



Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Electronics Engineering

Skin Lesions Classification Using Convolutional Neural Networks in Clinical Images

Author: João Vitor Rodrigues Baptista
Advisor: Dr. Nilton Correia da Silva

Brasília, DF
2019



João Vitor Rodrigues Baptista

Skin Lesions Classification Using Convolutional Neural Networks in Clinical Images

Work submitted to the undergraduate course Electronics Engineering of Brasília, as a partial requirement to obtain a Software Engineer Bachelor's Degree Electronics Engineering.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Nilton Correia da Silva

Brasília, DF

2019

João Vitor Rodrigues Baptista

Skin Lesions Classification Using Convolutional Neural Networks in Clinical
Images/ João Vitor Rodrigues Baptista. – Brasília, DF, 2019-
119 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Nilton Correia da Silva

Work submitted to the undergraduate course – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2019.

1. Neural Networks. 2. Skin Lesion. I. Dr. Nilton Correia da Silva. II. Uni-
versidade de Brasília. III. Faculdade UnB Gama. IV. Skin Lesions Classification
Using Convolutional Neural Networks in Clinical Images

CDU 02:141:005.6

Errata

Elemento opcional da ABNT (2011, 4.2.1.2). Caso não deseje uma errata, deixar todo este arquivo em branco. Exemplo:

FERRIGNO, C. R. A. **Tratamento de neoplasias ósseas apendiculares com reimplantação de enxerto ósseo autólogo autoclavado associado ao plasma rico em plaquetas: estudo crítico na cirurgia de preservação de membro em cães.** 2011. 128 f. Tese (Livre-Docência) - Faculdade de Medicina Veterinária e Zootecnia, Universidade de São Paulo, São Paulo, 2011.

Folha	Linha	Onde se lê	Leia-se
1	10	auto-conclavo	autoconclavo

João Vitor Rodrigues Baptista

Skin Lesions Classification Using Convolutional Neural Networks in Clinical Images

Work submitted to the undergraduate course Electronics Engineering of Brasília, as a partial requirement to obtain a Software Engineer Bachelor's Degree Electronics Engineering.

Trabalho aprovado. Brasília, DF, 01 de junho de 2013:

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

A dedicatória é opcional. Caso não deseje uma, deixar todo este arquivo em branco.

*Dedico esta, assim como todas s minhas 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

A inclusão desta seção de agradecimentos é opcional, portanto, sua inclusão fica a critério do(s) autor(es), que caso deseje(em) fazê-lo deverá(ão) utilizar este espaço, seguindo a formatação de *espaço simples e fonte padrão do texto (arial ou times, tamanho 12 sem negritos, aspas ou itálico.*

Caso não deseje utilizar os agradecimentos, deixar toda este arquivo em branco.

A epígrafe é opcional. Caso não deseje uma, deixe todo este arquivo em branco.

*“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)*

Resumo

O resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (...) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto. O texto pode conter no mínimo 150 e no máximo 500 palavras, é aconselhável que sejam utilizadas 200 palavras. E não se separa o texto do resumo em parágrafos.

Palavras-chaves: latex. abntex. editoração de texto.

Abstract

This is the english abstract.

Key-words: latex, abnTeX, text editoration.

Listas de ilustrações

Figura 1 – Comparação entre o neurônio e o perceptron.	37
Figura 2 – Operação de <i>max-pooling</i> usando um filtro 2 x 2 e stide 2.	41
Figura 3 – Comparativo entre as topologias pela acurácia em função da operação em função do tamanho da base de dados	44
Figura 4 – Blocos de construção da ResNet	45
Figura 5 – Diagrama das lesões separadas por categorias	49
Figura 6 – Representação da pele humana sem lesão	52
Figura 7 – Lesões de interesse neste trabalho	54
Figura 8 – Imagem da base do ISIC da Basal cell carcinoma	55
Figura 9 – Diagrama de processos utilizados para a preparação da base de dados	59
Figura 10 – Detalhes técnicos da infraestrutura	65
Figura 11 – Escopo da área XAI	70
Figura 12 – O panorama geral de interpretabilidade para modelos de machine learning	72
Figura 13 – Explicação usando LIME para as top 3 classes na classificação de uma imagem feita pelo modelo da Google Inception.	74
Figura 14 – Exemplos de visualização do gradiente das imagens	75
Figura 15 – Multiplicação da imagem de entrada com o gradiente de ativação res- petivo	76
Figura 16 – Representação dos filtros de cada camada convolucional de uma rede neural convolucional.	77
Figura 17 – Exemplo de mapa de calor usando a técnica do GradCAM	78
Figura 18 – Matriz de confusão do modelo para as 9 lesões	112
Figura 19 – Curva ROC para o restante das lesões. Continuação 1/2.	113
Figura 20 – Curva ROC para o restante das lesões. Continuação 2/2.	114

Lista de tabelas

Tabela 1 – Numero de amostras da base do MED-NODE	46
Tabela 2 – Numero de amostras da base do Edinburgh	46
Tabela 3 – Numero de amostras da base do ISIC	47
Tabela 4 – Numero de amostras da base do Atlas	48
Tabela 5 – Transformações aplicadas no processo de <i>data augmentation</i>	58
Tabela 6 – Numero de amostras da base agregada	60
Tabela 7 – Numero de amostras finais	61
Tabela 8 – Comparativo entre trabalhos correlatos as AUC	68
Tabela 9 – Reporte das métricas para cada lesão	111

Lista de abreviaturas e siglas

Fig. Area of the i^{th} component

456 Isto é um número

123 Isto é outro número

lauro cesar este é o meu nome

List of symbols

Γ Greek letter Gamma

Λ Lambda

ζ Greek letter minuscule zeta

\in Pertains

Sumário

Introdução	29
I ASPECTOS GERAIS	31
1 ASPECTOS GERAIS	33
1.1 Composição e estrutura do trabalho	33
1.2 Considerações sobre formatação básica do relatório	34
1.2.1 Tipo de papel, fonte e margens	34
1.2.2 Numeração de Páginas	35
1.2.3 Espaços e alinhamento	35
1.2.4 Quebra de Capítulos e Aproveitamento de Páginas	35
1.3 Cópias	36
2 REFERENCIAL TEÓRICO	37
2.1 Redes Neurais	37
2.1.0.1 Breve histórico	38
2.1.0.2 Rede Neural Convolucional	38
2.1.0.2.1 Camada convolucional	39
2.1.0.2.2 Camada de <i>pooling</i>	40
2.1.0.2.3 Camada <i>fully-connected</i>	41
2.1.0.2.4 Funções de ativação	41
2.2 Treinamento da rede neural	42
2.3 Avaliação da rede neural	42
2.3.1 Métricas de treino e validação	43
2.3.2 Métricas para teste	43
2.4 Arquitetura de rede neural artificial	44
2.4.1 ResNet	45
2.5 Base de dados	45
2.5.1 MED-NODE	46
2.5.2 Edinburgh	46
2.5.3 ISIC	47
2.5.4 Atlas	47
2.6 Lesões de interesse	48
2.6.1 Actinic Keratosis	48
2.6.2 Basal cell carcinoma	50

2.6.3	Dermatofibroma	50
2.6.4	Hemangioma	51
2.6.5	Intraepithelial carcinoma	51
2.6.6	Malignant melanoma	51
2.6.7	Melanocytic nevus	52
2.6.8	Pyogenic Granuloma	52
2.6.9	Squamous cell carcinoma	53
2.7	Natureza da base de dados	53
2.7.1	Dificuldades	53
2.7.2	Amostra de dados	56
2.7.3	Rótulos	56
2.8	Preparação das amostras	56
2.8.1	Transfer Learning	57
2.8.2	Data augmentation	57
2.8.2.1	Metodos de Augmentation	58
2.8.2.1.1	Transformações	58
2.9	Preparação da base de dados	58

II	TEXTO E PÓS TEXTO	63
3	RESULTADOS PRÉVIOS	65
3.1	Infraestrutura	65
3.2	Classificação	66
3.2.1	Processo de treinamento	66
3.2.2	Parâmetros livres	66
3.2.3	Resultados	67
4	PRÓXIMOS PASSOS	69
4.1	Interpretabilidade e Explicabilidade	69
4.1.1	Conceitos	70
4.2	Métodos de interpretabilidade	71
4.2.1	Método Model-agnostic	71
4.2.1.1	Perturbation-based	73
4.2.1.2	LIME	73
4.2.1.3	Gradient-based	75
4.2.1.4	GradCAM	76
	REFERÊNCIAS	79

APÊNDICES	81
APÊNDICE A – WEB SCRAPING	83
APÊNDICE B – CONVERSÃO DAS IMAGENS E REDIMENSIONAMENTO PRÉVIO	85
APÊNDICE C – SEPARAÇÃO DA BASE DE FORMA ESTRATIFICADA	87
APÊNDICE D – DATA AUGMENTATION	89
APÊNDICE E – TREINAMENTO E VALIDAÇÃO DA REDE	93
APÊNDICE F – TESTE E CRIAÇÃO DAS MÉTRICAS	101
APÊNDICE G – MÉTRICAS DE AVALIAÇÃO PARA O MELHOR EXPERIMENTO	111
ANEXOS	115
ANEXO A – PRIMEIRO ANEXO	117
ANEXO B – SEGUNDO ANEXO	119

Introdução

Este documento apresenta considerações gerais e preliminares relacionadas à redação de relatórios de Projeto de Graduação da Faculdade UnB Gama (FGA). São abordados os diferentes aspectos sobre a estrutura do trabalho, uso de programas de auxílio a edição, tiragem de cópias, encadernação, etc.

