50 % probability
lightning = Lightning(probability=0.5)

for p in pipelines.values():
    # 50 % of rotation the imagem with max left and max right
    p.rotate(probability=0.5, max_left_rotation=10, max_right_rotation=10)

    # 40 % of zoom inside the imagem with a 90 % cover area
    p.zoom_random(probability=0.4, percentage_area=0.9)

    # 70 % of mirror vertical imagem for 50 % left or righth
```

```
p.flip_left_right(probability=0.7)

# 50 % of mirror horizontal
p.flip_top_bottom(probability=0.5)

# Applying some distortion in the image
p.random_distortion(probability=0.8, grid_width=5, grid_height=5,
                     magnitude=15)

# Custom lightning of 50 %
p.add_operation(lightning)

# Rezise all the imagens size for default restnet 224x224
p.resize(probability=1.0, width=224, height=224)

# If a equal sampling of the lesions is needed
# Mind that the final MULTIPLIER can scale many times if True

SAME_SAMPLING = False
for p in pipelines.values():
    if SAME_SAMPLING:
        diff = max(classes_count) - len(p.augmentor_images)
        p.sample((len(p.augmentor_images) + diff)*MULTIPLIER + diff)
    else:
        p.sample(len(p.augmentor_images)*MULTIPLIER)
```

APÊNDICE E – Treinamento e validação da rede

Código Fonte E.1 - Treinamento e validação em pythorch

```

import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import torchvision
import csv
import matplotlib.pyplot as plt
import time
import os
import copy
from torchvision import datasets, models, transforms
from __future__ import print_function
from __future__ import division
from torch.optim import lr_scheduler

print("PyTorch Version: ",torch.__version__)
print("Torchvision Version: ",torchvision.__version__)

# Top level data directory. Here we assume the format of the directory conforms
# to the ImageFolder structure
data_dir='/content/gdrive/My
Drive/UnB/TCC-1/TCC1-1-dataset-final/dataset_final'

MODEL_SAVE_DIR='/content/gdrive/My Drive/UnB/TCC-1/TCC1-1-dataset-final'

CSV_TRAIN_DIR='/content/gdrive/My Drive/UnB/TCC-1/TCC1-1-dataset-final'

# If you dont have a pre-trained model or want to recovery traing
# let this field None or False
PREVIOUS_TRAINED_MODEL=False

# Models to choose from [resnet]
model_name='resnet'

```

```
# Number of classes in the dataset
num_classes=9

# Batch size for training (change depending on how much memory you have)
batch_size=32

# Number of epochs to train for
num_epochs=100

# Flag for feature extracting. When False, we finetune the whole model,
# when True we only update the reshaped layer params
feature_extract=False

# Hyperparamerts
STEP_SIZE_CONST=1

GAMMA_CONST=0.1

LR_CONST=0.001

MOMENTUM_COST=0.9

def train_model(model, dataloaders, criterion, scheduler, optimizer,
    num_epochs=25, is_inception=False):
    ...

The train_model function handles the training and validation of a given model.
As input, it takes a PyTorch model,
a dictionary of dataloaders, a loss function, an optimizer, a specified
number of epochs to train and validate for,
and a boolean flag for when the model is an Inception model. The
is_inception flag is used to accomodate the
Inception v3 model, as that architecture uses an auxiliary output and the
overall model loss respects both the
auxiliary output and the final output, as
described here
<https://discuss.pytorch.org/t/how-to-optimize-inception-model-with-auxiliary-classifier3/4279>
The function trains for the specified number of epochs and after each epoch
runs a full validation step. It also
keeps track of the best performing model (in terms of validation accuracy),
and at the end of training returns the
best performing model. After each epoch, the training and validation
```

```
accuracies are printed.  
'''  
since = time.time()  
  
val_acc_history = []  
  
best_model_wts = copy.deepcopy(model.state_dict())  
best_acc = 0.0  
  
for epoch in range(num_epochs):  
    since_epoch = time.time()  
    print('Epoch {}/{}'.format(epoch, num_epochs - 1))  
    print('-' * 10)  
  
    # Each epoch has a training and validation phase  
    for phase in ['train', 'val']:  
        if phase == 'train':  
            model.train() # Set model to training mode  
  
        else:  
            model.eval() # Set model to evaluate mode  
  
        running_loss = 0.0  
        running_corrects = 0  
  
        # Iterate over data.  
        for inputs, labels in dataloaders[phase]:  
            inputs = inputs.to(device)  
            labels = labels.to(device)  
  
