Pontifícia Universidade Católica
do Rio de Janeiro

Iam Palatnik de Sousa

Inteligência Artificial Explicável para Classificadores de Imagens Médicas

Tese de Doutorado

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós–graduação em Engenharia Elétrica, do Departamento de Engenharia Elétrica da PUC-Rio.

Orientador : Prof. Eduardo Costa da Silva
Coorientador: Profa. Marley Maria Bernardes Rebuzzi Vellasco

Rio de Janeiro
Março de 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Iam Palatnik de Sousa

Inteligência Artificial Explicável para Classificadores de Imagens Médicas

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós–graduação em Engenharia Elétrica da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Eduardo Costa da Silva
Orientador
Departamento de Engenharia Elétrica – PUC-Rio

Profa. Marley Maria Bernardes Rebuzzi Vellasco
Coorientador
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Carlos Roberto Hall Barbosa
Programa de Pós-graduação em Metrologia – PUC-Rio

Prof. Wouter Caarls
Departamento de Engenharia Elétrica – PUC-Rio

Profa. Karla Tereza Figueiredo Leite
UERJ

Prof. Jorge Luís Machado do Amaral
UERJ

Prof. Renato Cerceau
UERJ

Profa. Sandra Eliza Fontes de Ávila
Unicamp

Rio de Janeiro, 30 de Março de 2021

Iam Palatnik de Sousa

Graduou-se em Física pela Universidade Federal do Rio de Janeiro. Fez mestrado no Programa de Metrologia da PUC-Rio, com ênfase em confiabilidade metrológica de equipamentos de estimulação magnética.

Ficha Catalográfica

Palatnik de Sousa, Iam

Inteligência Artificial Explicável para Classificadores de Imagens Médicas / Iam Palatnik de Sousa; orientador: Eduardo Costa da Silva; coorientador: Marley Maria Bernardes Rebuzzi Vellasco. – 2021.

132 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2021.

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Engenharia de Materiais – Teses. 3. Aprendizado Profundo. 4. Inteligência Artificial Explicável. 5. Imageamento Médico. 6. Visão Computacional. 7. Classificação. 8. COVID-19. I. da Silva, Eduardo Costa. II. Vellasco, Marley M.B.R.. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 620.11

Dedicado à memória de Marcos Palatnik, o *zeyde* (vovô).

Agradecimentos

Desde o ensino médio, frequentemente encarei a ideia do doutorado como um desafio técnico. Um "topo da montanha" que deve ser atingido caso alguém deseje ser um cientista. Uma questão de ter o equipamento certo de escalada, o treinamento apropriado, e fazer o esforço adequado.

Porém, no momento em que eu finalmente me vi escalando essa montanha, percebi que os maiores desafios para o cientista nem sempre vem de tentar aprender uma nova técnica, entender uma equação, implementar um código ou escrever um artigo. A vida como um todo pode nos surpreender com desafios inesperados, nada relacionados com o nosso treino anterior, e que também fazem parte dessa escalada. Tenho certeza que ao menos o ano de 2020 demonstrou isso para todos.

Ao longo do desenvolvimento desta tese, perdi meu avô e avó maternos. Eles teriam gostado de ver este trabalho, e gostaria de agradecer a eles por tudo o que fizeram para possibilitar e incentivar meus estudos. Estendo esse agradecimento a minha família como um todo: mãe, tia, primos, que sempre encorajaram empreitadas acadêmicas e me apoiaram em vários momentos difíceis.

Neste tempo também fiz novos amigos e perdi antigos. Passei por venturas e desventuras, talvez com uma ênfase desagradável nestas últimas. Contudo esses momentos mostram quais pessoas foram constantes em sua presença, independente das circunstâncias. Queria agradecer especialmente aos meus amigos Mari e Gudeco, por me ouvir mesmo nos momentos mais pesados.

Também tive sorte com os meus orientadores. Tanto o professor Eduardo quanto a professora Marley foram compreensivos em vários momentos críticos do projeto, e sou muito grato pelos seus conselhos e por terem me guiado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Também foi recebido financiamento através do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ). Gostaria de estender o agradecimento à estas três agências de fomento, sem as quais este projeto não teria sido possível.

Resumo

Palatnik de Sousa, Iam; da Silva, Eduardo Costa; Vellasco, Marley M.B.R.. **Inteligência Artificial Explicável para Classificadores de Imagens Médicas**. Rio de Janeiro, 2021. 132p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

A inteligência artificial tem gerado resultados promissores na área médica, especialmente na última década. Contudo, os modelos de melhor desempenho apresentam opacidade em relação ao seu funcionamento interno. Nesta tese, são apresentadas novas metodologias e abordagens para o desenvolvimento de classificadores explicáveis de imagens médicas. Dois principais métodos, Squaregrid e EvEx, foram desenvolvidos. O primeiro consiste em uma geração mais grosseira, porém rápida, de *heatmaps* explicativos via segmentações em grades quadrados, enquanto o segundo baseia-se em otimização multi-objetivo, baseada em computação evolucionária, visando ao ajuste fino de parâmetros de segmentação. Notavelmente, ambas as técnicas são agnósticas ao modelo, o que facilita sua utilização para qualquer tipo de classificador de imagens. O potencial destas abordagens foi avaliado em três estudos de caso de classificações médicas: metástases em linfonodos, malária e COVID-19. Para alguns destes casos foram analisados modelos de classificação existentes, publicamente disponíveis. Por outro lado, em outros estudos de caso, novos modelos tiveram que ser treinados. No caso do estudo de COVID-19, a ResNet50 treinada levou a F-scores acima de 0,9 para o conjunto de teste de uma competição para classificação de coronavirus, levando ao terceiro lugar geral. Adicionalmente, técnicas de inteligência artificial já existentes como LIME e GradCAM, bem como *Vanilla*, *Smooth* e *Integrated Gradients* também foram usadas para gerar *heatmaps* e possibilitar comparações. Os resultados aqui descritos ajudaram a demonstrar e preencher parcialmente lacunas associadas à integração das áreas de inteligência artificial explicável e medicina. Eles também ajudaram a demonstrar que as diferentes abordagens de inteligência artificial explicável podem gerar *heatmaps* que focam em características diferentes da imagem. Isso por sua vez demonstra a importância de combinar abordagens para criar um panorama mais completo sobre os modelos classificadores, bem como extrair informações sobre o que estes aprendem.

Palavras-chave

Aprendizado Profundo; Inteligência Artificial Explicável; Imageamento Médico; Visão Computacional; Classificação; COVID-19.

Abstract

Palatnik de Sousa, Iam; da Silva, Eduardo Costa (Advisor); Vellasco, Marley M.B.R. (Co-Advisor). **Explainable Artificial Intelligence for Medical Image Classifiers**. Rio de Janeiro, 2021. 132p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Artificial Intelligence has generated promissing results for the medical area, especially on the last decade. However, the best performing models present opacity when it comes to their internal working. In this thesis, methodologies and approaches are presented for the develpoment of explainable classifiers of medical images. Two main methods, Squaregrid and EvEx, were developed. The first consistts in a rough, but fast, generation of heatmaps via segmentations in square grids, and the second in genetic multi objective optimizations aiming at the fine-tuning of segmentation parameters. Notably, both techniques are agnostic to the model,which facilitates their utilization for any kind of image classifier. The potential of these approaches was demonstrated in three case studies of medical classifications: lymph node mestastases, malária and COVID-19. In some of these cases, already existing classifier models were analyzed, while in some others new models were trained. For the COVID-19 study, the trained ResNet50 provided F-scores above 0.9 in a test set from a coronavirus classification competition, resulting in the third place overall. Additionally, already existing explainable artificial intelligence techniques, such as LIME and GradCAM, as well as Vanilla, Smooth and Integrated Gradients, were also used to generate heatmaps and enable comparisons. The results here described help to demonstrate and improve the gaps in integrating the areas of explainable artificial intelligence and medicine. They also aided in demonstrating that the different types of approaches in explainable artificial intelligence can generate heatmaps that focus on different characteristics of the image. This shows the importance of combining approaches to create a more complete overview of classifier models, as well as extracting informations about what they learned from data.

Keywords

Deep Learning; Explainable Artificial Intelligence; Medical Imaging; Computer Vision; Classification; COVID-1.

Sumário

Lista de figuras

Lista de tabelas

Caminante, no hay camino. Se hace el camino
al andar.

Antonio Machado, *Proverbios y cantares.*

1
Introdução

A medicina frequentemente se beneficia com os desenvolvimentos tecnológicos de outras áreas, conforme aplicações específicas são desenvolvidas. Nos últimos anos, progressos na área de ciência de dados e engenharia biomédica têm criado novos aparelhos e algoritmos que devem lidar com grandes volumes de dados. Contudo, essa quantidade de informação não é acompanhada por um crescimento similar do número de profissionais médicos [1].

Consequentemente, a quantidade de trabalho atribuída individualmente pode se tornar excessiva em alguns casos, motivando o uso de tecnologias que visem a auxiliar estes profissionais, tais como os sistemas de diagnóstico auxiliado por computador, ou *Computer Aided Diagnostics* (CAD).

Com os desenvolvimentos da área de inteligência artificial (IA), particularmente de 2012 em diante, a aplicação destes CADs é cada vez mais usual para problemas de processamento de imagens, em tarefas como classificação e segmentação [2].

Gradualmente, a utilização de sistemas de Inteligência Artificial (IA) para análises de imagens em aplicações médicas tem sido relatada na literatura [2]. Estes sistemas de auxílio à decisão em muitos casos apresentam desempenhos similares ao de patologistas humanos, com a vantagem adicional da escalabilidade.

Atualmente, os modelos com melhor desempenho para classificação de imagens são de uma classe conhecida como Redes Neurais, ou, mais especificamente, Redes Neurais Convolucionais (*Convolutional Neural Network* – CNN). Estas foram aplicadas com sucesso em tarefas de classificação de várias doenças. Uma lista não exaustiva de exemplos inclui: Metástases em linfonodos [3], câncer de mama [4], melanomas [5], malária [6], pneumonia [7], entre muitas outras. Dado este panorama, a utilização de redes neurais e outros sistemas relacionados para diversas aplicações médicas tem se tornado uma área cada vez mais estudada. Isso torna a presença de sistemas de *Machine Learning* (ML) e *Deep Learning* (DL) cada vez mais comuns na área médica [1].

1.1 Motivação

Apesar do panorama favorável acima descrito, ressalta-se que a utilização de redes neurais e sistemas similares para tarefas de classificação de imagem possui o problema de opacidade. A complexidade e não linearidade destes modelos, que justamente os tornam tão eficientes na extração de padrões e atributos de interesse, em geral levam a comportamento descrito na literatura como "caixa-preta" ou *black-box*.

Essa expressão representa sistemas onde não é possível determinar exatamente como uma entrada influencia na saída, em termos compreensíveis para o usuário do sistema. Isso prejudica uma análise transparente do processo de tomada de decisões do sistema. Ressalta-se que é possível destrinchar os componentes internos de uma rede neural, a fim de saber o valor de todas as ativações e pesos sinápticos (conceitos que serão descritos em detalhes no Capítulo 2) em qualquer momento. Porém, estes valores não são interpretáveis para o usuário.

Esse aspecto cria obstáculos para a adoção mais abrangente de sistemas de IA na medicina clínica, pois a área médica requer transparência em relação aos aspectos considerados para decisões críticas [8].

Por sua vez, a área de Inteligência Artificial Explicável (*eXplainable Artificial Intelligence* – XAI) visa justamente a aumentar a transparência destes sistemas *black-box*, por meio da adoção de uma série de novas técnicas.