Parte I

Aspectos Gerais

1 Aspectos Gerais

Estas instruções apresentam um conjunto mínimo de exigências necessárias a uniformidade de apresentação do relatório de Trabalho de Conclusão de Curso da FGA. Estilo, concisão e clareza ficam inteiramente sob a responsabilidade do(s) aluno(s) autor(es) do relatório.

As disciplinas de Trabalho de Conclusão de Curso (TCC) 01 e Trabalho de Conclusão de Curso (TCC) 02 se desenvolvem de acordo com Regulamento próprio aprovado pelo Colegiado da FGA. Os alunos matriculados nessas disciplinas devem estar plenamente cientes de tal Regulamento.

1.1 Composição e estrutura do trabalho

A formatação do trabalho como um todo considera três elementos principais: (1) pré-textuais, (2) textuais e (3) pós-textuais. Cada um destes, pode se subdividir em outros elementos formando a estrutura global do trabalho, conforme abaixo (as entradas itálico são *opcionais*; em itálico e negrito são **essenciais**):

Pré-textuais

- Capa
- Folha de rosto
- *Dedicatória*
- *Agradecimentos*
- *Epígrafe*
- Resumo
- Abstract
- Lista de figuras
- Lista de tabelas
- Lista de símbolos e
- Sumário

Textuais

- *Introdução*

- *Desenvolvimento*
- *Conclusões*

Pós-Textuais

- Referências bibliográficas
- *Bibliografia*
- Anexos
- Contracapa

Os aspectos específicos da formatação de cada uma dessas três partes principais do relatório são tratados nos capítulos e seções seguintes.

No modelo L^AT_EX, os arquivos correspondentes a estas estruturas que devem ser editados manualmente estão na pasta **editáveis**. Os arquivos da pasta **fixos** tratam os elementos que não necessitam de edição direta, e devem ser deixados como estão na grande maioria dos casos.

1.2 Considerações sobre formatação básica do relatório

A seguir são apresentadas as orientações básicas sobre a formatação do documento. O modelo L^AT_EX já configura todas estas opções corretamente, de modo que para os usuários deste modelo o texto a seguir é meramente informativo.

1.2.1 Tipo de papel, fonte e margens

Papel - Na confecção do relatório deverá ser empregado papel branco no formato padrão A4 (21 cm x 29,7cm), com 75 a 90 g/m².

Fonte – Deve-se utilizar as fontes Arial ou Times New Roman no tamanho 12 pra corpo do texto, com variações para tamanho 10 permitidas para a wpaginação, legendas e notas de rodapé. Em citações diretas de mais de três linhas utilizar a fonte tamanho 10, sem itálicos, negritos ou aspas. Os tipos itálicos são usados para nomes científicos e expressões estrangeiras, exceto expressões latinas.

Margens - As margens delimitando a região na qual todo o texto deverá estar contido serão as seguintes:

- Esquerda: 03 cm;
- Direita : 02 cm;

- Superior: 03 cm;
- Inferior: 02 cm.

1.2.2 Numeração de Páginas

A contagem sequencial para a numeração de páginas começa a partir da primeira folha do trabalho que é a Folha de Rosto, contudo a numeração em si só deve ser iniciada a partir da primeira folha dos elementos textuais. Assim, as páginas dos elementos pré-textuais contam, mas não são numeradas e os números de página aparecem a partir da primeira folha dos elementos textuais que é a Introdução.

Os números devem estar em algarismos arábicos (fonte Times ou Arial 10) no canto superior direito da folha, a 02 cm da borda superior, sem traços, pontos ou parênteses.

A paginação de Apêndices e Anexos deve ser contínua, dando seguimento ao texto principal.

1.2.3 Espaços e alinhamento

Para a monografia de TCC 01 e 02 o espaço entrelinhas do corpo do texto deve ser de 1,5 cm, exceto RESUMO, CITAÇÕES de mais de três linhas, NOTAS de rodapé, LEGENDAS e REFERÊNCIAS que devem possuir espaçamento simples. Ainda, ao se iniciar a primeira linha de cada novo parágrafo se deve tabular a distância de 1,25 cm da margem esquerda.

Quanto aos títulos das seções primárias da monografia, estes devem começar na parte superior da folha e separados do texto que o sucede, por um espaço de 1,5 cm entrelinhas, assim como os títulos das seções secundárias, terciárias.

A formatação de alinhamento deve ser justificado, de modo que o texto fique alinhado uniformemente ao longo das margens esquerda e direita, exceto para CITAÇÕES de mais de três linhas que devem ser alinhadas a 04 cm da margem esquerda e REFERÊNCIAS que são alinhadas somente à margem esquerda do texto diferenciando cada referência.

1.2.4 Quebra de Capítulos e Aproveitamento de Páginas

Cada seção ou capítulo deverá começar numa nova pagina (recomenda-se que para texto muito longos o autor divida seu documento em mais de um arquivo eletrônico).

Caso a última pagina de um capítulo tenha apenas um número reduzido de linhas (digamos 2 ou 3), verificar a possibilidade de modificar o texto (sem prejuízo do conteúdo e obedecendo as normas aqui colocadas) para evitar a ocorrência de uma página pouco aproveitada.

Ainda com respeito ao preenchimento das páginas, este deve ser otimizado, evitando-se espaços vazios desnecessários.

Caso as dimensões de uma figura ou tabela impeçam que a mesma seja posicionada ao final de uma página, o deslocamento para a página seguinte não deve acarretar um vazio na pagina anterior. Para evitar tal ocorrência, deve-se re-posicionar os blocos de texto para o preenchimento de vazios.

Tabelas e figuras devem, sempre que possível, utilizar o espaço disponível da página evitando-se a “quebra” da figura ou tabela.

1.3 Cópias

Nas versões do relatório para revisão da Banca Examinadora em TCC1 e TCC2, o aluno deve apresentar na Secretaria da FGA, uma cópia para cada membro da Banca Examinadora.

Após a aprovação em TCC2, o aluno deverá obrigatoriamente apresentar a versão final de seu trabalho à Secretaria da FGA na seguinte forma:

01 cópia encadernada para arquivo na FGA;

01 cópia não encadernada (folhas avulsas) para arquivo na FGA;

01 cópia em CD de todos os arquivos empregados no trabalho;

A cópia em CD deve conter, além do texto, todos os arquivos dos quais se originaram os gráficos (excel, etc.) e figuras (jpg, bmp, gif, etc.) contidos no trabalho. Caso o trabalho tenha gerado códigos fontes e arquivos para aplicações específicas (programas em Fortran, C, Matlab, etc.) estes deverão também ser gravados em CD.

O autor deverá certificar a não ocorrência de “vírus” no CD entregue a secretaria.

2 Referencial Teórico

A regra mais rígida com respeito a Introdução é que a mesma, que é necessariamente parte integrante do texto, não deverá fazer agradecimentos a pessoas ou instituições nem comentários pessoais do autor atinentes à escolha ou à relevância do tema.

2.1 Redes Neurais

São algorítimos no campo de aprendizado de máquina 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.