            # zero the parameter gradients  
            optimizer.zero_grad()  
  
            # forward  
            # track history if only in train  
            with torch.set_grad_enabled(phase == 'train'):  
                outputs = model(inputs)  
                _, preds = torch.max(outputs, 1)  
                loss = criterion(outputs, labels)  
  
                # backward + optimize only if in training phase  
                if phase == 'train':
```

```
        loss.backward()
        optimizer.step()

        # statistics
        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)

    if phase == 'train':
        scheduler.step()

    epoch_loss = running_loss / len(dataloaders[phase].dataset)
    epoch_acc = running_corrects.double() /
        len(dataloaders[phase].dataset)

    time_elapsed_epoch = time.time() - since_epoch
    print('Epoch complete in {:.0f}m {:.0f}s'.format(time_elapsed_epoch
        // 60, time_elapsed_epoch % 60))
    print('{} Loss: {:.4f} Acc: {:.4f}'.format(phase, epoch_loss,
        epoch_acc))

    # Write in csv training infos
    row = [phase, epoch_loss, epoch_acc]
    with open(CSV_TRAIN_DIR + '/train_val_phase.csv', 'a') as csvFile:
        writer = csv.writer(csvFile)
        writer.writerow(row)

    csvFile.close()

    # deep copy the model
    if phase == 'val' and epoch_acc > best_acc:
        best_acc = epoch_acc
        best_model_wts = copy.deepcopy(model.state_dict())
        model.class_to_idx = image_datasets['train'].class_to_idx
        state = {
            'epoch': epoch,
            'arch': 'resnet152',
            'state_dict': model.state_dict(),
            'class_to_idx': model.class_to_idx,
            'optimizer': optimizer.state_dict(),
        }

        torch.save(state, MODEL_SAVE_DIR +
```

```

        '/restnet_model152_trained_exp7.pt')

    if phase == 'val':
        val_acc_history.append(epoch_acc)

    print()

time_elapsed = time.time() - since
print('Training complete in {:.0f}m {:.0f}s'.format(time_elapsed // 60,
    time_elapsed % 60))
print('Best val Acc: {:.4f}'.format(best_acc))

# load best model weights
model.load_state_dict(best_model_wts)
return model, val_acc_history

def set_parameter_requires_grad(model, feature_extracting):
    """
    This helper function sets the .requires_grad attribute of the parameters in
    the model to False when we are feature
    extracting. By default, when we load a pretrained model all of the parameters
    have .requires_grad=True, which is fine
    if we are training from scratch or finetuning. However, if we are feature
    extracting and only want to compute
    gradients for the newly initialized layer then we want all of the other
    parameters to not require gradients.
    This will make more sense later.
    """
    if feature_extracting:
        for param in model.parameters():
            param.requires_grad = False

def initialize_model(model_name, num_classes, feature_extract,
use_pretrained=True):
    # Initialize these variables which will be set in this if statement. Each
    # of these
    #   variables is model specific.
    model_ft = None
    input_size = 0

    if model_name == "resnet":
        """ Resnet152

```

```
"""
model_ft = models.resnet152(pretrained=use_pretrained)
set_parameter_requires_grad(model_ft, feature_extract)
num_ftrs = model_ft.fc.in_features
model_ft.fc = nn.Linear(num_ftrs, num_classes)
input_size = 448

elif model_name == "vgg":
    """
    VGG11_bn
    """

    model_ft = models.vgg11_bn(pretrained=use_pretrained)
    set_parameter_requires_grad(model_ft, feature_extract)
    num_ftrs = model_ft.classifier[6].in_features
    model_ft.classifier[6] = nn.Linear(num_ftrs,num_classes)
    input_size = 224

else:
    print("Invalid model name, exiting...")
    exit()

return model_ft, input_size

# Initialize the model for this run
model_ft, input_size = initialize_model(model_name, num_classes,
                                         feature_extract, use_pretrained=True)

# Print the model we just instantiated
# print(model_ft)

# Data augmentation and normalization for training
# Just normalization for validation
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])
}
```

```
]),  
}  
  
print("Initializing Datasets and Dataloaders...")  
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),  
    data_transforms[x]) for x in ['train', 'val']}  
dataloaders_dict = {x: torch.utils.data.DataLoader(image_datasets[x],  
    batch_size=batch_size,  
    shuffle=True, num_workers=16)  
    for x in ['train', 'val']}  
  
dataset_sizes = {x: len(image_datasets[x])  
    for x in ['train', 'val']}  
  
class_names = image_datasets['train'].classes  
  
# Detect if we have a GPU available  
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")  
  
print(dataset_sizes)  
  