De forma geral geral, modelos não lineares como CNNs aprendem representações internas a partir dos dados de treino. Estas representações não são imediatamente compreensíveis para usuários humanos. Nas tarefas de classificação de imagem, a área de XAI visa tornar extrair visualizações úteis a partir das representações internas aprendidas pelos modelos classificadores. [9].

Contudo, apesar da literatura relacionada expressar claramente a necessidade de modelos de IA explicáveis para medicina [2, 10], há ainda uma relativa escassez de trabalhos explorando essa importante interface. Neste contexto, destaca-se que no início desta pesquisa não havia basicamente nenhum trabalho deste tipo, embora nos anos seguintes tenha sido observado um interesse crescente no assunto, com todas as principais conferências de IA incluindo *workshops* e encorajando submissões relacionadas a XAI, ou especificamente a XAI em medicina.

A Figura 1.1 mostra o número de publicações, por ano, para as *keywords* "explainable artificial intelligence" e "medical images". Nota-se que antes de 2017 o número é próximo a zero ou zero para todos os anos. Estes números foram obtidos através plataforma Dimensions [11].

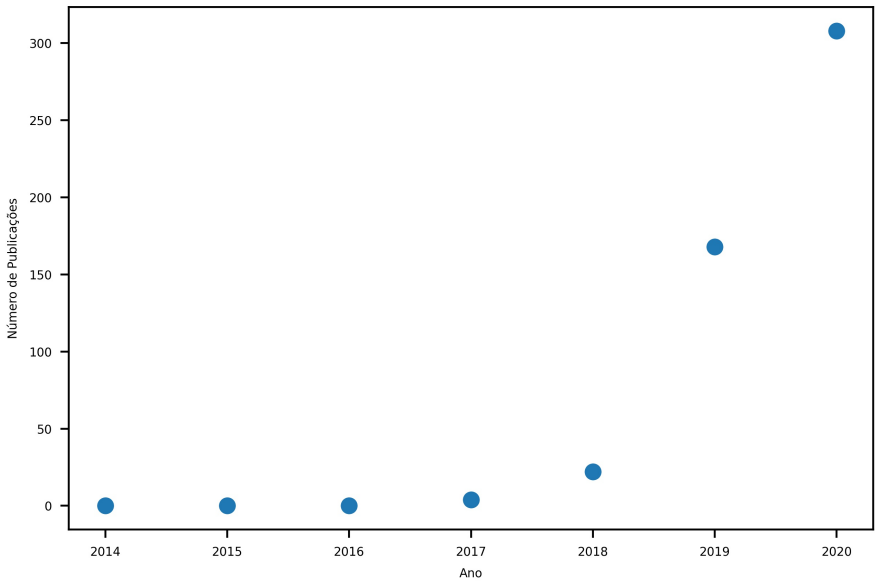


Figura 1.1: Publicações sobre XAI para imagens médicas, por ano.

A motivação deste projeto vem primariamente da necessidade de desenvolver e aprimorar técnicas e modelos mais transparentes para classificadores de imagens médicas.

1.2 Objetivos

O objetivo geral desta tese foi desenvolver sistemas reprodutíveis e universalmente aplicáveis de XAI para classificadores de imagens médicas.

Este objetivo principal pode ser subdividido nos seguintes objetivos específicos:

1. Proposição e/ou aprimoramento técnicas de XAI agnósticas ao modelo, que sejam facilmente implementáveis nos variados *frameworks* de DL.

2. Desenvolvimento de uma metodologia objetiva, propondo métricas de otimização, para ajuste de parâmetros de segmentação em técnicas que dependam de *superpixels*.

3. Proposição de alternativas para aproximar explicações sem necessidade de ajuste de hiper-parâmetros.

4. Aplicação e avaliação das técnicas propostas à dados médicos de imagem.

1.3 Contribuições

Visando a atingir os objetivos acima descritos, as principais contribuições descritas neste trabalho são:

1. Proposição de uma nova abordagem de segmentação em grades, chmada Squaregrid, sem parâmetros, para gerar *heatmaps* via perturbação agnóstica ao modelo.

2. Proposição de um novo esquema de ajuste automático de parâmetros, denominado EvEx, para segmentação necessária na geração de *heatmaps* explicativos, utilizando algoritmos genéticos multi-objetivo.

3. Apresentação de resultados pioneiros de aplicações de XAI por perturbação em dados médicos, mostrando que essas técnicas possibilitam a obtenção de resultados satisfatórios neste nicho de aplicação, dados os devidos cuidados.

4. Abordagem de diferentes modalidades de imagens médicas, incluindo patologias distintas.

5. Implementação e disponibilização, em repositórios públicos para uso geral, de algumas destas novas abordagens e aprimoramentos em abordagens existentes.

1.4 Estrutura da tese

Após este capítulo inicial de introdução, a tese foi dividida de forma a comportar os três grandes estudos de caso que foram desenvolvidos, baseados em *datasets* de metástases em linfonodos, malária e COVID-19. Este trabalho foi dividido em seis capítulos e um apêndice.

O Capítulo 2 introduz os principais conceitos de *Deep Learning* necessários para entender as metodologias adotadas em todos os estudos de caso. Isso inclui os algoritmos de treinamento, as arquiteturas estudadas, funções de ativação, métricas de avaliação de desempenho, funções de custo, entre outros.

O Capítulo 3 faz uma descrição similar, porém das técnicas de XAI, bem como dos algoritmos de segmentação necessários para algumas delas e dos demais conceitos requeridos.

O Capítulo 4 descreve as novas contribuições propostas nesta tese, dando ênfase às técnicas EvEx e Squaregrid, que são explicadas em detalhes. Ademais, outras contribuições menores, de aprimoramentos pontuais em técnicas existentes, estão descritas neste capítulo.

No Capítulo 5 estão descritos os três estudos de caso investigados, cada qual com sua metodologia explicitada. Neste capítulo se faz uso extensivo de todos os conceitos definidos nos capítulos anteriores, que servem como fundamentação. Também são apresentados e discutidos os resultados obtidos para cada estudo de caso.

O Capítulo 6 inclui a conclusão geral e sugestões de trabalho futuros.

Por sua vez, o Apêndice A destaca todas as publicações feitas ao longo do desenvolvimento deste projeto, bem como links para os repositórios das contribuições que já foram compartilhadas no Github.

2
Deep Learning

Neste capítulo são apresentados os conceitos fundamentais do aprendizado profundo (*Deep Learning* – DL), incluindo uma discussão das premissas básicas do tema, a descrição da terminologia para algoritmos de aprendizado, bem como os conceitos de métricas, otimizadores e funções de perda (*loss*). Na sequência, discute-se sobre *datasets* desbalanceados, bem como sobre as principais maneiras de se abordar este problema: *data augmentation*, *test time augmentation* (TTA), *oversampling* e *ensembling*.

Por fim, as características das diversas redes convolucionais usadas na implementação dos classificadores desenvolvidos ao longo das pesquisas conduzidas no escopo desta tese também são descritas. Como estas técnicas foram largamente usadas ao longo deste trabalho, note que este capítulo também inclui parte considerável da metodologia desta pesquisa.

2.1
Conceitos Básicos

2.1.1
Inteligência Artificial

O conceito de Inteligência Artificial (IA) é central para a adequada compreensão dos temas abordados nesta tese, e será citado múltiplas vezes ao longo do trabalho. De forma geral [12], modelos de inteligência artificial são sistemas que podem aprender de forma adaptativa a resolver tarefas, buscando se auto-ajustar para atingir um dado objetivo.

Alternativamente, também podem ser definidas, de forma mais simples, como sistemas que tentam automatizar tarefas intelectuais normalmente executadas por humanos [13].

Existem múltiplas técnicas e subdivisões dentro da pesquisa de IA. De maneira geral, estas técnicas compartilham aspectos em comum que serão descritos nas próximas seções.

2.1.2 *Machine Learning*

O Aprendizado de Máquina (*Machine Learning* – ML) é normalmente visto como um desdobramento ou abordagem dentro do âmbito da pesquisa de inteligência artificial.

A principal diferença entre abordagens de *machine learning* e aquelas inerentes a outros paradigmas de programação é que em ML as regras e operações para processar dados, realizar tarefas e atingir objetivos não são explicitamente definidas pelo programador [13]. Ao invés disso, o algoritmo é treinado com uma série de técnicas que permitem uma identificação automatizada das regras e operações que otimizam um dado objetivo. Essas técnicas serão detalhadas ao longo deste capítulo.

2.1.3 *Deep Learning*

Dentro do escopo de *machine learning*, uma das sub-áreas mais recentes inclui a adoção de modelos com um grande número de camadas de processamento de dados. O objetivo é que as camadas possam aprender representações dos dados, onde cada camada adicional aprende uma representação mais abstrata, possibilitando melhor desempenho em tarefas variadas [13].

Estes modelos com muitas camadas são comumente chamados de modelos "profundos". Por este motivo, esta sub-área de ML é conhecida como aprendizado profundo (*Deep Learning* – DL).

2.2 Algoritmos de Aprendizado

2.2.1 Treinamento de modelos

Um dos aspectos fundamentais de ML e DL envolve algoritmos para o treinamento de modelos e/ou sistemas, para que estes aprendam a realizar determinadas tarefas ou atingir objetivos específicos.

As duas principais formas de treinamento de algoritmos são: supervisionado e não-supervisionado.

2.2.1.1 Treinamento Supervisionado

Em geral este é o tipo mais comum de treinamento, que requer que se faça o mapeamento de um conjunto de dados para um conjunto de anotações (*labels*)

conhecido [13]. Estes *labels* podem ser codificados de várias maneiras, incluindo números inteiros, vetores *one-hot*, máscaras de pixels, strings de texto, entre outros.

O modelo precisa aprender a processar corretamente os dados para que sua saída (*output*) consiga identificar corretamente o *label* associado a cada amostra.

Um exemplo típico de aplicação de treinamento supervisionado, utilizado de forma recorrente nesta tese, é em tarefas de classificação automática de imagens. Nesse caso, o *dataset* em questão é um conjunto de imagens e os *labels* representam as diferentes classes às quais cada uma destas imagens podem pertencer. De forma geral, nesta tese, serão abordadas classificações de imagens médicas entre as classes "saudável" e "não saudável".

Ressalta-se que todas as aplicações desenvolvidas e discutidas nesta tese são casos de treinamento supervisionado.

2.2.1.2 Treinamento Não Supervisionado

O treinamento dos sistemas de AI também pode ser feito com dados sem anotações. Neste caso, diz-se que o treinamento é não supervisionado. Modelos treinados com esta abordagem, de forma geral, buscam padrões, transformações e/ou representações dos dados que sejam úteis para diversas aplicações. Estas podem incluir: visualização ou compressão de dados, anotação automática dos mesmos, geração de dados artificiais que seguem a mesma distribuição dos dados originais, entre outras [13].

2.2.2 Generalização

Ao treinar um modelo de IA, deseja-se que ele não apenas consiga realizar uma dada tarefa com os dados apresentados inicialmente, mas que possa também extrapolar esse desempenho para novos dados, não vistos durante as etapas de treinamento. Esse comportamento é comumente chamado de "generalização", sendo um dos objetivos fundamentais a serem alcançados em aplicações de IA, ML e DL. Ao contrário, em situações onde um modelo se ajusta excessivamente aos dados presentes no conjunto de treinamento e não consegue generalizar para novos dados, diz-se que o modelo apresenta *overfit* [12, 13].