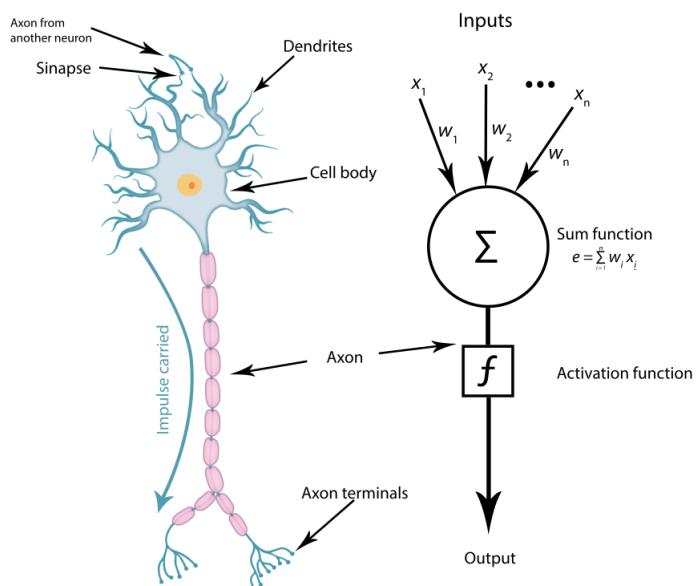


Figura 1 – Comparação entre o neurônio e o perceptron.
[**CITAÇÃO DANILO**] (BORDALO; FERZIGER; KLINE, 1989)

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, 1943). 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, 1982).

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 and 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.

$$\frac{d\mathbf{C}}{dw} = \frac{du}{dw} \cdot \mathbf{F}_u + \frac{dv}{dw} \cdot \mathbf{F}_v$$

(2.1)

Onde na “Eq. (2.1.0.2.1)” 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, 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 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 “Eq. (2.1.0.2.1)”

$$\frac{d\mathbf{C}}{dw} = \frac{du}{dw} \cdot \mathbf{F}_u + \frac{dv}{dw} \cdot \mathbf{F}_v$$

(2.2)

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

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, 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-polling* (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 da “Eq. (2.1.0.2.2)” é possível analisar a dimensão de saída.

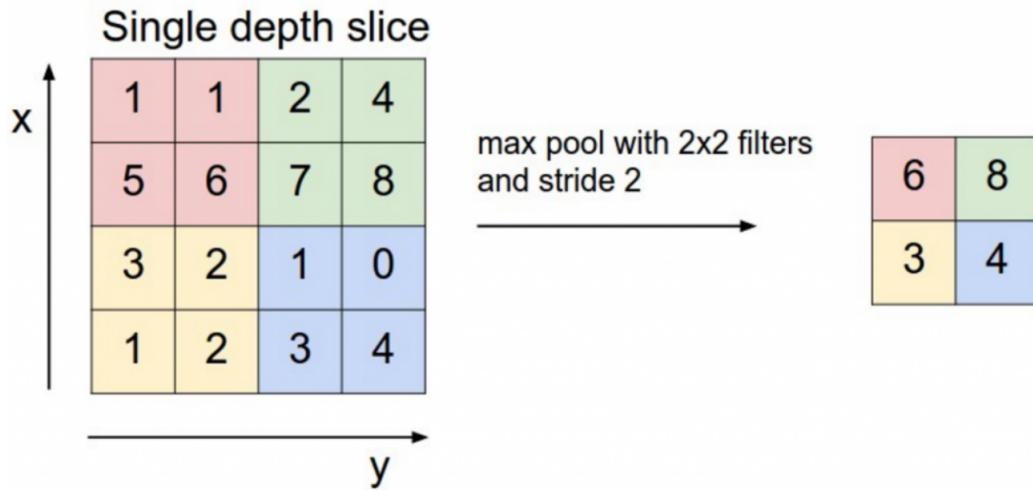


Figura 2 – Operação de *max-pooling* usando um filtro 2 x 2 e stide 2.
Karpathy (2018a)

$$\frac{d\mathbf{C}}{dw} = \frac{du}{dw} \cdot \mathbf{F}_u + \frac{dv}{dw} \cdot \mathbf{F}_v$$

(2.3)

Na “Equação (2.1.0.2.1)” H' e 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 de papel de adicionar não-linearidade nas sobre entrada. Existem diversas funções, contudo a mais utilizada é a *ReLU* (*Rectifier Linear Unit*) “Eq. (2.1.0.2.2)” Devido ao fator de acelerar a convergência de otimizadores como *SGD* (*stochastic gradient descent*), comprador com outras funções.

$$\frac{dC}{dw} = \frac{du}{dw} \cdot \mathbf{F}_u + \frac{dv}{dw} \cdot \mathbf{F}_v \quad (2.4)$$

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, 1999).
- ***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, 1999).

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 quão 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 quão bom o modelo esta 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 “Eq. (2.3.1)”Onde tp são os verdadeiros positivos, tn são os verdadeiros negativos e s é total de amostras utilizadas

$$\frac{d\mathbf{C}}{dw} = \frac{du}{dw} \cdot \mathbf{F}_u + \frac{dv}{dw} \cdot \mathbf{F}_v \quad (2.5)$$

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. (2.3.2)”pode-se observado as equações para calculo da precisão e do recall.

$$\frac{dC}{dw} = \frac{du}{dw} \cdot \mathbf{F}_u + \frac{dv}{dw} \cdot \mathbf{F}_v \quad (2.6)$$

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 a analisa o

quão 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 cálculo do grau de certeza que a lesão será classificada como verdadeiro positivo e a UAC calcula o quão exato o diagnóstico.

2.4 Arquitetura de rede neural artificial

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

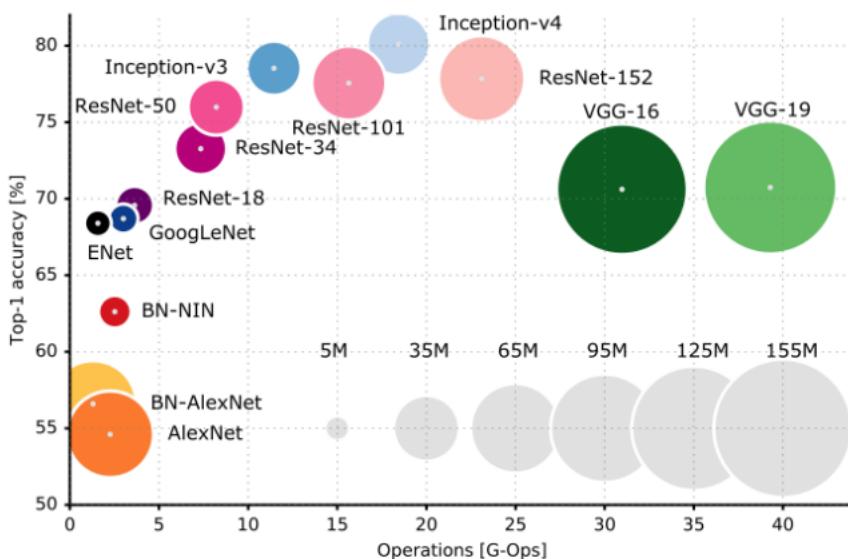


Figura 3 – Comparativo entre as topologias pela acurácia em função da operação em função do tamanho da base de dados

Canziani, Paszke and Culurciello (2016).

Existe diversas abordagens. Um exemplo, pode-se criar a topologia customizada do princípio, ao passo que, pode-se escolher topologias que já foram criadas e testadas por outros pesquisadores e que se provaram muito boas 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., 2015). Desda 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; GREGG; SCHMIDHUBER, 2015).

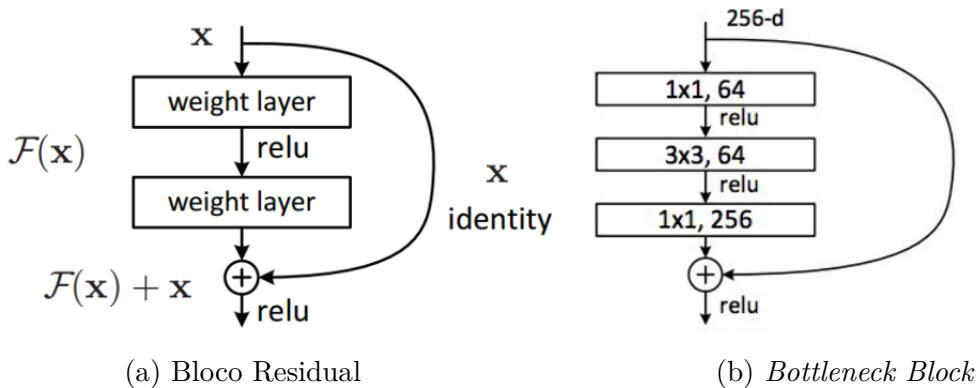


Figura 4 – Blocos de construção da ResNet

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 colatos como Seog 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 pele. Por essa razão a única restrição imposta ao montar a base é que devem ser imagens clinicas.

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 explicita e pode ser encontrada facilmente.¹

Tabela 1 – Numero 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 – Numero 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

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>>

2.5.3 ISIC

É um repositório de imagens de lesões de pele público chamado International Skin Imaging Collaboration (ISIC). 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 biópsias desnecessárias (International Society for Digital Imaging of the Skin, 2018)³. Esse repositório de imagens é organizado, imagens possuem *metadata* que contém 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.