# Send the model to GPU  
model_ft = model_ft.to(device)  
  
# Gather the parameters to be optimized/updated in this run. If we are  
# finetuning we will be updating all parameters. However, if we are  
# doing feature extract method, we will only update the parameters  
# that we have just initialized, i.e. the parameters with requires_grad  
# is True.  
  
params_to_update = model_ft.parameters()  
  
print("Params to learn:")  
  
if feature_extract:  
    params_to_update = []  
    for name,param in model_ft.named_parameters():  
        if param.requires_grad == True:  
            params_to_update.append(param)  
            print("\t",name)  
else:
```

```
for name,param in model_ft.named_parameters():
    if param.requires_grad == True:
        print("\t",name)

# Observe that all parameters are being optimized
# optimizer_ft = optim.SGD(params_to_update, lr=LR_CONST,
#                           momentum=MOMENTUM_COST)

optimizer_ft = optim.SGD(params_to_update, lr=0.01, weight_decay=0.00001,
                        momentum=MOMENTUM_COST)

exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft,
                                       step_size=STEP_SIZE_CONST, gamma=GAMMA_CONST)

# Find total parameters and trainable parameters
total_params = sum(p.numel() for p in model_ft.parameters())
print(f'{total_params:,} total parameters.')
total_trainable_params = sum(
    p.numel() for p in model_ft.parameters() if p.requires_grad)
print(f'{total_trainable_params:,} training parameters.')

# Should result in
# 58,162,249 total parameters.
# 58,162,249 training parameters.

# Setup the loss fxn
criterion = nn.CrossEntropyLoss()

# Train and evaluate
model_ft, hist = train_model(model_ft, dataloaders_dict, criterion,
                             exp_lr_scheduler,
                             optimizer_ft, num_epochs=num_epochs,
                             is_inception=(model_name=="inception"))
```

APÊNDICE F – Teste e criação das métricas

Código Fonte E.1 - Teste e criação das métricas e gráficos

```

import torch
import json
import seaborn as sn
import pandas as pd
import numpy as np
import torch.nn.functional as F
import sklearn.metrics as skm
import torch.optim as optim
import matplotlib.pyplot as plt
from torch import nn
from torch import optim
from torchvision import datasets, transforms, models
from torch.optim import lr_scheduler
from PIL import Image
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
from sklearn.metrics import multilabel_confusion_matrix
%matplotlib inline

# check if CUDA is available
train_on_gpu = torch.cuda.is_available()

if not train_on_gpu:
    print('CUDA is not available. Training on CPU ...')
else:
    print('CUDA is available! Training on GPU ...')

# Dataset dir base
dataset_dir = '/content/gdrive/My
Drive/UnB/TCC-1/TCC1-1-dataset-final/dataset_final'

```

```
test_dir = '/content/gdrive/My  
Drive/UnB/TCC-1/TCC1-1-dataset-final/dataset_final/test'  
  
# Const to save test infos in csv  
CSV_TEST_DIR='/content/gdrive/My  
Drive/UnB/TCC-1/TCC1-1-dataset-final/dataset_final'  
  
# Model dir  
pre_model_dir='/content/gdrive/My  
Drive/UnB/TCC-1/TCC1-1-dataset-final/restnet_model152_trained_exp7.pt'  
  
# Image test from test_dir  
image_test_dir='/content/gdrive/My  
Drive/UnB/TCC-1/TCC1-1-dataset/dataset-split/test/basal-cell-carcinoma/ \  
basal-cell-carcinoma_original_3210.jpg_6e88dd4a-6a26-4c3c-a21d-8c92f9cd35f8.jpg'  
  
batch_size=6  
  
  
mean = [0.485, 0.456, 0.406]  
std = [0.229, 0.224, 0.225]  
  
test_transforms = transforms.Compose([transforms.Resize(224),  
                                     transforms.ToTensor(),  
                                     transforms.Normalize(mean,  
                                     std)  
                                     ])  
  
test_data = datasets.ImageFolder(test_dir, transform = test_transforms)  
  
testloader = torch.utils.data.DataLoader(test_data, batch_size=1, shuffle=True)  
  
model_name='resnet'  
num_classes = 9  
feature_extract = False  
  
def set_parameter_requires_grad(model, feature_extracting):  
    if feature_extracting:  
        for param in model.parameters():  
            param.requires_grad = False
```