Para avaliar a capacidade de generalização de um modelo e reduzir o *overfit*, a estratégia mais convencional consiste em dividir os *datasets* em conjuntos de treino, validação e teste. O conjunto de treino, conforme o

próprio nome indica, é um subconjunto do *dataset* usado para treinar o modelo inicialmente. Por sua vez, o conjunto de validação é um subconjunto do *dataset* não apresentado ao modelo durante seu treinamento, utilizado a fim de permitir que se teste o modelo quanto a sua capacidade de generalização.

Dessa forma, é possível realizar ajustes nos parâmetros do modelo que não sejam automaticamente otimizados durante o treino. Tais parâmetros, que precisam ser decididos separadamente, são comumente chamados de hiperparâmetros. No entanto, o ajuste excessivo de hiperparâmetros também pode enviesar o modelo em relação ao conjunto de validação.

Consequentemente, para se ter um avaliação final da real capacidade de generalização do modelo, é comum usar também um conjunto de teste. Esse subconjunto do *dataset* não é usado nem no treino nem na etapa de ajuste de hiperparâmetros, ficando oculto para o modelo até o momento final de sua avaliação. O conjunto de teste é também chamado de *out-of-sample*

2.2.3 Métricas de Desempenho

Para medir e comparar o desempenho de modelos de IA desenvolvidos, a literatura estabelece algumas "métricas"[13, 14]. Para cada tipo de tarefa em ML ou DL existem diferentes métricas condizentes. Abaixo estão descritas as principais métricas utilizadas para problemas de classificação, que são o foco desta tese.

Antes de discutir as métricas específicas, contudo, convém definir as noções de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos. Imaginando-se uma classificação binária com classes “positiva” e “negativa”, tem-se:

(i) Verdadeiros Positivos (*True Positives – TP*) – O modelo identifica corretamente uma amostra como sendo da classe positiva;

(ii) Falsos Positivos (*False Positive – FP*) – O modelo identifica incorretamente uma amostra negativa como sendo positiva.

(iii) Verdadeiros Negativos (*True Negatives – TN*) – O modelo identifica corretamente uma amostra como sendo da classe negativa;

(iv) Falsos Negativo (*False Negative – FN*) – O modelo identifica incorretamente uma amostra como sendo negativa.

Estas noções são úteis para definir as métricas a seguir [15]:

2.2.3.1 Exatidão ou Acurácia

Uma das formas mais simples de avaliar o desempenho de um modelo de classificação é dividir o número de acertos pelo número total de classificações. Esta grandeza é conhecida como *accuracy* em Inglês.

Ressalta-se que, embora a tradução oficial de *accuracy*, segundo o Vocabulário Internacional de Metrologia [16] seja "exatidão", o termo "acurácia" é mais comum em publicações de ML e DL em Português, sendo portanto o termo usado neste trabalho.

A acurácia (acc) de um classificador pode ser escrita como:

$$acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2-1}$$

2.2.3.2 Sensibilidade

A sensibilidade (sn), também chamada às vezes de *recall* ou revocação (r) mede a fração de amostras positivas corretamente classificadas entre todas as predições positivas.

$$sn = \frac{TP}{TP + FN} \tag{2-2}$$

2.2.3.3 Especificidade

A especifidade (sp) mede a fração de amostras negativas corretamente classificadas entre todas as predições negativas.

$$sp = \frac{TN}{TN + FP} \tag{2-3}$$

2.2.3.4 Precisão

A precisão (p) mede a fração de positivos corretamente classificados entre todas as predições positivas.

$$p = \frac{TP}{TP + FP} \tag{2-4}$$

2.2.3.5 F-score

Conforme discutido acima, a acurácia é uma das métricas mais usadas para avaliação de modelos. Porém, ressalta-se que essa métrica não é adequada para *datasets* desbalanceados, em que uma classe esteja representada em

excesso. Note que, em casos como estes, é possível errar todas as classificações de uma dada classe e ainda obter acurácias altas.

Por sua vez, o F-Score, também conhecido como F1-score ou simplesmente F, mostra-se uma métrica mais adequada do que a acurácia nesses casos. Matematicamente, ela pode ser interpretada como a média harmônica entre a precisão e a revocação. Para *datasets* desbalanceados a acurácia não é uma métrica confiável. Por outro lado, buscar um equilíbrio entre precisão e revocação leva a resultados melhores, conforme feito pelo F-Score e discutido em [15].

$$F_{score} = \frac{2rp}{r+p} \tag{2-5}$$

2.2.3.6 AUC

As métricas discutidas até aqui requerem a escolha do chamado limiar (*threshold*) de decisão/discriminação. Esse é o valor a partir do qual se decide se uma dada classificação é da classe "positiva" (1) ou "negativa" (0). Tipicamente, o limiar de 0,5 é usado em uma classificação binária, porém é possível que a mudança deste limiar melhore a performance de um dado classificador.

Por sua vez, a AUC (*Area Under Curve* ou Área sob a Curva, que se refere ao valor da área abaixo da curva característica de operação do receptor (*Receiver Operating Characteristic Curve* – ROC), permite avaliar o desempenho de um classificador sem depender da escolha do limiar.

Esta curva é obtida por meio de um gráfico da taxa de verdadeiros positivos (*True Positive Ratio* – TPR) em função da taxa de falsos positivos (*False Positive Ratio* – FPR). Note que o TPR corresponde exatamente à sensibilidade, enquanto o FPR corresponde à $1-sp$. A Figura 2.1 exemplifica o comportamento de uma destas curvas, utilizada para cálculo da AUC. Ressalta-se que o melhor valor possível para o AUC seria 1.

2.2.4 Perda/Custo

Outro conceito fundamental, similar porém distinto às métricas do treino, é o das funções de perda, também conhecidas como funções de custo, ou *losses*.

Enquanto as métricas de desempenho até então discutidas fornecem uma ideia do desempenho do classificador e do número de erros/acertos de classificação cometidos, as funções de custo calculam a diferença entre os valores preditos e as *labels*. Dessa forma, as funções de custo quantificam

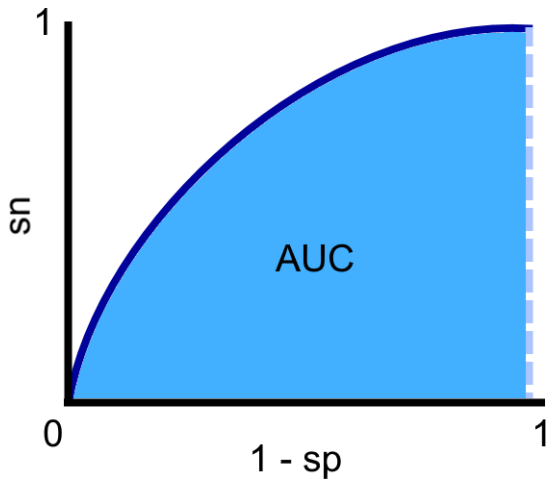


Figura 2.1: Representação gráfica da métrica AUC.

o erro cometido pelo classificador sem usar limiares de discriminação ou arrendondamentos. Isso é importante, dado que as funções de custo devem ser diferenciáveis para permitir o funcionamento dos otimizadores no algoritmo de treinamento, conforme discutiremos adiante.

É possível usar funções de custo como métricas para avaliar o desempenho do classificador, porém o mais comum é usar estas funções para a otimização do classificador, e usar métricas mais fáceis de interpretar apenas como uma visualização do desempenho, sem interferir diretamente no treino.

Como no caso das métricas de desempenho, há uma série de funções de custo aplicáveis a diferentes tipos de problemas e tarefas. Em particular, para esta tese, que lida com classificações binárias ou multiclasse, as funções de custo mais relevantes são apresentadas a seguir.

2.2.4.1 Entropia Cruzada Binária

Esta função de custo é utilizada para problemas de classificação em que as *labels* foram codificadas de forma binária (0 ou 1) [13].

A Entropia Cruzada Binária é dada por[17]:

$$L_{BCE}(y_{true}, y_{pred}) = -\left[y_{true} \log y_{pred} + (1 - y_{true}) \log(1 - y_{pred})\right] \qquad (2\text{-}6)$$

onde y_{true} e y_{pred} são as *labels* do *dataset* e as predições do classificador, respectivamente.

2.2.4.2 Entropia Cruzada Categórica

É possível generalizar a entropia cruzada binária para problemas multi-classe. Nestes casos utiliza-se a função de custo denominada Entropia Cruzada Categórica [17], que pressupõe uma codificação *one-hot* das predições e *labels* [13]. Na codificação *one-hot* os valores são transformados em vetores com combinações de bits onde somente um dos bits tem valor um, e todos os demais têm valor 0.

Esta função de custo é dada por:

$$L_{CCE}(y_{true}, y_{pred}) = -\sum_{c}^{C} y_{true}^{c} \log y_{pred}^{c} \tag{2-7}$$

onde y_{true} e y_{pred} são as *labels* do *dataset* e as predições do classificador, C é número total de classes e c são as classes do problema de classificação multi-classe.

2.2.5 Otimizadores

As funções de custo descritas acima basicamente determinam quais objetivos o modelo tentará otimizar durante o treinamento. Já em relação ao algoritmo de otimização, existem várias abordagens na literatura, as quais variam com o tipo de modelo treinado e o tipo de problema estudado. Em particular, o tipo de modelo de classificação utilizado nesta tese são redes neurais (mais especificamente, redes neurais convolucionais), e para minimizar a função de custo optou-se pelo uso de otimizadores do tipo Gradiente Decrescente ou similares [18].

Um otimizador essencialmente é um algoritmo que toma uma função de custo e tenta minimizá-la ou maximizá-la, ajustando os parâmetros do modelo em treinamento. Nesta subseção, detalham-se os dois otimizadores usados para as aplicações estudadas nesta tese. Estes dois otimizadores são, de forma geral, os mais usados na literatura, porém existem dezenas de outras variantes minuciosamente discutidas em [18], as quais não serão aqui detalhadas por estarem fora do escopo deste trabalho.

2.2.5.1 Gradiente Decrescente

O algoritmo de Gradiente Decrescente envolve o cálculo do gradiente da função de custo causado por mudança nos parâmetros do modelo que está sendo treinado. O algoritmo faz ajustes nos parâmetros que vão na direção contrária (ou descendente) ao gradiente da função de custo. O gradiente pode

ser calculado: (1) após apresentar cada amostra ao modelo durante o treino, (2) após apresentar o *dataset* inteiro como um *batch* de amostras, ou (3) após apresentar pequenos *batches*, frequentemente chamados de *mini-batches*, de forma sucessiva.

Na prática, conforme mencionado por Ruder [18], está demonstrado que o uso de *mini-batches* oferece os melhores resultados para o treino de redes neurais em diversas tarefas. No entanto, por uma questão de nomenclatura, é comum se referir tanto ao Gradiente Decrescente feito amostra por amostra, quanto àquele feito com *mini-batches*, como Gradiente Decrescente Estocástico (*Stochastic Gradient Descent* – SGD).

A atualização de parâmetros usados por este algoritmo, para uma função de custo L em função dos parâmetros do modelo, θ, é:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t, x^{batch}, y^{batch}_{true}) \tag{2-8}$$

onde η é a taxa de aprendizado, x são as amostras do conjunto de treino e y^{batch}_{true} são as *labels* referentes às mesmas amostras, para um dado *mini-batch*.

De forma geral, a taxa de aprendizado e o tamanho do *batch* são tratados como hiperparâmetros no treino de um modelo de classificação.

2.2.5.2 Momento

É possível interpretar o gradiente decrescente como um algoritmo que percorre a superfície da função de custo, em uma série de passos de atualização, tentando encontrar o mínimo global. Entretanto, na prática, a função de custo não é suave, fazendo com que o SGD possa ficar frequentemente preso em mínimos locais, ou oscilando ao redor destes, sem conseguir progredir em direção ao mínimo global.