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

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.danderm-pdv.is.kkh.dk/atlas/index.html>>, <<http://www.hellenicdermatlas.com/en/?params=en>>.

Tabela 4 – Número 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

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.

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).

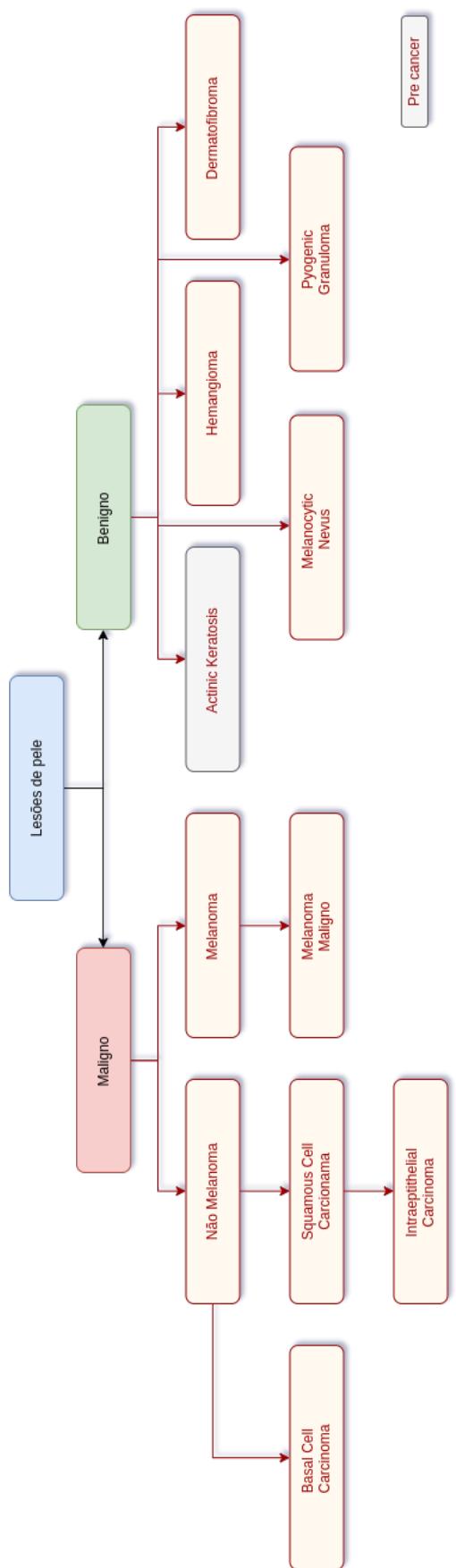


Figura 5 – Diagrama das lesões separadas por categorias
Autor

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 (American Cancer Society, 2016).

Essa é o tipo de câncer de pele mais comum, onde 80 % dos cânceres de pele são 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 (American Cancer Society, 2017).

Caso os tratamento não for efetivo ou apropriado esse tipo de lesão pode ser ocorrer na mesma região de pele (American Cancer Society, 2016). Estima-se que no Brasil em 2015 foram aproximante 1958 mortes por canceres não melanoma e 1794 mortes por casos de melanoma (BRASIL; Ministério da Saúde, 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. (BRASIL; 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) (BRASIL; 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 (BRASIL; 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 (BRASIL; SBD). 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 (BRASIL; SBD).

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; JAMES; BERGER, 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 (National Cancer Institute, 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 (American Cancer Society, 2018). No Brasil estima-se que 6360 novos casos em 2018 (Instituto Nacional de Câncer José Alencar Gomes da Silva, 2018).

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.

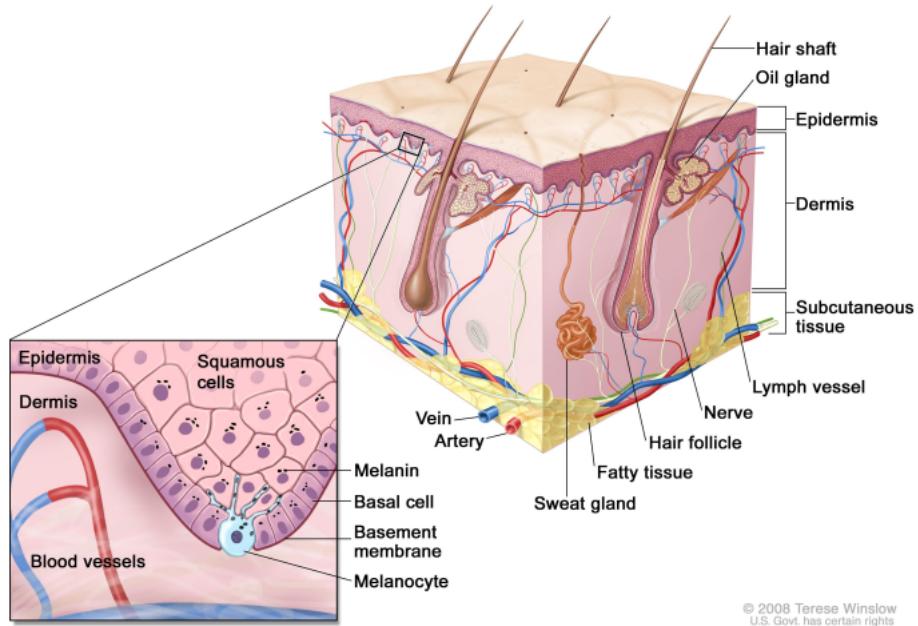


Figura 6 – Representação da pele humana sem lesão
National Cancer Institute (2018c).

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 (National Cancer Institute, 2018b).

2.6.8 Pyogenic Granuloma

Tumor benigno de lesão vascular relativamente comum na pele na mucosa, porem 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 cancer 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 7h. Normalmente esta é encontrada nas partes do corpo que são expostas ao sol, principalmente na cabeça e mãos (American Cancer Society, 2016).

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 (HOWELL; RAMSEY, 2017; MOY, 2000).

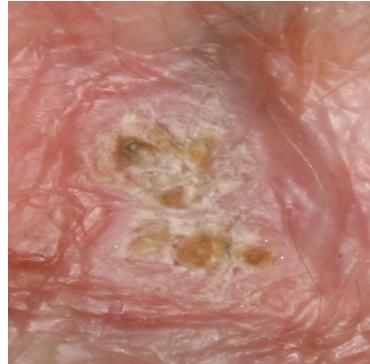
2.7 Natureza da base de dados

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 adiciona para suplementar as demais. A base do ISIC trás uma



(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

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.

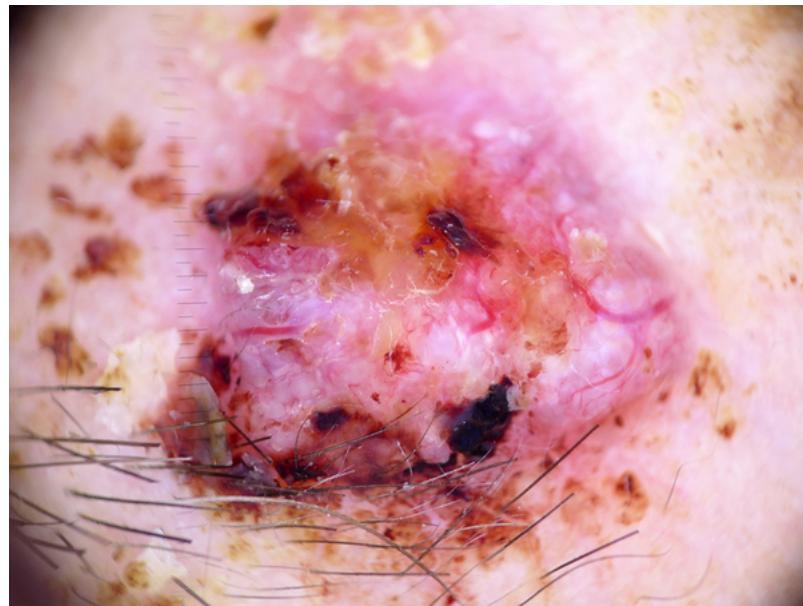


Figura 8 – Imagem da base do ISIC da Basal cell carcinoma
Base do ISIC

- **Etnia:** As diversas etnias 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 Seog 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.