```
def initialize_model(model_name, num_classes, feature_extract,
    use_pretrained=True):
    # Initialize these variables which will be set in this if statement. Each
    # of these
    #   variables is model specific.
    model_ft = None
    input_size = 0

    if model_name == "resnet":
        """
        Resnet152
        """

        model_ft = models.resnet152(pretrained=use_pretrained)
        set_parameter_requires_grad(model_ft, feature_extract)
        num_ftrs = model_ft.fc.in_features
        model_ft.fc = nn.Linear(num_ftrs, num_classes)
        input_size = 224

    else:
        print("Invalid model name, exiting...")
        exit()

    return model_ft, input_size

# Initialize the model for this run
model_ft, input_size = initialize_model(model_name, num_classes,
    feature_extract, use_pretrained=True)

if train_on_gpu:
    state = torch.load(pre_model_dir)
else:
    state = torch.load(pre_model_dir, map_location='cpu')

# Loading weights in restnet architecture
model_ft.load_state_dict(state['state_dict'])

classes_skin = state['class_to_idx']
print(classes_skin)

# Expected
# {'actinic-keratosis': 0,
#  'basal-cell-carcinoma': 1,
#  'dermatofibroma': 2,
```

```
# 'hemangioma': 3,
# 'intraepithelial-carcinoma': 4,
# 'malignant-melanoma': 5,
# 'melanocytic-nevus': 6,
# 'pyogenic-granuloma': 7,
# 'squamous-cell-carcinoma': 8}

def process_image(image):
    ''' Scales, crops, and normalizes a PIL image for a PyTorch model,
        returns an Numpy array
    '''

    # Resize where shortest side is 256px, keeping aspect ratio
    minside = 256
    img = Image.open(image)
    imagex, imagey = img.size
    aspect = float(imagex) / float(imagey)

    if imagex <= imagey:
        width = 256
        height = int(width / aspect)
    else:
        height = 256
        width = int(height * aspect)

    img = img.resize((width, height), Image.ANTIALIAS)

    # Crop out center 224 x 224
    left = (width - 224) / 2
    top = (height - 224) / 2
    right = (width + 224) / 2
    bottom = (height + 224) / 2
    img = img.crop((left, top, right, bottom))

    # Convert image to numpy array
    np_image = np.array(img)
    np_image = np_image / 255

    # Normalize image
    np_image -= [0.485, 0.456, 0.406]
    np_image /= [0.229, 0.224, 0.225]
```

```
# Transpose array:  
result = np_image.transpose(-1, 0, 1)  
  
return result  
  
def imshow(image, ax=None, title=None):  
    """Imshow for Tensor."""  
    if ax is None:  
        fig, ax = plt.subplots()  
  
    # PyTorch tensors assume the color channel is the first dimension  
    # but matplotlib assumes is the third dimension  
    image = image.numpy().transpose((1, 2, 0))  
  
    # Undo preprocessing  
    mean = np.array([0.485, 0.456, 0.406])  
    std = np.array([0.229, 0.224, 0.225])  
    image = std * image + mean  
  
    # Image needs to be clipped between 0 and 1 or it looks like noise when  
    # displayed  
    image = np.clip(image, 0, 1)  
  
    ax.imshow(image)  
  
    return ax  
  
imagem_teste = '/content/gdrive/My  
Drive/UnB/TCC-1/TCC-1-dataset/dataset-split/dataset-test/dermatofibroma/10.jpg'  
  
result = process_image(imagem_teste)  
res = torch.from_numpy(result)  
imshow(res)  
  
def predict(image_path, model, topk=5):  
    ''' Predict the class (or classes) of an image using a trained deep  
    learning model.  
    ...  
  
    # TODO: Implement the code to predict the class from an image file  
    model.eval()
```

```
image = process_image(image_path)
image = torch.from_numpy(image)

if train_on_gpu:
    model.cuda()
    image = image.cuda()
image = image.float().unsqueeze(0)
out = model.forward(image)
logps = F.log_softmax(out)
ps = torch.exp(logps)
probs, classes = ps.topk(topk, dim=1)

if train_on_gpu:
    probs = list(probs.squeeze(0).cpu().detach().numpy())
    classes = list(classes.squeeze(0).cpu().detach().numpy())
else:
    probs = list(probs.squeeze(0).detach().numpy())
    classes = list(classes.squeeze(0).detach().numpy())
idx_class_mapping = dict((v, k) for k, v in test_data.class_to_idx.items())
classes = list(map(lambda x: idx_class_mapping[x], classes))

return probs, classes

probs, classes = predict(imagem_teste, model_ft)
print(probs)
print(classes)

# Expected
# [0.5677406, 0.34506133, 0.08256749, 0.004574562, 2.4455296e-05]
# ['actinic-keratosis', 'dermatofibroma', 'basal-cell-carcinoma',
# 'squamous-cell-carcinoma', 'malignant-melanoma']

skin_list = list(classes_skin.keys())

def plot_bar(image_path, model):
    result = process_image(image_path)
    res = torch.from_numpy(result)
    fig, (ax1, ax2) = plt.subplots(figsize=(6,9), ncols=2)
    ax1 = imshow(res, ax1)
    probs, classes = predict(image_path, model)
    ax2.bart(np.arange(len(probs)), probs)
```