Uma das soluções comumente adotadas para contornar este problema [19] consiste em somar uma fração de cada passo de atualização, multiplicada por um coeficiente, ao passo de atualização seguinte. Dessa forma, as componentes de cada passo que coincidam são favorecidas, e as componentes que oscilam são atenuadas. Por sua vez, o coeficiente que multiplica a fração do passo anterior é frequentemente chamado de momento, ou termo de momento (γ).

Incluindo o termo de momento à equação 2-8, tem-se:

$$\begin{aligned} \Delta_t &= \gamma \Delta_{t-1} - \eta \nabla_\theta L(\theta, x^{batch}, y^{batch}_{true}) \\ \theta_{t+1} &= \theta_t - \Delta_t \end{aligned} \tag{2-9}$$

2.2.5.3 Adam

Conforme até então discutido, o otimizador de gradiente decrescente com momento possui uma taxa de aprendizado e um termo de momento constantes durante todo o treino. Por outro lado, otimizadores mais sofisticados podem ter um comportamento adaptativo, com a taxa de aprendizado e momento sofrendo ajustes ao longo do processo de treinamento, dependendo de características dos parâmetros de atualização, da função de custo e do seu gradiente.

Um dos otimizadores adaptativos mais utilizados é o otimizador de Estimativa Adaptativa de Momento (*Adaptive Moment Estimation* – Adam) [20]. Em várias aplicações este otimizador converge mais rapidamente que quaisquer outros para mínimos satisfatoriamente pequenos da função de custo, alcançado assim bons desempenhos [18]. Consequentemente, se tornou um otimizador extremamente popular.

A regra para atualização de parâmetros no algoritmo Adam é:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2-10}$$

onde $\hat{m}_t$ e $\hat{v}_t$ são estimativas da média e variância do gradiente ao longo treinamento, e η é uma constante para estabilidade numérica.

Uma discussão mais detalhada, com equacionamento mais extenso, pode ser encontrada em [18, 20].

2.2.5.4 SWATS

Keskar e Socher identificaram que, embora o Adam convirja mais rápido e possibilite o encurtamento do tempo de treinamento de redes neurais em geral, o algoritmo SGD frequentemente consegue encontrar mínimos melhores, quando executado por tempos de treinamento suficiente longos [21].

Estudos conduzidos por estes autores concluíram que pode haver casos onde é desejável usar Adam para iniciar o treino e convergir rapidamente, e então trocar o algoritmo de otimização para SGD para finalizar o treino, a fim de se obter as vantagens das duas abordagens. Esta estratégia é chamada de "*Switch from Adam To SGD*", ou SWATS [21].

2.2.6 *Backpropagation of Errors*

O algoritmo de retropropagação de erros (mais comumente conhecido pelo nome em inglês, *backpropagation*), permite usar todos os componentes

discutidos até aqui para efetuar o ajuste de parâmetros do modelo.

Os principais *frameworks* de *deep learning* atuais, como Tensorflow/Keras, por exemplo, usam diferenciação simbólica [13] para obter os gradientes usados pelos otimizadores.

Estas atualizações de peso são propagadas da camada de saída até a camada de entrada utilizando a regra da cadeia. Para uma função arbitrária C que depende de uma função w que depende de z, pode-se expressar a regra da cadeia por meio da seguinte relação:

$$\frac{\partial C(z)}{\partial w} = \frac{\partial C(z)}{\partial z}\frac{\partial z}{\partial w} \tag{2-11}$$

No caso do algoritmo de *backpropagation*, C é a função de custo adotada pelo modelo, w são os pesos sinápticos e z a soma ponderada de entradas para uma dada função de ativação.

Uma demonstração detalhada desse cálculo para diferentes arquiteturas e ativações pode ser encontrada em [14].

Na prática, os *frameworks* de DL apenas permitem definir funções de custo e ativações que sejam diferenciáveis, de forma que o usuário tipicamente não faz cálculos explícitos de gradiente. Contudo, é possível obter os gradientes caso isso seja importante, conforme será discutido na seção 3.2.1.

2.3 *Datasets* Desbalanceados

Ainda considerando o exemplo ilustrativo de classificação binária, é importante discutir um possível e comum problema que ocorre em vários *datasets* reais, inclusive em um dos estudados nesta tese: dados desbalanceados.

O desbalanceamento ocorre quando há mais exemplos de uma classe do que outra. Quanto maior esta diferença, mais nocivos os efeitos se tornam. Nestes casos, conforme mencionado na discussão das métricas de desempenho, é inteiramente possível ter métricas que mostrem resultados aparentemente satisfatórios mesmo que se esteja errando todas as classificações de uma dada classe, desde que esta classe tenha pouca representatividade no *dataset*. Esse problema também inclui as funções de custo, afetando o treinamento como um todo, criando um classificador enviesado.

De fato, é comum que *datasets* apresentem este problema. Por exemplo, um *dataset* médico com uma classe saudável e uma doente, frequentemente, tem mais exemplos da classe saudável, justamente pela doença ser um evento mais raro ou incomum. A Figura 2.2 mostra exemplos de *datasets* balanceados e desbalanceados.

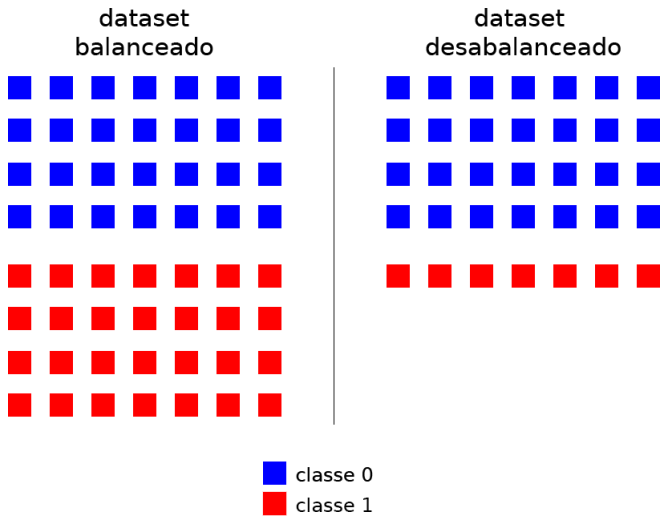


Figura 2.2: Exemplo visual de um *dataset* balanceado, e um desbalanceado. Cada dataset é dividido em duas classes, 0 e 1.

Ao se trabalhar com *datasets* desbalanceados, faz-se necessário fazer algum tipo de correção, a fim de se evitar que os classificadores desenvolvidos sejam enviesados pelo desbalanceamento. As próximas subseções discutem algumas abordagem frequentemente usadas em aplicações de DL, visando mitigar problemas decorrentes do desbalanceamento de dados.

2.3.1 *Undersampling*

A abordagem mais simples é conhecida como subamostragem ou *under-sampling*, consistindo em remover amostras da classe super-representada no *dataset* até que ambas as classes tenham o mesmo número de instâncias.

Esta abordagem mostra-se eficaz quando se dispõe de um grande volume de dados, de modo que a remoção de instâncias não torne o *dataset* pequeno demais para treinar o modelo com sucesso. O processo pode ser repetido n vezes estocasticamente, com uma média sendo calculada ao final. Entretanto, quando se está lidando com um *dataset* pequeno, é preferível usar outras técnicas para evitar o desperdício de dados viáveis.

2.3.2 *Data Augmentation*

O Aumento de dados, mais comumente conhecido pelo nome em Inglês, *Data Augmentation*, é outra estratégia adotada tanto para aumentar a quantidade de dados de treinamento, quanto para equilibrar o número de exemplos de cada classe presentes no *dataset*.

Para *datasets* de imagens, como os que foram usados nesta tese, esta abordagem envolve criar cópias das imagens do *dataset* e aplicar uma série de operações que causem perturbações. Por sua vez, estas novas imagens modificadas podem ajudar o modelo que se deseja treinar à generalizar melhor, dado que ele passa a ser exposto a uma gama maior de padrões e atributos.

Algumas das perturbações comumente usadas para imagens são:

1. Rotação – Rodar a imagem por ângulo aleatório;
2. Translação – Transladar a imagem por uma quantidade arbitrária;
3. *Flip* – Espelhar a imagem, verticalmente ou horizontalmente;
4. Zoom – Mudar o nível de zoom, aproximando-se ou afastando-se da imagem.

É importante ressaltar que algum nível de conhecimento sobre as características do problema em questão é requerido para selecionar quais destas perturbações fazem sentido. Caso contrário, apresentar imagens que não sejam verossímeis para o problema pode, na verdade, prejudicar o desempenho do classificador. A Figura 2.3 ilustra exemplos de pertubações convencionalmente aplicadas a imagens, para *Data Augmentation*.

Atualmente, diferentes *frameworks* de *deep learning* para treinamento de redes permitem aplicar *data augmentation* de várias formas:

1. Antes do treino, aumentando o *dataset* e criando um conjunto maior de imagens antes de iniciar o processo de otimização do modelo;
2. Em tempo real, usando funções de aumento que aplicam as perturbações ao mesmo tempo em que carregam os arquivos das imagens e os passam ao modelo [17];

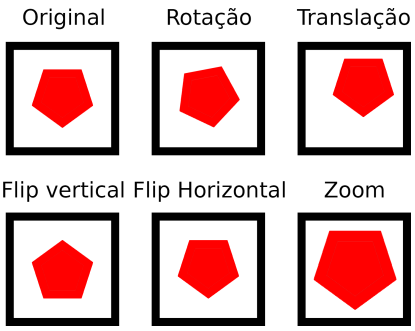


Figura 2.3: Exemplos de aumento das imagens.

3. Em tempo real, como camadas que se encontram dentro do modelo, e portanto podem desfrutar da paralelização e aceleração oferecida por unidades gráficas (*Graphics Processing Units* – GPU). Notavelmente, esta é a forma sugerida nas versões mais recentes do *framework* Tensorflow 2.0 [17].

2.3.2.1 *Test Time Augmentation*

A técnica de *data augmentation* pode ser usada não só durante o treino da rede, mas também durante a etapa de inferência do modelo, em que as classificações estão sendo feitas. Esta abordagem objetiva tentar melhorar o desempenho do modelo.

Para esse fim, quando uma imagem é apresentada ao classificador, várias versões modificadas da imagem também são apresentadas posteriormente. Por sua vez, a predição final é uma média de todas as predições feitas pelo modelo, para a imagem original e suas versões alteradas por *data augmentation*.

2.3.3 *Oversampling*

Em contraponto à estratégia de *undersampling*, existe a abordagem de *oversampling* que consiste em repetir exemplos da classe menos representativa do *dataset* até que ela tenha a mesma quantidade de amostras da maior classe. Entretanto, esta estratégia apresenta um grande risco de enviesar o classificador, ao gerar muitos exemplos idênticos.

Por este motivo, frequentemente, a estratégia de *oversampling* é combinada com a estratégia de *data augmentation*, de forma a não criar exemplos repetidos, permitindo que se complete o *dataset* com imagens diferentes entre si, associadas à classe menos representativa.

2.3.4 *Ensembling*

Todas as estratégias aqui discutidas podem ser combinadas de modo a se implementar uma abordagem conhecida como *ensembling*, em que se treina mais de um modelo e as predições são obtidas como combinações das predições dos vários modelos treinados. É possível fazer essa combinação tomando-se a média das predições, a média ponderada, fazendo-se um esquema de votação, entre outras [22].

Na prática, de forma geral, se usam combinações de todas estas abordagens para um dado problema, de modo a se poder avaliar quais têm um melhor desempenho.