- **Idade:** A idade pode influenciar na aparência que a lesão se apresenta. Pelos trabalho do Seog 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 está atrelado principalmente a habilidade para coletar a imagem. Por outro lado, a localização da lesão obriga a capturar outras partes do corpo, como por exemplo 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 fator que pode ser levado em consideração é a mudança de elasticidade da pele humana a medida que o paciente 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 número 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 consolidadas 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 garantiria 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 originais 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 princípio 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 indústria quanto nas pesquisas científicas. Isso impõe um grande obstáculo 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, porém 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, porém é alterado a última camada da rede, geralmente a camada *fully-connected*, para o número de classes da nova base de dados e então atualiza os parâmetros via *backpropagation*. Normalmente é mais demorado porém 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 do princípio 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 relacionados 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 transformações 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 uma única base. E foi feita uma verificação nas imagens para descartar qualquer

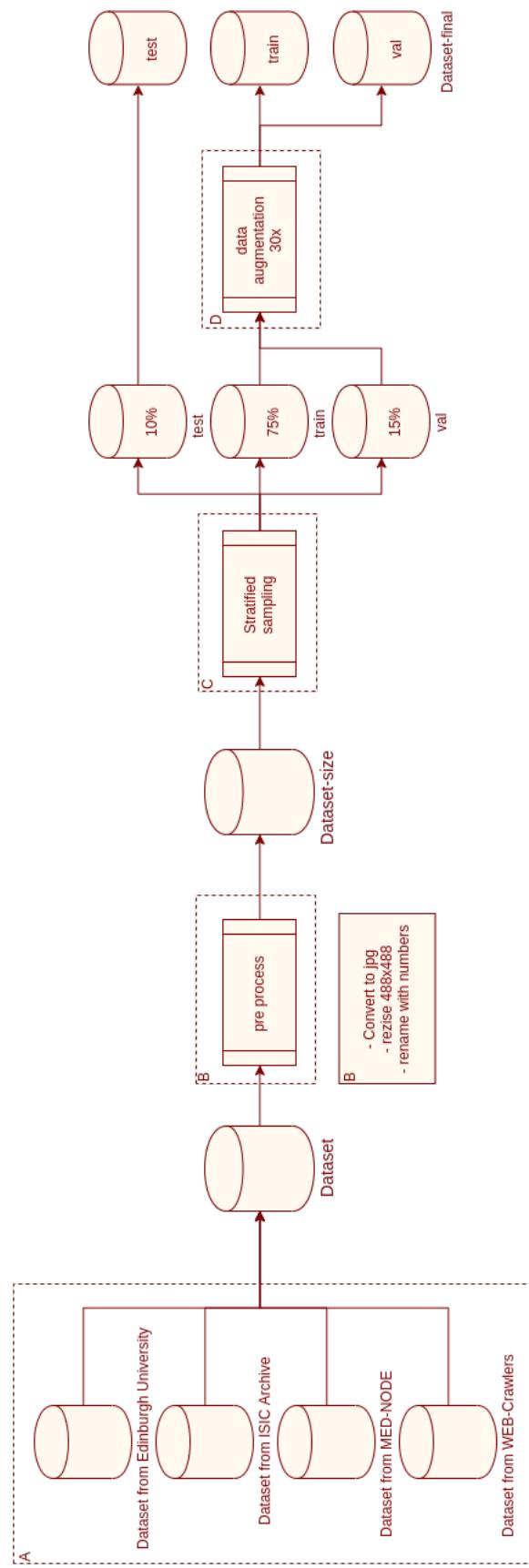


Figura 9 – Diagrama de processos utilizados para a preparação da base de dados
Autor

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](#).

Tabela 6 – Número 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 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](#).

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](#).

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

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

Parte II

Texto e Pós Texto

3 Resultados prévios

Esse espaço esta reservados para os 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 Google¹

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.

¹ Disponível em: <<https://colab.research.google.com/notebooks/welcome.ipynb>>

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 Seog 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***: Defino 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 9 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 2017)	Seog Han(2018)	Danilo(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, 1983; CLANCEY; SHORTLIFFE, 1984; CHANDRASEKARAN; TANNER; JOSEPHSON, 1989). A ideia é a dos dias atuais, sistemas que lidam com decisões sensíveis devem ser capaz de responder o "*Por quê ?*".

4.1.1 Conceitos

Biran and 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.

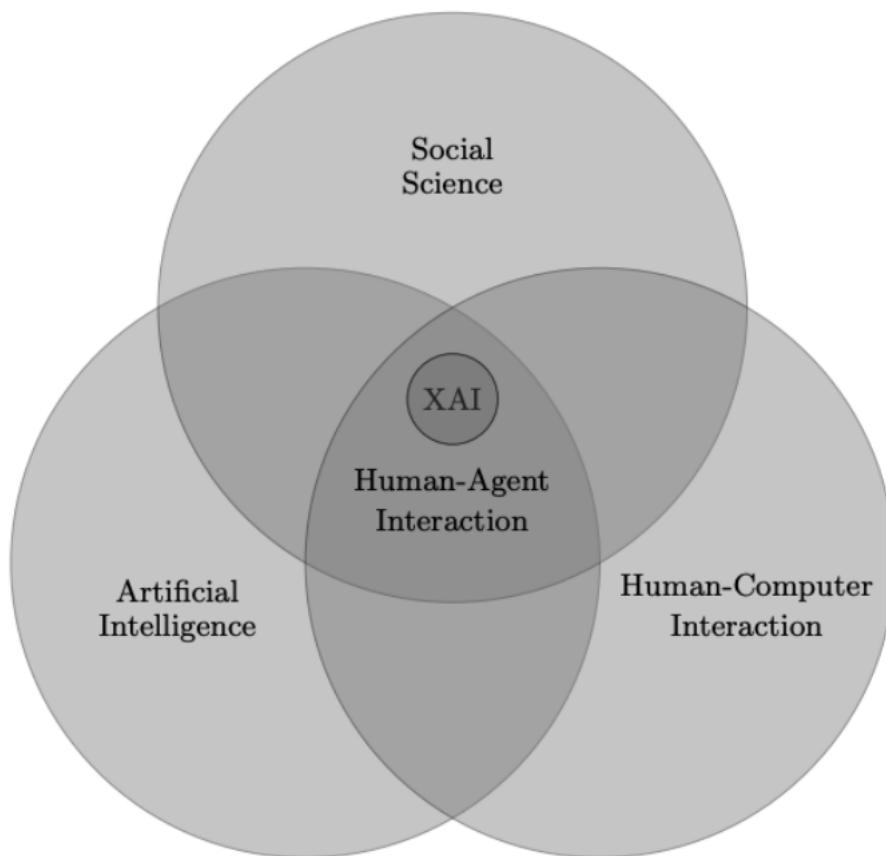


Figura 11 – Escopo da área XAI
Autor

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, 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

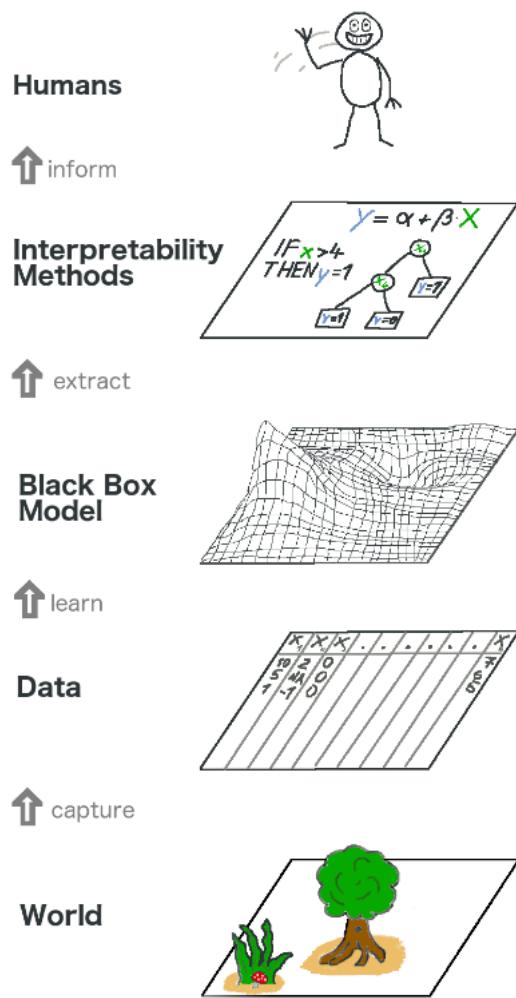


Figura 12 – O panorama geral de interpretabilidade para modelos de machine learning
Autor

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-ŠIKONJA; BOHANEC, 2018).

Brevemente, a abordagem baseadas em nos gradientes usa o cálculo dos gradien-

tes 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 abreviação de (*Local Interpretable Model-agnostic Explanations*) é uma metodologia apresentada por Ribeiro, Singh and Guestrin (2016a) que implementa um modelo local que explica previsões individuais. LIME usa o conceito de perturbação apresentado na subseção anterior para treinar um modelo interpretável. Esse modelo treinado deve possuir boa interpretabilidade local.