```
ax2.set_aspect(0.1)
ax2.set_yticks(np.arange(len(probs)))
v = list(skin_list)
# classes = list(map(lambda x: skin_lesion_to_name[x], classes))
ax2.set_yticklabels(classes, size='small');
ax2.set_title('Class Probability')
ax2.set_xlim(0, 1.1)
plt.tight_layout()

plot_bar('/content/gdrive/My
Drive/UnB/TCC-1/TCC1-1-dataset-final/dataset_final/test/actinic-keratosis/1604.jpg',
model_ft)

def test_label_predictions(model, device, test_loader):
    model.eval()
    actuals = []
    predictions = []
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            prediction = output.argmax(dim=1, keepdim=True)
            actuals.extend(target.view_as(prediction))
            predictions.extend(prediction)
    return [i.item() for i in actuals], [i.item() for i in predictions]

actuals, predictions = test_label_predictions(model_ft, 'cuda', testloader)

print('Confusion matrix:')
print(confusion_matrix(actuals, predictions))
print('F1 score: %f' % f1_score(actuals, predictions, average='macro'))
print('Accuracy score: %f' % accuracy_score(actuals, predictions))

# Results
# Confusion matrix:
# [[13  3  0  0  0  1  0  0  3]
# [ 0 73  2  0  1  4  1  1  2]
# [ 0  1 18  1  1  0  0  0  1]
# [ 0  0  0 16  0  1  0  2  0]
# [ 0  1  0  1  7  0  0  0  6]
# [ 0  1  0  2  1 60  2  0  3]
```

```
# [ 0 0 0 1 0 2 48 0 0]
# [ 0 0 0 1 0 0 0 10 0]
# [ 6 12 1 1 4 2 0 0 17]]
# F1 score: 0.741553
# Accuracy score: 0.784431

# Confusion Matrix
array= confusion_matrix(actuals, predictions)
df_cm = pd.DataFrame(array, index = [i for i in skin_list],
                      columns = [i for i in skin_list])
f, ax = plt.subplots(figsize=(13, 10))
sn.heatmap(df_cm, annot=True, square=True, linewidth=0.5, fmt="d", cmap="OrRd")
ax.set_ylimits((9,0))
plt.tight_layout()
plt.savefig('cm_data_set_exp1.png' ,format='png', dpi=100)

print(skm.classification_report(actuals, predictions))

# Results
#          precision    recall  f1-score   support
#
#           0       0.68      0.65      0.67       20
#           1       0.80      0.87      0.83       84
#           2       0.86      0.82      0.84       22
#           3       0.70      0.84      0.76       19
#           4       0.50      0.47      0.48       15
#           5       0.86      0.87      0.86       69
#           6       0.94      0.94      0.94       51
#           7       0.77      0.91      0.83       11
#           8       0.53      0.40      0.45       43
#
#    accuracy                           0.78      334
#   macro avg       0.74      0.75      0.74      334
# weighted avg       0.78      0.78      0.78      334

# ROC Curves
def plot_roc(true, predictions, class_name, save_path=None):
    '''
    Function to plot roc curve
    - inputs
        true: label true data
        predictions: model predictitons for the input
    
```

```

class_name: name of the analysis class
save_path: will save if there is a valid path
'',
fpr, tpr, thresholds = metrics.roc_curve(true,
                                         predictions,
                                         pos_label=1)
print(' - ' * 10)
print('Classe: ', str(class_name))

auc = "% .2f" % metrics.auc(fpr, tpr)
# title = 'ROC Curve, AUC = ' + str(auc) + ' for ' + class_name
title = 'ROC Curve for ' + class_name + ' with UAC = ' + str(auc)
plt.rcParams['axes.facecolor'] = '#FFF9EF'
with plt.style.context('seaborn-paper')):
    fig, ax = plt.subplots()
    ax.plot(fpr, tpr, color='#8B0000',
             lw=2,
             label='ROC curve for {}'.format(class_name))
    ax.plot([0, 1], [0, 1], 'k--', label='Baseline')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend(loc='lower right')
    plt.title(title)
    plt.tight_layout()
if save_path:
    plt.savefig(save_path +
                '/roc_curve_data_set_2{}.png'.format(class_name), format='png')

return fig

def test_class_probabilities(model, device, test_loader, which_class):
    model.eval()
    actuals = []
    probabilities = []
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            prediction = output.argmax(dim=1, keepdim=True)

```