2.4 Arquiteturas Estudadas

Nesta seção será feita uma breve revisão dos conceitos fundamentais de redes neurais e redes neurais convolucionais. Também será feita uma apresentação detalhada das principais características das arquiteturas estudadas nesta tese.

2.4.1 Redes Neurais

Todos os modelos estudados nesta tese fazem parte de uma família de classificadores conhecidos como redes neurais. Nesta subseção estes modelos serão discutidos minuciosamente, a fim de se definir e esclarecer detalhes fundamentais para o entendimento dos tópicos abordados ao longo desta tese.

2.4.1.1 Camadas

De forma geral, as redes neurais são compostas por camadas que recebem uma entrada, a processam, passam uma soma ponderada das entradas processadas por uma função de ativação e fornecem uma saída que segue para a entrada da camada seguinte.

Por sua vez, os neurônios artificiais são a menor divisão de cada camada. Este tipo de modelo tem inspiração no modelo biológico do cérebro, por isso muito da terminologia é emprestada da área de anatomia neurológica. A organização destes neurônios e camadas, bem como as conexões entre estes, define a arquitetura da rede neural. A Figura 2.4 mostra um exemplo visual destes conceitos.

Em problemas de classificação de imagens, por exemplo, os neurônios na camada inicial (também chamada de camada de entrada) recebem os valores (R,G,B) de cada pixel da imagem.

Na prática, há uma quantidade enorme de maneiras de organizar os neurônios nas camadas e conectá-los àqueles de outras camadas. Os vários tipos de conexões e arranjos por vezes têm nomes específicos, e estão associados com arquiteturas específicas de redes neurais.

Talvez, a primeira ideia que surja seja conectar as saídas de todos os neurônios de uma dada camada às entradas de todos os neurônios da camada

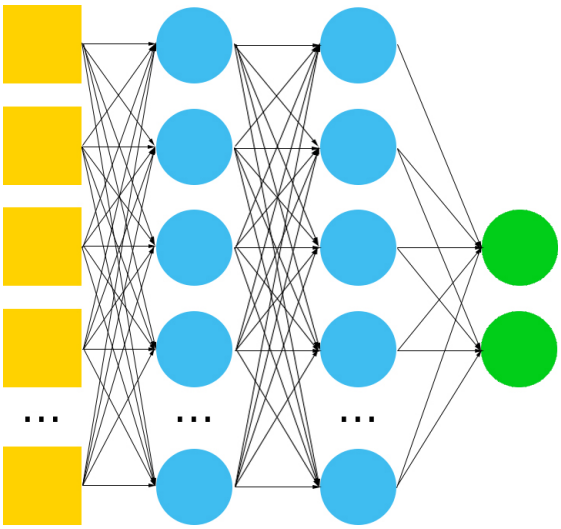


Figura 2.4: Exemplo de arquitetura de rede neural. Cada forma geométrica individual representa um neurônio, e estes estão agrupados em camadas separadas por cores. A camada amarela é conhecida como camada de entrada ou *input*, as camadas azuis são camadas intermediárias, ou ocultas, e a camada verde é a de *output*, ou saída.

seguinte. Este tipo de arranjo é amplamente utilizado, formando camadas totalmente conectadas, também conhecidas como densamente conectadas. Frequentemente, são utilizadas como um bloco classificador ao final de modelos de redes neurais profundas, para tentar capturar padrões e correlações entre os dados já processados.

Modelos com duas ou mais camadas densamente conectadas também são conhecidos como perceptrons multicamada, mais comumente chamados pelo termo em Inglês *multilayer perceptron* – MLP [23]. O exemplo da Figura 2.4 é uma MLP com 2 camadas ocultas.

Outras camadas e blocos de camadas comumente utilizadas serão discutidas nas subseções seguintes:

- Camadas de convolução;
- Camadas de *pooling*;
- Camadas de normalização em *batch*;
- Blocos residuais.

2.4.2 Funções de ativação

As camadas tipicamente recebem dados de uma série de conexões e calculam uma soma ponderada das entradas pelos pesos sinápticos. Esta soma é então apresentada às funções de ativação. A escolha adequada destas funções é fundamental para o desempenho das redes na tarefa de classificação.

De forma geral, existem cinco funções de ativação convencionalmente usadas em redes neurais profundas destinadas a tarefas de classificação:

- Sigmóide
- Tangente Hiperbólica
- Linear
- ReLU
- *Softmax*

Vários trabalhos comprovam a capacidade de redes neurais multicamadas de atuarem como aproximadores universais [24], bem como a importância das não-linearidades presentes nestes modelos para poder capturar correlações mais complexas entre os dados e possibilitar esta aproximação. De forma geral, estas não linearidades decorrem das funções de ativação descritas nas próximas subseções [25].

2.4.2.1 Sigmóide

A função de ativação sigmóide é dada pela relação:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2-12}$$

Funções como estas são conhecidas como funções de "achatamento", ou "*squashing*", por reduzirem suas saídas a intervalos finitos. Por muito tempo, as ativações sigmóides foram as mais utilizadas para treinar redes com múltiplas camadas, até serem substituídas por ativações tipo ReLU ou similares, conforme discutido a seguir.

Atualmente, no escopo de classificação de imagens, as funções sigmóide são principalmente utilizadas como ativações da camada de saída em problemas de classificação binários, por possibilitarem uma interpretação fácil da saída como uma probabilidade, entre 0 e 1.

2.4.2.2 Tangente Hiperbólica

A ativação *tanh* é dada por:

$$\sigma(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2-13}$$

Esta ativação também é uma função de achatamento, similar à sigmóide, porém achata os valores para o intervalo [-1;1] ao invés de [0;1]. Dependendo da codificação dos dados de entrada pode fazer mais sentido usar a ativação *tanh* do que a sigmóide.

2.4.2.3 Linear

A ativação linear é dada por:

$$\sigma(x) = x \tag{2-14}$$

Esta função é normalmente usada como ativação da camada de saída em problemas de regressão, onde não se deseja limitar a saída final a um intervalo específico.

2.4.2.4 ReLU

As unidades de retificação linear (*Rectified Linear Units* – ReLU) são ativações lineares com uma não-linearidade introduzida para valores de entrada negativos.

Esta é a principal função de ativação usada para as camadas intermediárias/ocultas das redes convolucionais destinadas a classificação de imagens, desde que se demonstrou que elas solucionavam o problema do gradiente evanescente e permitiam o treinamento de redes profundas [26].

A equação das ativações ReLU é apresentada a seguir:

$$\sigma(x) = max(0, x) \tag{2-15}$$

2.4.2.5 *Softmax*

A ativação *softmax*, apresentada na Equação (2-16), atua como uma extensão da ativação sigmóide para problemas de ativação multiclasse.

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_j^N e^{x_j}} \tag{2-16}$$

onde i, j são índices referentes às entradas e saídas da camada em questão, e N é o número de neurônios de saída.

Com base nesta definição, pode-se perceber que a soma total das ativações para todos os neurônios de saída é 1. Dessa forma, é comum interpretar as saídas *softmax* como as probabilidades de predição de cada classe, sendo a soma total de todas as probabilidades 100%.

O uso da ativação *softmax* pressupõe que as saídas serão comparadas com *labels one-hot* e que o problema tem apenas uma classe possível para cada exemplo. Isso implica uma correlação entre as probabilidade das diferentes classes, de modo que um aumento na probabilidade da presença de uma classe diminua a probabilidade da presença de outra.

2.4.3 *Dropout*

Outra técnica relevante para o treino de redes neurais, útil para reduzir o *overfit*, envolve desligar aleatoriamente as saídas dos neurônios de uma camada.

Ao fazer isso durante o treino, a ideia desta abordagem é de incentivar a rede neural a não depender de padrões espúrios que podem tem surgido em determinadas conexões entre neurônios.

Esta técnica se chama *dropout*, e é frequentemente utilizada para melhorar a performance de modelos de classificação [13].

Após o treino, na etapa de inferência onde o classificador faz predições, o *dropout* não é aplicado.

2.4.4 Redes Convolucionais

As redes MLP sofrem algumas limitações, que deterioram seu desempenho quando são aumentadas indiscriminadamente, a fim de se tentar identificar padrões mais complexos. Como todos os neurônios de uma camada estão conectados a todos os neurônios da camada seguinte, isso rapidamente aumenta o número de pesos sinápticos da rede conforme novas camadas são adicionadas. Por sua vez, este grande número de pesos não só torna o treino mais custoso computacionalmente, mas também dificulta consideravelmente a convergência, favorecendo o *overfit*.

Dessa forma, havia especial dificuldade em se desenvolver aplicações de processamento de imagens e visão computacional por redes neurais, dado que imagens frequentemente têm dimensões grandes como *input*. Por exemplo, para uma imagem de 100 por 100 pixels em modo RGB, há 30000 atributos de entrada a serem passados para a rede. Para se reduzir o tamanho da entrada, a abordagem tradicional consistia em se extrair manualmente atributos (*hand-*

crafted features) destas imagens por meio do conhecimento de especialistas, e posteriormente passar estes atributos para as MLPs.

Apenas na década de 2000 a 2010, os primeiros modelos de redes neurais com camadas especializadas para extrair *features* de imagens foram publicados [27]. Tanto a LeNet [27] quando a AlexNet [28] superaram o desempenho de classificadores baseados em atributos *handcrafted*, estabelecendo esta nova classe de redes neurais como modelos padrão para classificação de imagens.

As camadas ou blocos extratores de atributos presentes nestas redes são baseadas principalmente na operação de convolução (discutida nas próximas subseções). Por este motivo, estas redes são conhecidas como Redes Neurais Convolucionais (*Convolutional Neural Network* – CNN).

Em 2012, a AlexNet de Krizhevsky et al. [28] teve um desempenho que constituía o estado da arte para a competição ILSRVC (*ImageNet Large Scale Visual Recognition Challenge*), um dos principais eventos de classificação de imagens, que usa o *dataset* Imagenet. Este *dataset* é uma referência para a área de desenvolvimento de classificadores de imagens por possuir dezenas de milhares de classes e mais de 10 milhões de imagens [29]. Frequentemente, este *dataset* é usado como *benchmark* de novas arquiteturas de CNN. Modelos pré-treinados no Imagenet também são frequentemente fornecidos em *frameworks* de *deep learning*, para tarefas de *transfer learning* [17].

2.4.4.1 Convolução

A equação que rege a operação de convolução, conforme descrita por Goodfellow et al. [14], é dada por:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2-17}$$

onde I é a imagem sendo estudada, K é chamado de filtro ou *kernel* de convolução, i e j são as coordenadas de cada pixel, e m e n são as coordenadas dos pixels da área onde a convolução está sendo aplicada.

A operação de convolução é muitas vezes denotada por um asterisco, como visto acima. Por sua vez, o *output* de uma operação de convolução é muitas vezes chamado de mapa de atributos ou *feature map*.

Um exemplo ilustrativo da equação (2-17) pode ser visto na Figura 2.5.