LIME pode ser expresso matematicamente pela Equação X. onde x é a instância 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 quanto perto a explicação está da previsão do modelo original f , em quanto a complexidade do modelo (g) é mantida baixa, G é a família de possíveis explicações.

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

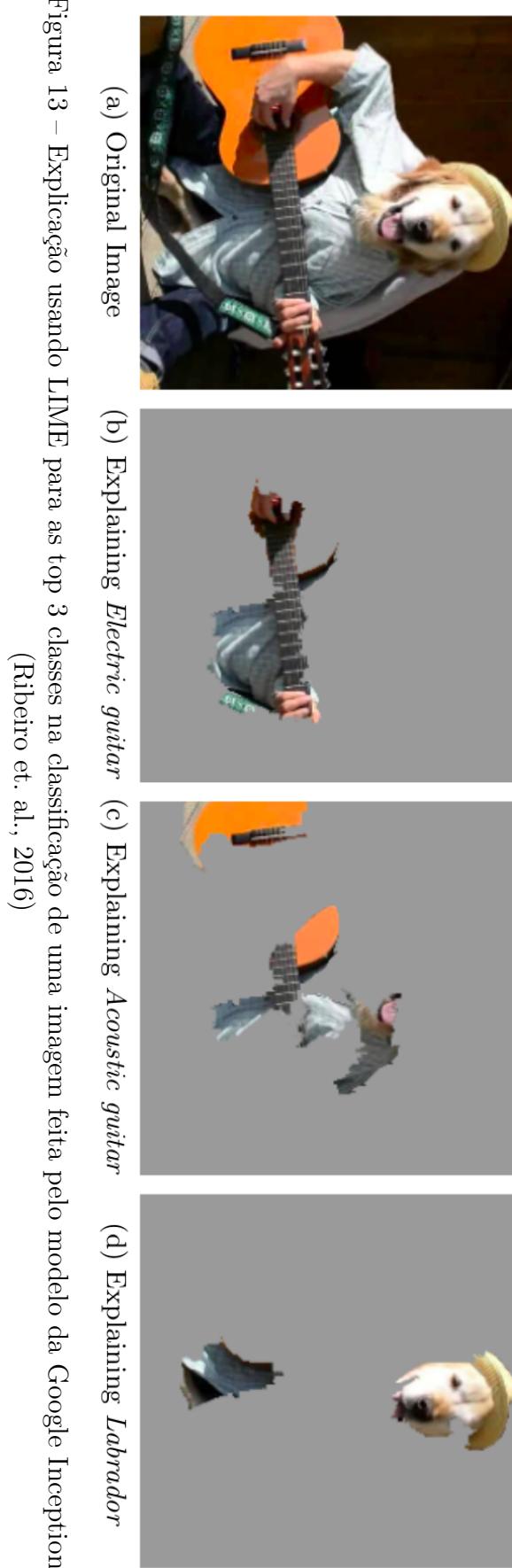


Figura 13 – Explicação usando LIME para as top 3 classes na classificação de uma imagem feita pelo modelo da Google Inception.
(Ribeiro et. al., 2016)

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; SUNDARARA- JAN; 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
Autor

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 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

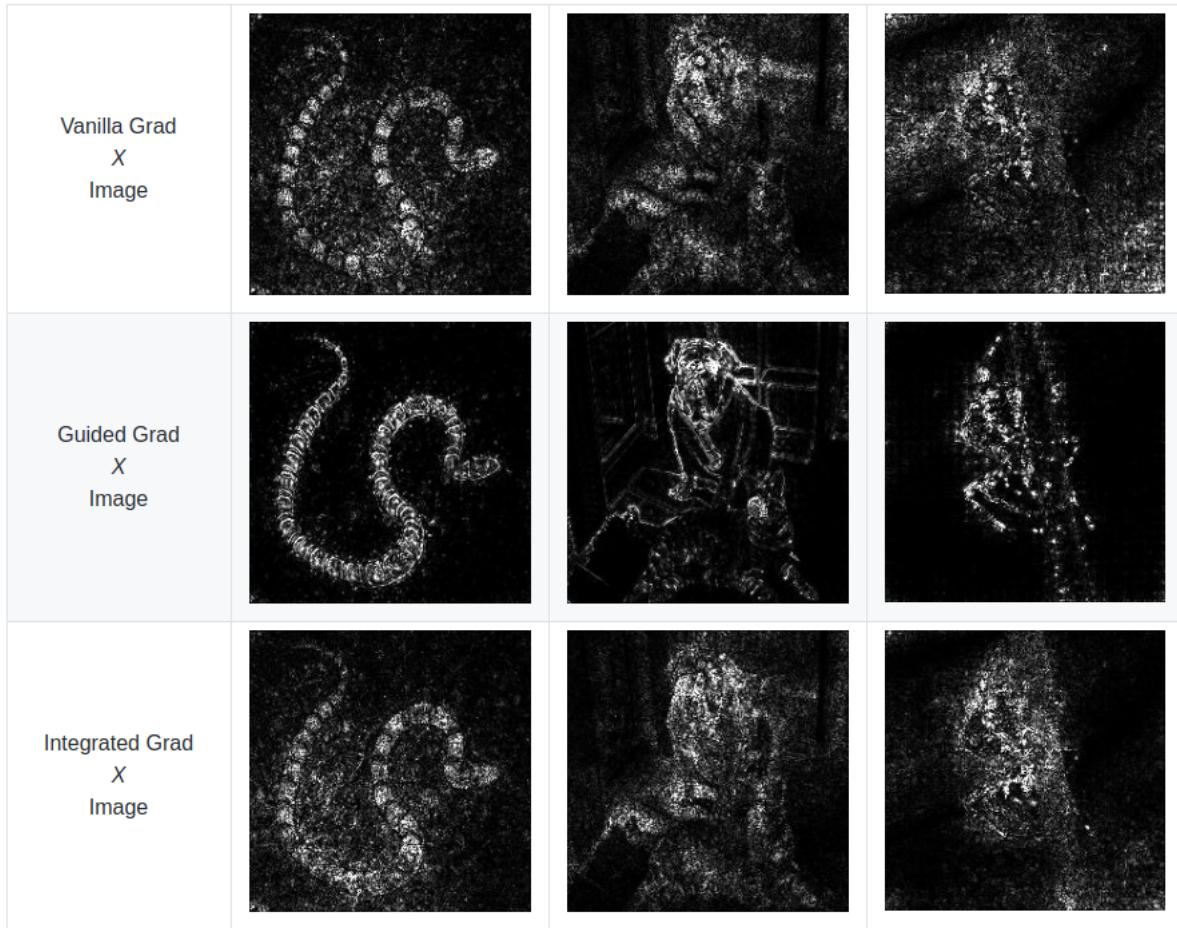


Figura 15 – Multiplicação da imagem de entrada com o gradiente de ativação respetivo Autor

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.