```
actuals.extend(target.view_as(prediction) ==
               torch.from_numpy(np.array(which_class)))
probabilities.extend(np.exp(output[:, 
               torch.from_numpy(np.array(which_class))].cpu()))
return [i.item() for i in actuals], [i.item() for i in probabilities]

true = []
predicton = []
for i, class_name in enumerate(skin_list):
    actuals, class_probabilities = test_class_probabilities(model_ft, 'cuda',
        testloader, i)
    true.append(actuals)
    predicton.append(class_probabilities)

for i, skin_lesion in enumerate(skin_list):
    true_label = np.multiply(np.array(true[i]), 1)
    pred_label = np.multiply(np.array(predicton[i]), 1)
    plot_roc(true_label.round(), pred_label.round(),
             skin_lesion, '/content/gdrive/My
             Drive/UnB/TCC-1/TCC-1-dataset/imagens_exp6')
```

APÊNDICE G – Métricas de avaliação para o melhor experimento

Resultados G.1 - Métricas de avaliação para o melhor experimento

Tabela 10 – Reporte das métricas para cada lesão

| Tipo de lesão | Precisão | Recall | F1 Score | Support |
|---------------------------|-----------------|---------------|-----------------|----------------|
| Actinic Keratosis | 0.68 | 0.65 | 0.67 | 20 |
| Basal Cell Carcinoma | 0.80 | 0.87 | 0.83 | 84 |
| Melanocytic Nevus | 0.94 | 0.94 | 0.94 | 51 |