O uso de filtros que varrem a imagem e geram mapas de atributos permite focar em padrões da imagem localmente. Entre outros motivos, esta é uma das principais vantagens das redes convolucionais comparadas com o uso de MLPs diretamente em tarefas de classificações de imagens, pois as informações locais

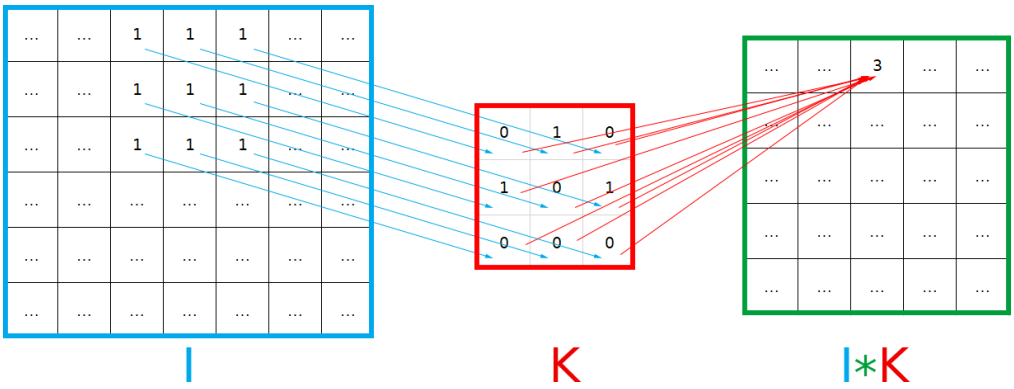


Figura 2.5: Exemplo de uma operação de convolução. Comparável com a equação (2-17), usando $m = 3, 4, 5$ e $n = 1, 2, 3$.

são perdidas quando a informação das entradas chega nas camadas densamente conectadas, onde todos os neurônios estão conectados com todos da camada seguinte.

Como visto na Figura 2.5, a operação de convolução em geral implica que o mapa de atributos tem dimensões menores do que a a imagem original.

Uma das formas de controlar as dimensões do mapa de atributos é adicionar *pixels* de valor nulo ao redor da imagem, como um preenchimento, também conhecido como "*padding*". Adicionalmente também é possivel alterar o tamanho do passo que o filtro de convolução toma ao varrer a imagem. Este passo também é conhecido como "*stride*". A Figura 5.15 mostra um exemplo onde não é usado *padding*, e onde o *stride* tem valor 1, significando que o filtro varre a imagem com movimentos *pixel* a *pixel*.

De certa forma, é possível interpretar as camadas de convolução de uma CNN como camadas capazes de aprender os melhores atributos a serem extraídos da imagem, a fim de permitir que as camadas densamente conectadas classifiquem com bom desempenho. Na prática, se demonstrou que essa abordagem supera a obtenção manual de atributos [30].

2.4.4.2 *Pooling*

Frequentemente, as operações de convolução são associadas com operações de *pooling*, que reduzem o tamanho dos mapas de atributos, possibilitando que a rede aprenda uma hierarquia espacial entre os padrões visuais da imagem [13].

Embora conceitualmente parecidas com a camada de convolução, as camadas de *pooling* usam um elemento, normalmente de tamanho $2x2$ ou $3x3$, que varre a imagem e retorna o valor máximo ou médio dentro da janela de

varredura. Nestes casos, a operação é conhecida como *Max-pooling* e *Average-pooling*, respectivamente.

Ainda de acordo com Chollet [13], a redução do tamanho do mapa de atributos, antes do bloco de camadas densamente conectadas, contribui para uma redução do *overfit*, uma vez que o número de conexões e pesos sinápticos antes da camada densamente conectada é drasticamente reduzido.

2.4.4.3 *Batch Normalization*

Camadas de normalização em *batch* (*Batch Normalization* – BN) surgiram em 2015 [31]. Os autores da técnica notaram que, no processo de treinamento por *mini-batches*, frequentemente, a cada *batch*, a atualização de pesos flutuava de acordo com a composição aleatória das amostras que compõem o *batch*.

Por sua vez, ao se criar um tipo de camada que normaliza cada *batch* em tempo real, conforme eles são apresentados às camadas, notou-se que a convergência e o tempo de treino melhoraram, para uma série de aplicações. Dessa forma, estas camadas se tornaram uma adição frequente em arquiteturas mais recentes de redes convolucionais, usadas em problemas de classificação de imagem.

Para um batch $\mathcal{B}$ com N valores $x = x_1, x_2, ..., x_N$, o BN corresponde à Equação (2-18):

$$BN_{\gamma,\beta}(\hat{x}_i) = \gamma\hat{x}_i + \beta \tag{2-18}$$

onde γ e β são parâmetros aprendidos pela rede ao longo do treino e $\hat{x}_i$ corresponde ao valor normalizado de um elemento x_i do batch.

O valor $\hat{x}_i$, por sua vez é obtido através da Equação (2-19):

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{2-19}$$

sendo $\mu_{\mathcal{B}}$ e $\sigma_{\mathcal{B}}^2$ a média e a variância daquele batch (atualizadas a cada batch durante o treino). Adicionalmente, ϵ é uma constante utilizada para estabilidade numérica. A notação aqui utilizada segue a publicação original da abordagem de BN por Ioffee e Szegedy [31].

Com base nesta discussão sobre os principais tipos de camadas presentes em CNNs, é possível abordar em mais detalhes as principais arquiteturas utilizadas nos dias de hoje para classificações de imagens.

2.4.4.4 *Flatten*

O formato da saída da rede neural depende tanto do formato dos dados de entrada, quanto das operações matemáticas realizadas ao longo da passagem pela rede.

Conforme os dados passam pelas operações de convolução e *pooling*, o formato da matriz vai sendo alterado até chegar ao bloco densamente conectado, para classificação. Este bloco espera uma entrada unidimensional, porém frequentemente os dados estão em um formato multidimensional.

A camada *flatten* é uma transformação que reorganiza a matriz de dados para um formato unidimensional.

2.4.5 VGG

As arquiteturas VGG [22], do *Visual Geometry Group* da Universidade de Oxford, foram as vencedoras do concurso *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) de 2014. São inspiradas no mesmo princípio da AlexNet e LeNet, possuindo blocos de convolução para extração de atributos e blocos densamente conectados para classificação.

As redes VGG são comumente usadas para tarefas de classificação de imagens, sendo compostas por 5 blocos de convolução e um bloco densamente conectado. As convoluções nos sucessivos blocos usam, respectivamente, 64, 128, 256, 512 e 512 filtros de convolução. As arquiteturas originais tinham 11, 13, 16 e 19 camadas ajustáveis de peso, sendo as duas últimas (frequentemente conhecidas como VGG16 e VGG19) as mais utilizadas na literatura.

2.4.6 ResNet

Uma das dificuldades encontradas ao treinar redes progressivamente mais profundas, para tarefas complexas de classificação de imagens, é que após um certo número de camadas o desempenho parecia estagnar ou piorar [32].

Teoricamente, para uma rede convolucional com muitos blocos de convolução, seria esperado que a extração de atributos ideais acontecesse em certos blocos de convolução, e os demais blocos, que não estivessem contribuindo diretamente, apenas aprendessem a operação de identidade, sem prejudicar o desempenho da rede. Porém, na prática, isto não ocorre e a convergência do treino é prejudicada pela presença dos blocos excedentes. Possivelmente este comportamento é causado por *overfitting* da quantidade excedente de pesos, entre outros fatores.

Dessa forma, He et al. propuseram usar *skip-connections* entre blocos de convolução [32]. As *skip-connections* são a função de identidade com uma função adicional que a rede aprende. Caso o bloco em questão não esteja contribuindo, é mais fácil para a rede identificar isso e usar a operação de identidade para "ignorar" aquele bloco. A função adicional é também chamada de função residual, motivo pelos quais estas redes são conhecidas como redes residuais, ou ResNets.

Com a estratégia das *skip connections* He et al. conseguiram superar por muito o número de camadas das arquiteturas VGG16 e VGG19, chegando a treinar redes com até 152 camadas no *dataset* Imagenet, e com até 1000 camadas no *dataset* CIFAR10 [32]. Em 2016, as ResNets venceram a competição ILSVRC.

A Figura 2.6 mostra uma comparação das arquiteturas do tipo VGG19 e ResNet50. Devido à grande quantidade de camadas das ResNet, é difícil representar o diagrama de blocos do modelo completo em uma imagem. Dessa forma, mostrou-se apenas a parte mais essencial da rede, o bloco convolucional com a *skip-connection*, na Figura 2.6. Note que a *skip-connection* pode ser vista como a seta direita no bloco convolucional da ResNet50, representado na Figura 2.6.

2.4.7 Outras arquiteturas

2.4.7.1 Model1

Aqui está descrita uma arquitetura adicional utilizada para o estudo de caso com imagens de metástases, apresentado na Seção 5.1.2.

O objetivo dos estudos conduzidos nesta tese para o *dataset* de metástases era gerar explicações para uma CNN com bom desempenho de classificação, ao invés de otimizar e treinar arquiteturas CNN para esta tarefa a partir do zero. Dessa forma, qualquer classificador de alto desempenho existente poderia ser usado para essa finalidade.

Consequentemente, os modelos publicamente disponíveis na competição baseada em P-CAM [33] foram avaliados quanto ao seu desempenho, com base na métrica AUC (vide Seção 2.2.3.6). Ademais, tendo em vista que os experimentos executados neste projeto foram implementados em Tensorflow/Keras 2.0 [17], buscou-se identificar um modelo de CNN também implementado em Keras, disponível publicamente e com alto valor de AUC [33].

Dessa forma, selecionou-se a rede associada à submissão pública do

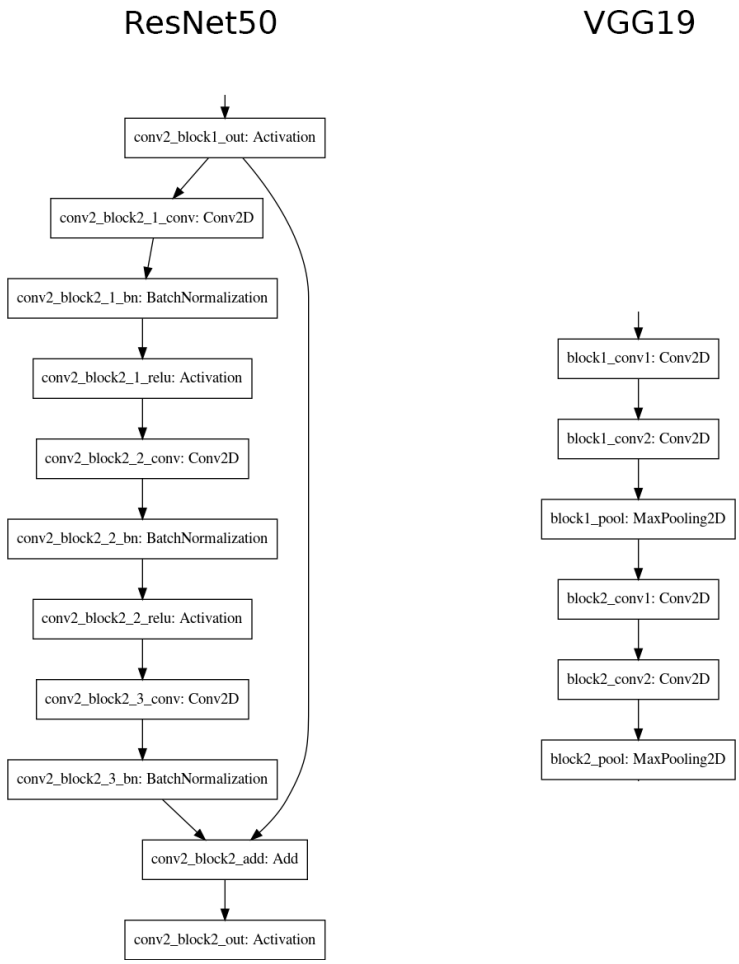


Figura 2.6: Exemplos de blocos convolucionais das redes VGG19 e ResNet50. Vale notar a *skip-connection* e as camadas de *batch normalization* na rede ResNet, comparado com a VGG19.

usuário *Marsh* para a competição. Mais precisamente, o arquivo h5 com os pesos treinados da CNN foi utilizado e o modelo foi então reimplementado usando esses pesos para fins de execução dos experimentos.