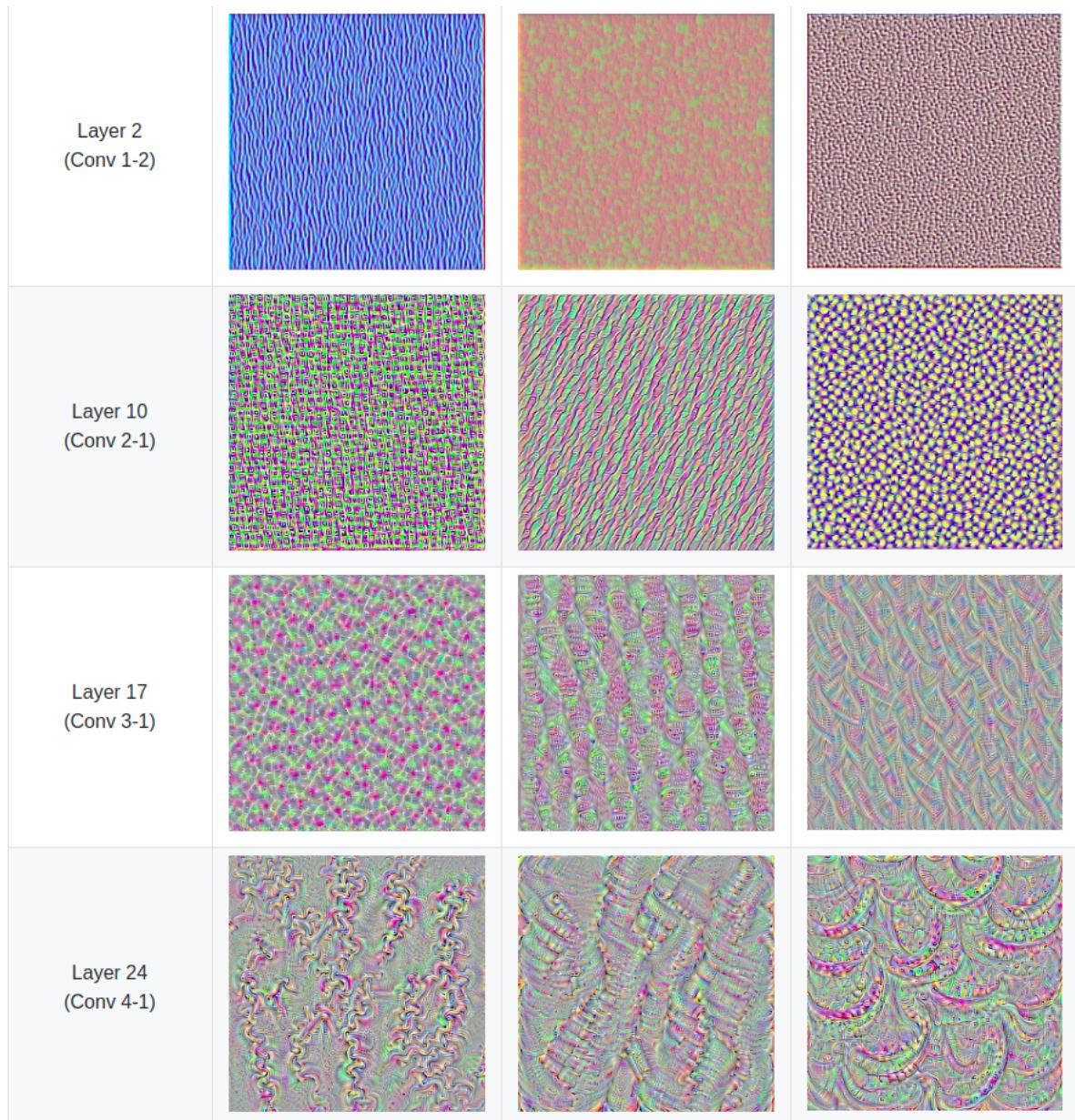


Figura 16 – Representação dos filtros de cada camada convolucional de uma rede neural convolucional.

Autor

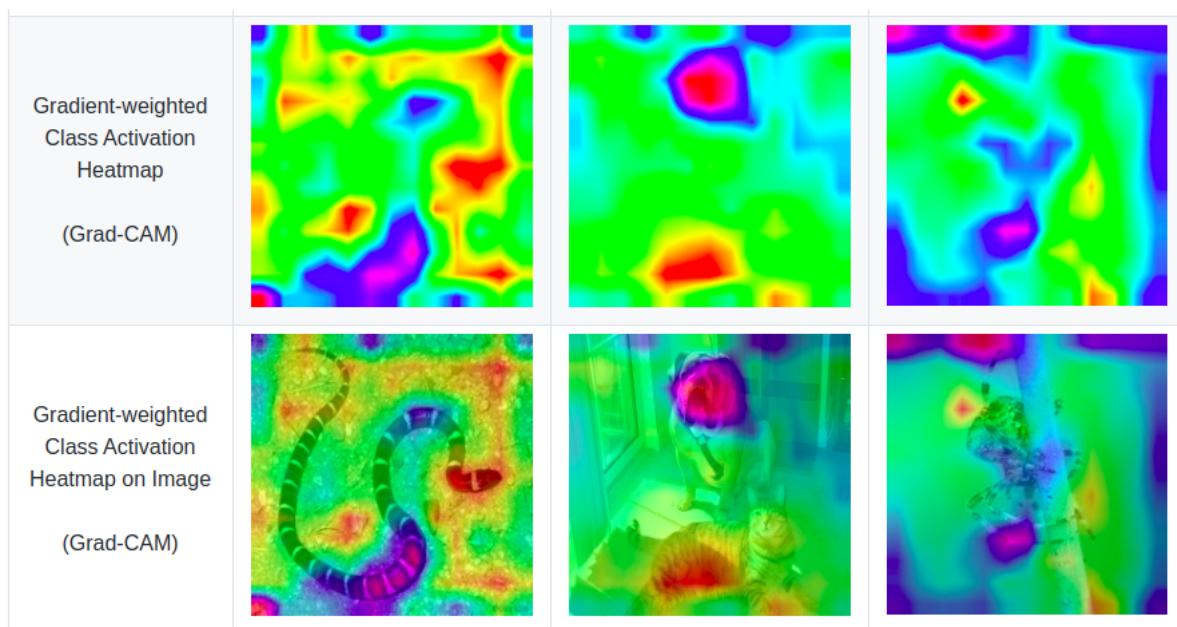


Figura 17 – Exemplo de mapa de calor usando a técnica do GradCAM
Autor

Referências

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 14724: Informação e documentação — trabalhos acadêmicos — apresentação*. Rio de Janeiro, 2011. 15 p. Citado na página [3](#).

BORDALO, S. N.; FERZIGER, J. H.; KLINE, S. J. The development of zonal models for turbulence. In: *Proceedings of the 10th Brazilian Congress of Mechanical Engineering*. [S.l.: s.n.], 1989. v. 1, p. 41–44. Citado na página [37](#).

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 images in base dataset and convert all to RGB  
'''  
  
import os, sys  
from PIL import Image  
  
# diretório /<pastas das lesões>  
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 images with increasing number, convert images to rgb and  
    resize all images to 448x448 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

# Hyperparamets
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-classifiers/113>

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.barh(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 Matriz
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 predictions 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 9 – 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