| Squamous Cell Carcinoma | 0.53 | 0.40 | 0.45 | 43 |
| Intraepithelial Carcinoma | 0.50 | 0.47 | 0.48 | 15 |
| Pyogenic Granuloma | 0.77 | 0.91 | 0.83 | 11 |
| Haemangioma | 0.70 | 0.84 | 0.76 | 19 |
| Dermatofibroma | 0.86 | 0.82 | 0.84 | 22 |
| Malignant Melanoma | 0.86 | 0.87 | 0.86 | 69 |
| Media/total | 0.78 | 0.78 | 0.78 | 334 |

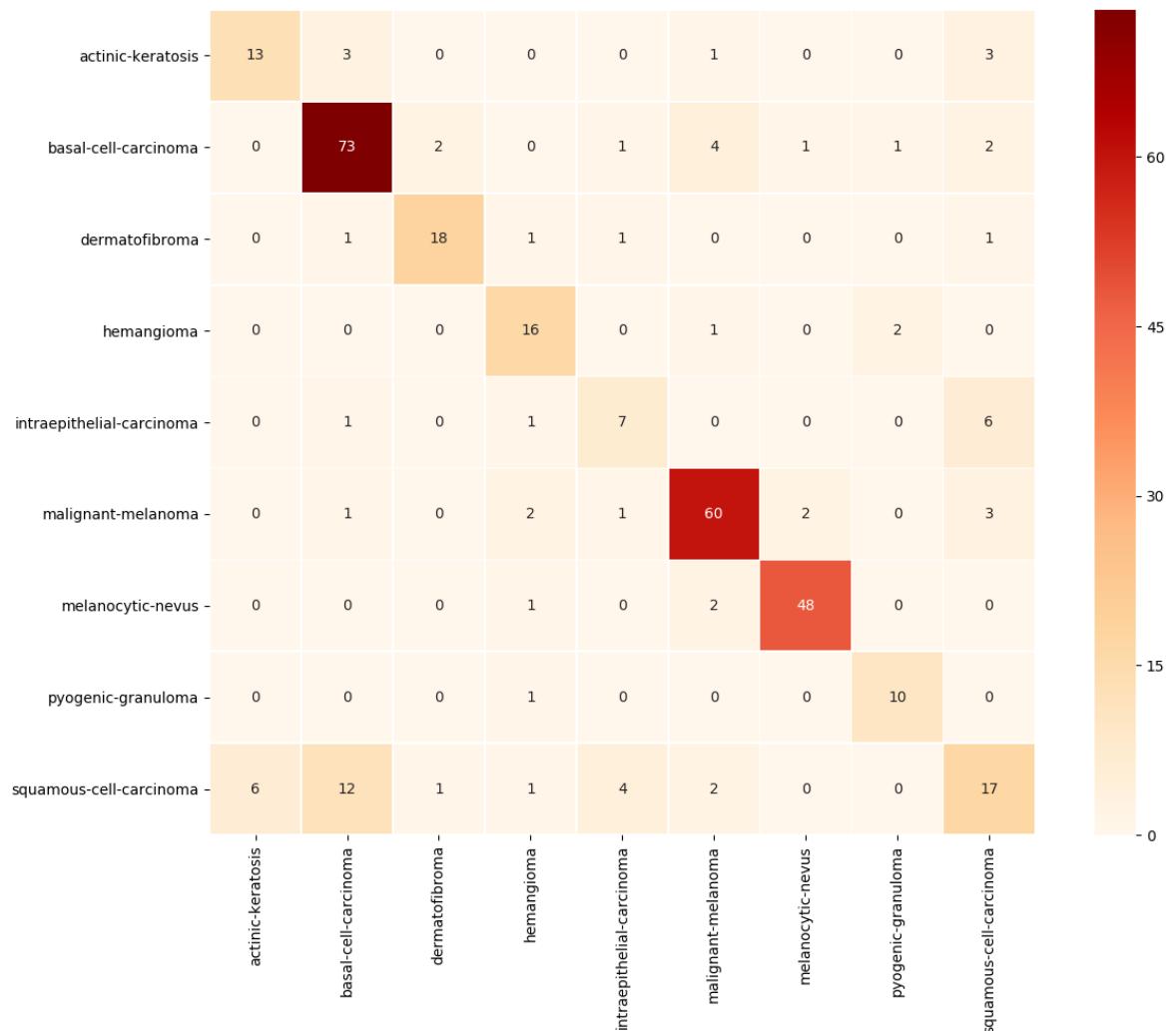


Figura 18 – Matriz de confusão do modelo para as 9 lesões
Autor

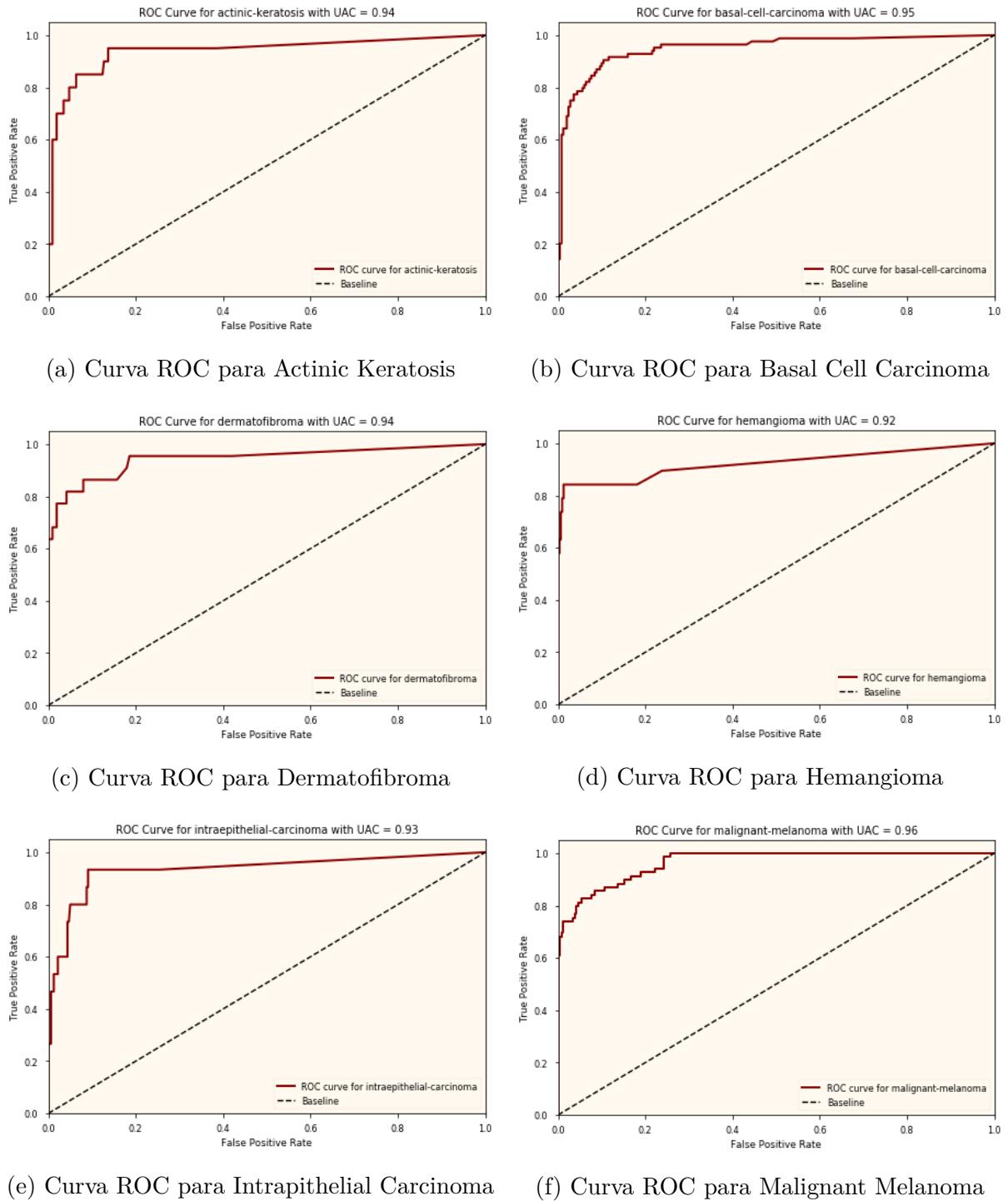


Figura 19 – Curva ROC para o restante das lesões. Continuação 1/2.

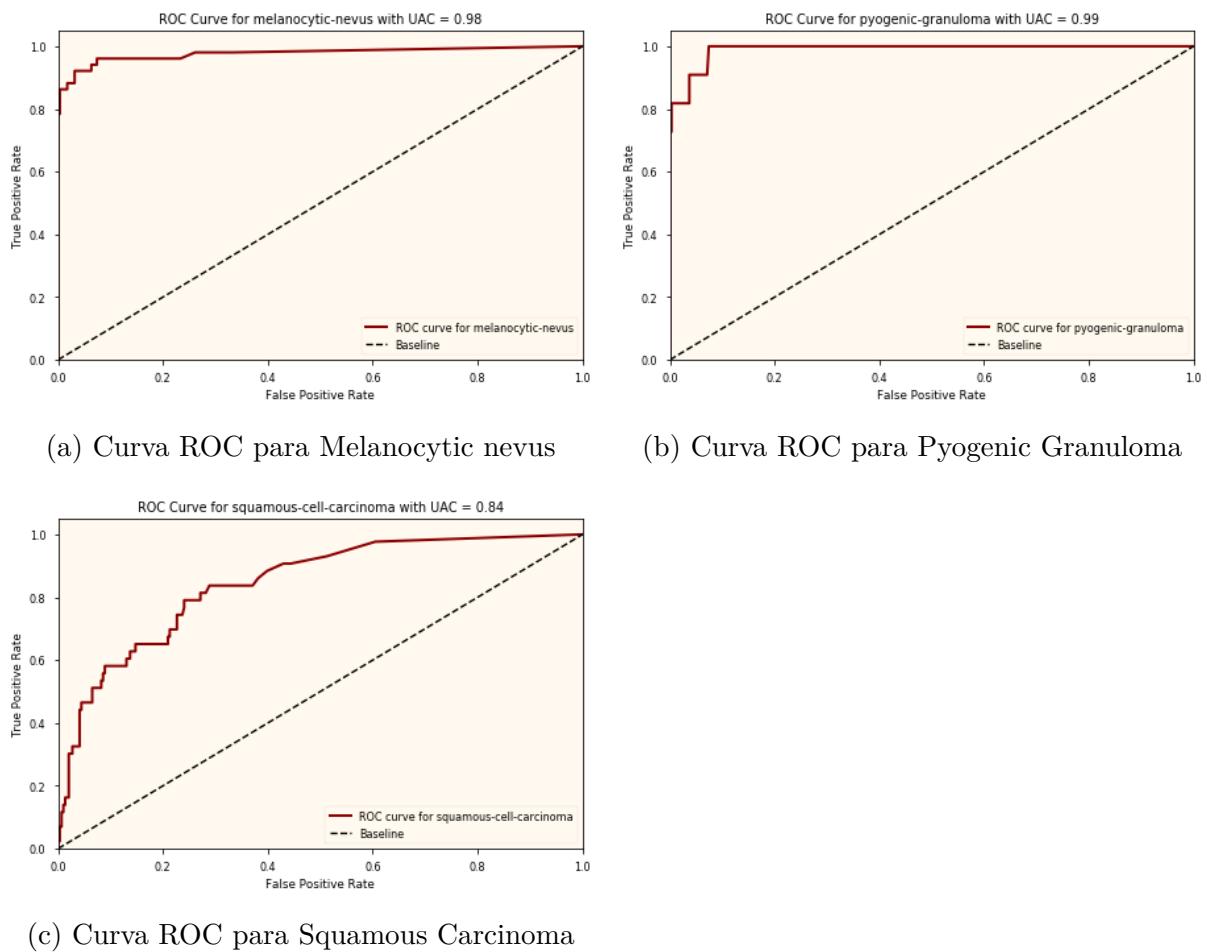


Figura 20 – Curva ROC para o restante das lesões. Continuação 2/2.