Como este modelo é uma arquitetura customizada, aparentemente sem um nome específico, ele será denominado "Model1" ao longo deste manuscrito. Esse modelo consiste em uma CNN com AUC de 0,9528 para o conjunto de teste público.

O "Model1" é uma rede neural convolucional com as seguintes camadas do Keras:

- Input Layer (96x96x3)
- Convolution layer (32 filtros, kernels 3×3 , Ativação ReLU)
- Convolution layer (32 filtros, kernels 3×3, Ativação ReLU)
- Convolution layer (32 filtros, kernels 3×3, Ativação ReLU)
- Max pooling (pooling 2×2)

- Dropout (30%)
- Convolution layer (64 filtros, kernels 3×3, Ativação ReLU)
- Convolution layer (64 filtros, kernels 3×3, Ativação ReLU)
- Convolution layer (64 filtros, kernels 3×3, Ativação ReLU)
- Max pooling (pooling 2×2)
- Convolution layer (128 filtros, kernels 3×3 Ativação ReLU)
- Convolution layer (128 filtros, kernels 3×3, Ativação ReLU)
- Convolution layer (128 filtros, kernels 3×3, Ativação ReLU)
- Max pooling (pooling 2×2)
- Flatten
- Dense (256 neurônios, Ativação ReLU)
- Dropout (30%)
- Dense (2 neurônios, Ativação *softmax*)

Consequentemente, nota-se que o "Model1" é uma arquitetura convolucional com três blocos de convolução e um bloco denso de classificação, similar às arquiteturas tradicionais como VGG.

2.4.8 Transfer Learning

Em muitos casos, treinar uma rede neural do zero, a partir de pesos inicializados aleatoriamente, pode ser uma tarefa custosa computacionalmente, além de requerer bastante *fine-tuning* de parâmetros [34]. Este problema se intensifica especialmente para as arquiteturas mais profundas, como aquelas discutidas nas seções anteriores.

A fim de se contornar este problema, uma estratégia comumente utilizada com redes neurais se chama *transfer learning*. Esta abordagem consiste em tomar uma rede já treinada, para uma tarefa mais geral, com muitos dados, e ajustar apenas alguns dos pesos, ou todos em etapas, para uma nova tarefa com menos dados disponíveis.

Na área de classificação de imagens, a ideia é que a rede tenha aprendido a abstrair conceitos fundamentais de processamento de imagens a partir de um *dataset* maior e mais variado, podendo então ser apenas *fine-tuned* para um *dataset* menor e mais específico, onde só as camadas de classificação são treinadas enquanto o resto do modelo é congelado, ou onde se "descongelam" as camadas sucessivamente da última (classificação) até a primeira (entrada).

2.5 Framework

Atualmente, existe uma série de ferramentas disponíveis para implementação destes modelos, em diferentes linguagens computacionais. Por sua vez, as baseadas na linguagem Python são as mais usadas na literatura, sendo as duas principais biliotecas/*frameworks* chamadas pyTorch [35] e Tensorflow [17].

Os códigos desta tese foram desenvolvidos na versão mais recente de Tensorflow 2.0, usando principalmente a API (*Application Programming Interface*) de alto nível chamada Keras. Os códigos foram desenvolvidos em Python 3.8, e as bibliotecas utilizadas são sempre as versões mais atuais.

3
Explainable AI

Neste capítulo são delineados os principais conceitos de Inteligência Artificial Explicável, incluindo uma descrição detalhada das técnicas de XAI utilizadas neste projeto e uma discussão sobre suas vantagens e desvantagens.

Os modelos de classificação discutidos no capítulo anterior, baseados em redes neurais convolucionais, têm o melhor desempenho em geral para tarefas de classificação de imagens. No entanto, muitos outros tipos de modelos podem ser utilizados para este fim, incluindo árvores de decisão [36], *Random Forests* [37], árvores com *gradient boosting* [38], *Support Vector Machines* (SVM) [39], regressões lineares, entre outros.

Entretanto, ressalta-se que desde o advento das redes convolucionais, estes outros modelos têm sido cada vez menos utilizados para classificação de imagens, apesar de ainda serem frequentemente aplicados em outras áreas. Como estes outros modelos não foram aplicados nesta tese, uma descrição mais detalhada deles ficaria fora do escopo do texto. No entanto, convém notar que há uma característica fundamental que varia entre todas estas diferentes abordagens: a explicabilidade do modelo.

Em certos modelos é mais fácil compreender a influência das variáveis de entrada em uma dada saída, enquanto que em outros o modelo atua como uma caixa preta, ou *black box*, em que o processamento dos dados não produz uma visualização ou representação clara para o usuário. Redes neurais, com sua grande complexidade e número de parâmetros ajustáveis, estão na classe de modelos do tipo caixa preta.

Frequentemente, existe um *trade-off* entre a capacidade de interpretar um modelo e seu desempenho. Modelos mais complexos conseguem capturar representações mais ricas dos dados, mas a interpretação dessas representações se distancia de algo compreensível em termos humanos. Essa é a questão fundamental da área de pesquisa de Inteligência Artificial Explicável.

Ressalta-se que, dentro desta área, há um movimento crescente na literatura com intuito de diferenciar dois termos básicos: interpretabilidade e explicabilidade. Segundo as definições de Gilpin et al. [40] e Arrieta el al. [10], tem-se:

– Interpretabilidade – Termo mais abrangente, vagamente definido como a

ciência da compreensão sobre o que um modelo fez (ou pode ter feito). Descrição dos processos internos de um modelo em termos compreensíveis para um humano [40]. Habilidade de explicar ou providenciar significado em termos compreensíveis para um humano [10].

– Explicabilidade – Se refere a modelos capazes de sumarizar as razões para seu próprio comportamento, ganhar a confiança de usuários, ou produzir perspectivas sobre as causas de suas decisões [40]. Está associada com a noção de uma explicação como uma interface entre humanos e um tomador de decisões que, ao mesmo tempo, é um *proxy* preciso do tomador de decisões, e é compreensível para humanos [10].

Ainda segundo [40], todos os modelos que são explicáveis são interpretáveis, mas o oposto não é necessariamente verdadeiro.

Alguns autores, como no caso de Arrieta et al. [10], ainda definem termos adicionais como inteligibilidade (*understandability/intelligibility*), compreensibilidade (*comprehensibility*), transparência, entre outros.

No entanto, vale notar que estas definições ainda não foram universalmente adotadas, sendo comum que estes termos sejam usados de forma intercambiável, ou com significados que variam sutilmente, entre diferentes trabalhos na literatura científica. De forma geral, contudo, todos os trabalhos têm a preocupação central de gerar modelos com decisões mais facilmente compreensíveis, quebrando o comportamento de caixa preta. Em particular, nesta tese, os modelos caixa preta em questão são as CNNs discutidas no capítulo anterior.

Dado este preâmbulo, é importante notar que técnicas de extração de conhecimento (*knowledge exctraction* e outras abordagens similares para aumentar a explicabilidade de sistemas de IA existem há décadas, em formas básicas [9]. Porém, se entende de forma geral que a área de Inteligência Artificial Explicável teve um renascimento considerável nos últimos 5 a 10 anos, acompanhando os desenvolvimentos mais recentes da área de *deep learning* [9, 10]. A maioria das técnicas de XAI utilizadas para classificação de imagens foi criada a partir de 2016.

Entre 2014 e 2020 o número de publicações e novas técnicas nessa área aumentou consistentemente. Algumas revisões sistemáticas nos últimos anos tentaram fazer uma taxonomia destas técnicas, dividindo-as em alguns tipos básicos [9, 10]. Arrieta et al. [10] fazem uma divisão primária entre técnicas agnósticas ao modelo e técnicas específicas para modelos específicos.

Técnicas agnósticas podem ser aplicadas a qualquer tipo de classificador, por exemplo, sejam estes redes neurais ou SVMs. Por sua vez, técnicas

específicas podem ter sua aplicação limitada a determinados aspectos da estrutura de um dado modelo, não sendo aplicáveis para outros modelos.

Por sua vez, dentro do conjunto de técnicas agnósticas a modelo, uma das categorias mais representativas são técnicas que perturbam a entrada do modelo, a fim de analisar a influência destas perturbações no comportamento das saídas do modelo. Destaca-se que, duas das técnicas utilizadas nesta tese pertencem a esta categoria: *Local Interpretable Model-Agnostic Explanations* (LIME) [41] e *Randomized Input Sampling for Explanation* (RISE) [42]. Ademais, as novas técnicas propostas no Capítulo 4, EvEx e Squaregrid, também são técnicas de perturbação agnósticas ao modelo.

Por outro lado, dentro do conjunto de técnicas para modelos específicos, há aquelas que focam em redes convolucionais. Dentre estas, se diferenciam técnicas de visualização por mapas de ativação e técnicas de saliência. As demais técnicas utilizadas neste projeto (Vanilla Gardients, GradCAM, Smoothgrad e Integrated Gradients) pertencem a estas categorias, e serão discutidas nas seções seguintes.

Assim como no caso das terminologias discutidas anteriormente nesta seção, esta taxonomia também pode apresentar pequenas variações entre diferentes referências. Entretanto, isso não afeta o entendimento desta tese, pois abaixo está feita uma descrição mais detalhada de cada uma destas abordagens e classes de técnicas de XAI.

3.1 Técnicas baseadas em perturbação

De forma simplificada, o princípio de funcionamento das técnicas baseadas em perturbação consiste em substituir parte de uma imagem por pixels de outro tipo. A substituição mais comum é trocar o grupo de pixels em questão pelo valor 0, que nas codificações mais usuais corresponde à cor preta. Note que, de forma geral, isso não requer acesso aos mecanismos internos do modelo analisado, ou informações específicas sobre que tipo de classificador se está utilizando.

A principal técnica representativa desta classe, com maior número de citações na literatura e *forks* no seu repositório público específico, se chama LIME [10, 41]. A sigla é uma abreviação para *Local Interpretable Model-Agnostic Explanations*. Outras duas técnicas representativas desta classe são: RISE (*Randomized Input Sampling for Explanation*) [42] e *Meaningful Perturbations* [43]. É importante ressaltar que todas estas técnicas são bastante recentes, tendo sido publicadas entre 2016 e 2019. Consequentemente, muitas vezes, suas implementações estão sendo constantemente atualizadas e melhoradas, sendo que

ainda podem ter limitações.

Também é interessante definir aqui a característica de técnicas agnósticas ao *framework* de implementação. Embora não seja um aspecto fundamental do ponto de vista teórico, na prática se percebeu ao longo do desenvolvimento deste trabalho que a implementação da técnica de XAI pode ser extremamente dependente de funções ou aspectos específicos de um *framework*.

Por exemplo, até o presente momento do desenvolvimento desta tese, a implementação da técnica *Meaningful Perturbations* só está disponível para redes com arquitetura VGG19 implementadas em pyTorch, sendo sua re-implementação em tensorflow não trivial. No caso da técnica RISE, a re-implementação é mais direta, e a técnica LIME é facilmente aplicável para qualquer tipo de *framework*, desde que haja uma função de ativação que forneça uma saída *softmax*. Pode-se dizer que LIME é a única técnica que, além de ser agnóstica ao modelo, é também agnóstica ao *framework* de implementação do modelo.

As técnicas LIME e RISE serão descritas nas subseções seguintes deste capítulo. Demais técnicas de explicação por perturbação de *inputs*, que não tenham implementações no *framework* utilizado nesta tese, como a técnica *Meaningful Perturbations*, ou não sejam aplicáveis para problemas de imagem, não serão discutidas nesta sessão, por não terem sido estudadas no escopo deste trabalho.