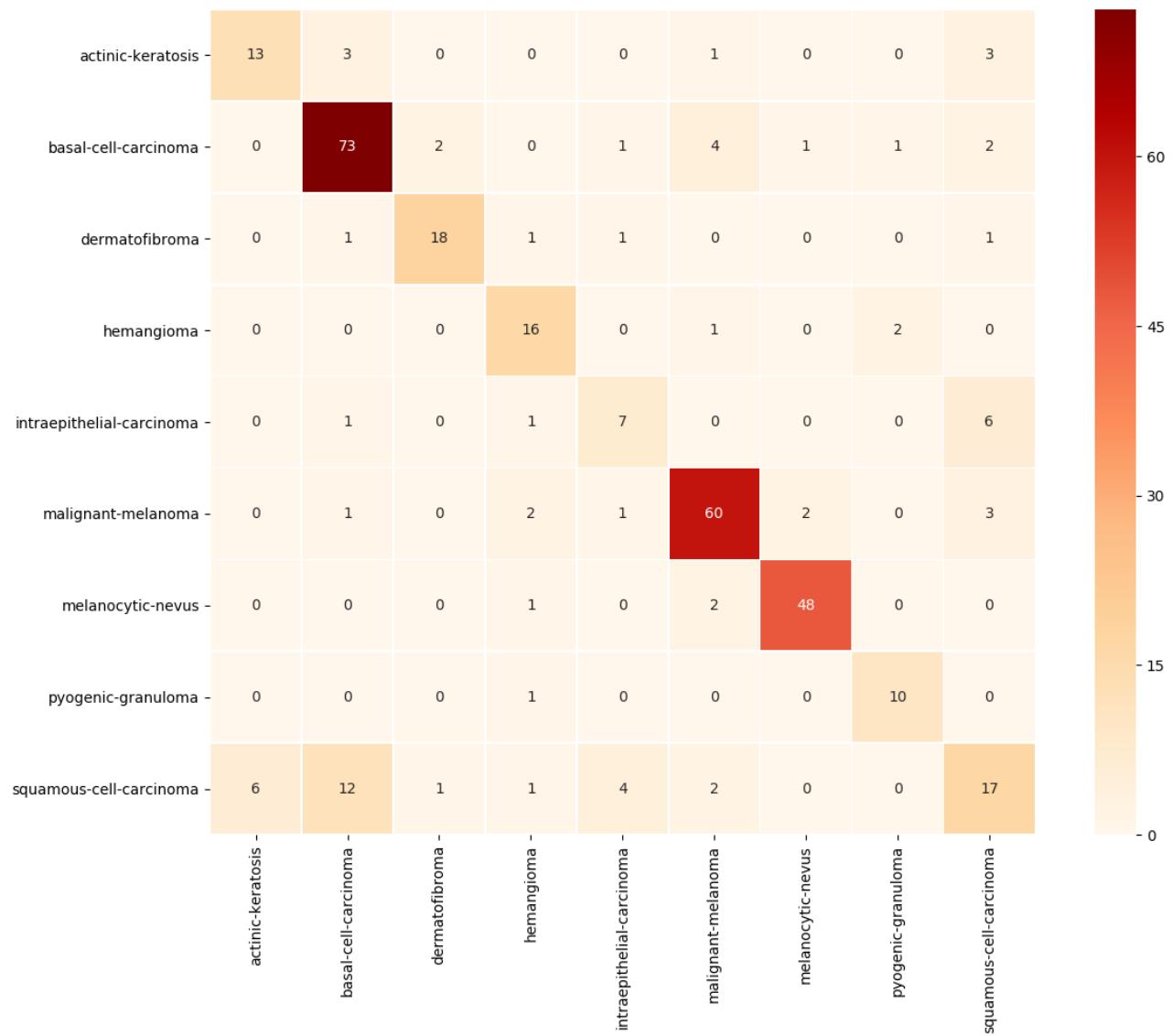


Figura 18 – Matriz de confusão do modelo para as 9 lesões
Autor

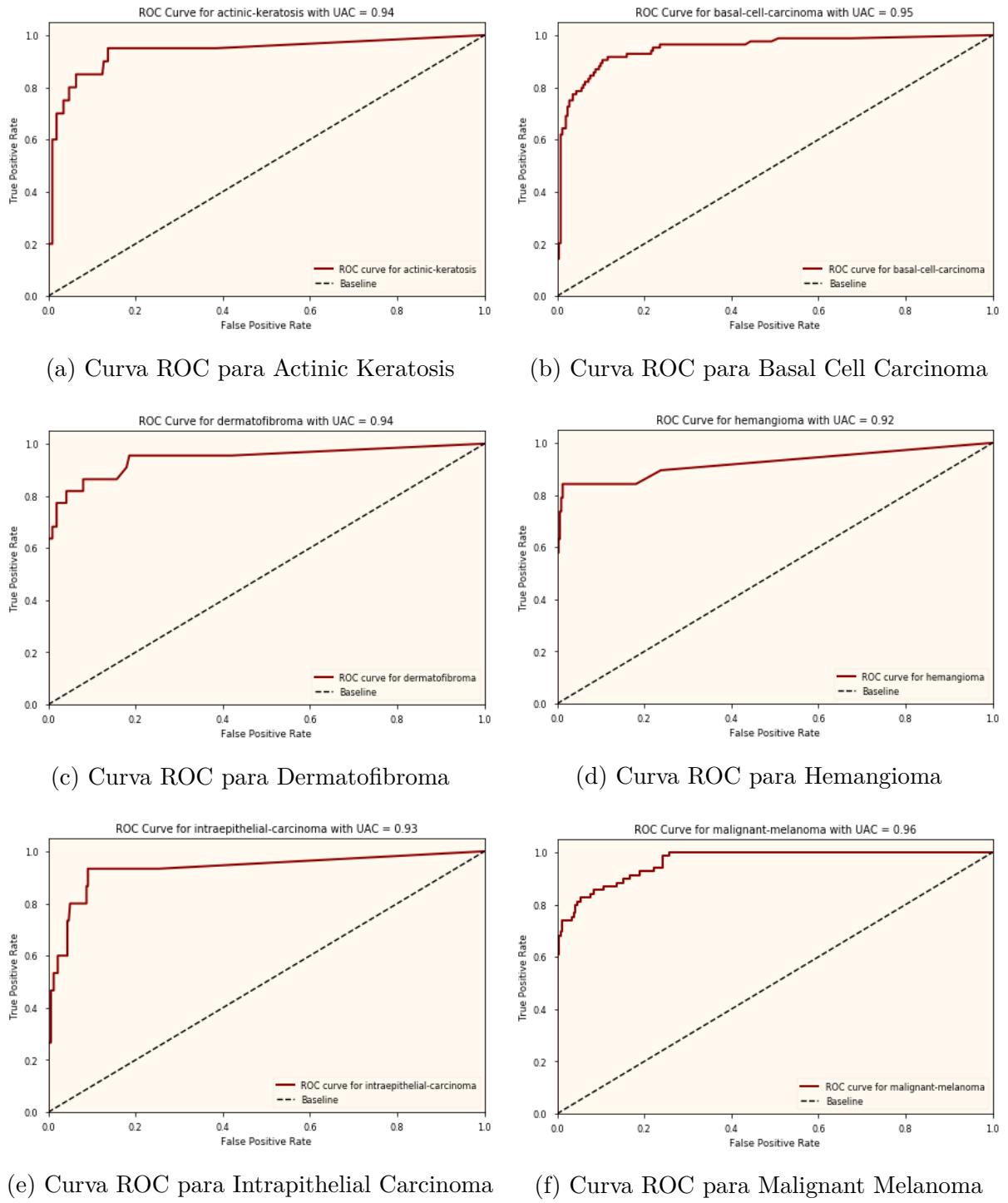


Figura 19 – Curva ROC para o restante das lesões. Continuação 1/2.

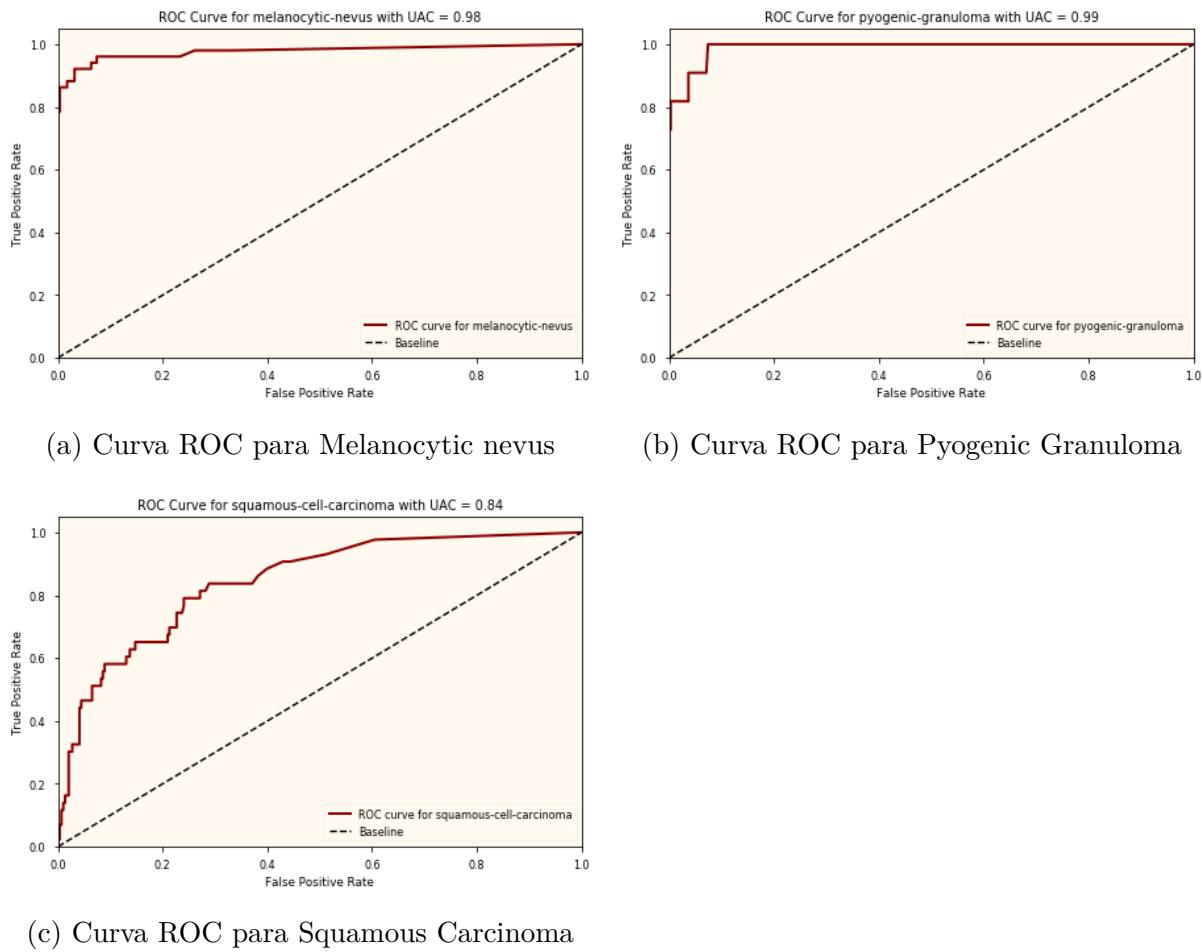


Figura 20 – Curva ROC para o restante das lesões. Continuação 2/2.

Anexos

ANEXO A – Primeiro Anexo

Texto do primeiro anexo.

ANEXO B – Segundo Anexo

Texto do segundo anexo.