Vale ressaltar que estas técnicas específicas foram escolhidas como linha de frente para este problema devido a sua universalidade e quantidade mínima e/ou inexistente de pressuposições sobre os modelos classificadores, e seus componentes internos.

3.1.1 Algoritmos de Segmentação

Conforme discutido anteriormente, as explicações via LIME envolvem perturbar as imagens de entrada de um classificador. Para realizar essas perturbações, no caso desta técnica, é necessário dividir a imagem em sub-regiões que serão cobertas pelas perturbações.

Como será discutido em mais detalhes na Subseção 3.1.2, tais subregiões são chamadas na literatura de superpixels ou segmentos, por agruparem pixels que possuam padrões de texturas ou cores com algum nível de similaridade. A ideia é que estas regiões sejam representativas de conceitos visuais importantes para a classificação.

Para possibilitar a divisão da imagem em superpixels, é necessário utilizar um algoritmo de segmentação. Os três algoritmos incluídos na implementação

atual do repositório público da LIME [44] são: Felzenswalb [45], SLIC [46] e Quickshift [47]. A seguir, as principais características de cada um deles são detalhadas.

3.1.1.1 Felzenszwalb

O algoritmo de segmentação Felzenszwalb [45], também conhecido como Algoritmo de segmentação de Felzenszwalb e Huttenlocher (*Felzenszwalb Huttenlocher Algorithm* – FHA), é uma técnica baseada em teoria de grafos para divisão de uma imagem em superpixels. Ressalta-se ainda que dos três algoritmos fornecidos na implementação pública de LIME [44, 48], o FHA é o mais rápido.

O algoritmo de segmentação se baseia na definição de uma métrica de comparação entre regiões da imagem, candidatas a serem superpixels da segmentação final. Na publicação original de Felzenszwalb e Huttenlocher [45] estas regiões são chamadas de componentes. Por sua vez, para duas componentes arbitrárias, *C1* e *C2*, a métrica de comparação é expressa por:

$$D(C_1, C_2) = \begin{cases} \text{verdadeiro se} & \text{Dif}(C_1, C_2) > M\text{Int}(C_1, C_2) \\ \text{falso em outro caso} \end{cases} \tag{3-1}$$

onde $D(C_1, C_2)$ diz se existe uma borda entre C_1 e C_2, $\text{Dif}(C_1, C_2)$ é a diferença mínima entre as duas componentes, e $M\text{Int}(C_1, C_2)$ é uma grandeza chamada *diferença interna mínima*, definida como:

$$M\text{Int}(C_1, C_2) = min(\text{Int}(C_1) + \tau(C_1), \text{Int}(C_2) + \tau(C_2)) \tag{3-2}$$

onde $\text{Int}(C)$ é a *diferença interna* de uma componente, definida como o menor peso de uma métrica para grafos conhecida como árvore de extensão mínima (*Minimum Spanning Tree* – MST) da componente C.

Uma interpretação direta deste processo como função destas métricas não é trivial, porém, pela Eq. (3-1), nota-se que o algoritmo busca distinguir se duas componentes têm uma borda entre si ou não, avaliando se a diferença mínima entre essas componentes supera valores referentes às diferenças internas das componentes.

Ademais, é possível notar que a variável τ, na Eq. (3-2), influencia a segmentação. Conforme τ aumenta, aumenta-se o limiar necessário para que duas regiões sejam consideradas distintas. Isso contribui para que mais componentes sejam aglutinadas em um mesmo superpixel, reduzindo a quantidade total de superpixels.

Segundo Felzenszwalb e Huttenlocher [45], este é um fator importante, pois não é confiável usar a métrica MST para superpixels muito pequenos. Dessa forma, a variável τ é definida como uma função do tamanho da componente/segmento C:

$$\tau(C) = k/|C| \tag{3-3}$$

onde k é uma constante e $|C|$ é o tamanho da componente.

Conforme descrito no artigo original [45], a variável k define um nível de observação, onde valores altos de k criam uma preferência por superpixels maiores, ao mesmo tempo que reduzem a quantidade total de superpixels.

Com base nas definições do algoritmo FHA, a variável k é a única que permite controlar o tamanho e quantidade de superpixels da segmentação FHA. Entretanto, note que este controle se dá de maneira indireta.

A implementação deste algoritmo na biblioteca *skimage* [48] chama este parâmetro k de *scale*. Na prática, a interpretação deste parâmetro, bem como o equacionamento subjacente, não são triviais, motivo pelo qual a própria biblioteca recomenda testar diferentes valores até encontrar um mais apropriado.

Adicionalmente, além dos parâmetros mencionados no artigo original [45], a implementação da biblioteca *skimage* possui mais dois parâmetros ajustáveis para o algoritmo FHA:

1. *sigma* – Largura de um *kernel* Gaussiano aplicado à imagem.

2. *min_size* – Tamanho do menor superpixel, expresso em pixels. Forçado via pós-processamento da imagem, após a aplicação das equações de FHA.

Dessa forma, observa-se que existem três parâmetros ajustáveis associados a este tipo de segmentação: k, *sigma* e *min_size*.

3.1.1.2 Quickshift

O algoritmo Quickshift, assim como o SLIC descrito na Seção 3.1.1.3, baseiam-se em técnicas de *clustering* para realizar a segmentação. Mais especificamente, no caso do Quickshift, trata-se de uma segmentação de uma imagem em modo RGB, onde os *clusters* são identificados num espaço $5D$, composto pelas três dimensões de cor e duas dimensões espaciais dos pixels nas imagens [47, 49].

O algoritmo computa uma floresta de pixels da imagem, onde cada ramo é identificado com um valor de distância. Cada segmento corresponde então

a uma sub-árvore [49]. Os superpixels mais úteis são identificados cortando aqueles segmentos com uma distância acima de um dado limiar, que pode ser definido pelo usuário ou por algum tipo de validação cruzada.

De acordo com [49], para um dado pixel (x,y) numa imagem $I(x, y)$) , o algoritmo Quickshift considera $(x, y, I(x, y))$ como uma amostra de um espaço vetorial de dimensão $d+2$, sendo d a dimensão de $I(x, y)$). Em seguida, calcula a estimativa da densidade de probabilidade por Janelas de Parzen, usando um *kernel* gaussiano de desvio padrão σ:

$$P(x, y, I(x, y)) = \sum_{x'y'} \frac{1}{(2\pi\sigma)^{d+2}} \exp\left(-\frac{1}{2\sigma^2}\begin{bmatrix} x - x' \\ y - y' \\ I(x, y) - I(x', y') \end{bmatrix}\right) \quad (3\text{-}4)$$

Na sequência, o algoritmo constrói uma arvore conectando cada pixel (x,y) da imagem ao seu vizinho mais próximo que tenha um valor maior de densidade. Formalmente, tem-se $(x', y') > P(x, y)$ se, e somente se [49]:

$$P(x', y', I(x', y')) > P(x, y, I(x, y)) \quad (3\text{-}5)$$

Cada pixel (x,y) é então conectado ao pixel parente de maior densidade e mais próximo, com a distância mínima dada por:

$$\min_{(x',y')>P(x,y)} \left((x - x')^2 + (y - y')^2 + \|I(x, y) - I(x', y')\|_2^2\right) \quad (3\text{-}6)$$

Na prática, existem 5 parâmetros ajustáveis para o algoritmo Quickshift presentes na função disponível na biblioteca *skimage* [48], adotados para a segmentação de imagens no repositório LIME [44]. São eles:

1. *ratio* – Define o peso dado para proximidade entre pixels no espaço de cores, ou no espaço da imagem;

2. *kernel_size* – Largura do *kernel* Gaussiano usado para suavizar as amostras;

3. *max_dist* – Limiar para distâncias entre dados;

4. *sigma* – Largura para pré-processamento Gaussiano;

5. *convert2lab* – Define se a imagem deve ser convertida para o espaço de cores *Lab* antes da segmentação. Assume que a imagem original está no espaço RGB.

3.1.1.3 SLIC

O algoritmo de segmentação de *clustering* linear iterativo simples (*Simple Linear Iterative Clustering* – SLIC) guarda similaridade com o Quickshift por usar técnicas de *cluster* e representar a imagem em um espaço *5D*, que combina as dimensões espaciais da imagem com as cores em espaço *Lab*.

Seguindo a notação de [46], que é a referência original do método, o algoritmo toma K superpixels aproximadamente do mesmo tamanho, em uma imagem de N pixels. A métrica de distância utilizada, D_s, para diferenciar pixels, é dada por:

$$D_s = d_{lab} + \frac{m}{S} d_{xy} \tag{3-7}$$

onde d_{lab} e d_{xy} são as normas euclidianas calculadas, respectivamente, no espaço de cores e no espaço xy da imagem; S é o intervalo do *grid* inicial de K superpixels que o algoritmo gera no seu primeiro passo; e m é uma variável que dita o quão compacto é um superpixel.

Note que, quanto maior for m, maior a proximidade espacial é enfatizada, fazendo com que o *cluster* torne-se mais compacto [46]. O algoritmo realiza um agrupamento do tipo *k-means*[46, 48] até convergir para os superpixels finais.

Vale notar também que, embora o artigo original do SLIC diga que é possível controlar o algoritmo com uma única variável (possivelmente N ou m), em usos típicos, a implementação da biblioteca *skimage* [48] utiliza ao menos 8 variáveis, que podem influenciar a geração dos superpixels. Mais especificamente, ao se utilizar este algoritmo de segmentação na biblioteca *skimage*, o usuário terá 14 parâmetros ajustáveis, apesar de alguns não serem diretamente relacionados com o algoritmo SLIC e sim com o k-means subjacente ou detalhes sobre *data-types* em Python.

Achanta et al. [46] discutem o fato de que a presença de muitos parâmetros ajustáveis dificulta o uso e interpretação dos mesmos para a tarefa de segmentação. Esta dificuldade os motivou a desenvolverem um algoritmo de segmentação, análogo ao SLIC, mas que pode ser teoricamente regido por apenas um parâmetro.

Os parâmetros da função SLIC na biblioteca *skimage*, relativos à quantidade e tamanho dos superpixels, são [48]:

1. *n_segments* – O número aproximado de superpixels da segmentação final. Corresponde ao parâmetro *N*, conforme discutido previamente;

2. *compactness* – Balanceia a proximidade na dimensão das cores e na dimensão espacial. Corresponde ao parâmetro m na Eq.(3-7);

3. *sigma* – Largura do *kernel* Gaussiano usado no pré-processamento;

4. *spacing* – O tamanho do *voxel*[1] de espaçamento em cada dimensão. Controla os pesos das distâncias durante o *k-means*. O espaçamento padrão é uniforme;

5. *convert2lab* – Define se a imagem deve ser convertida para o espaço de cores *Lab* antes da segmentação. Assume que a imagem original está no espaço RGB;

6. $enforce\_connectivity$ – Define se as regiões dos superpixels precisam ser contínuas ou não;

7. min_size_factor – Tamanho mínimo dos superpixels;

8. max_size_factor – Tamanho máximo dos superpixels.

3.1.2 LIME

A técnica LIME, descrita pela primeira vez em 2016 por Ribeiro et al. [41], pode ser usada não só para imagens, mas também para classificadores do tipo *black-box* em geral.

A formulação mais geral desta técnica, seguindo a notação e definições originais de Ribeiro et al. [41], é dada por:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \quad \mathcal{L}(f, g, \pi_x) + \Omega g \tag{3-8}$$

O termo g indica uma das potenciais explicações para um modelo, e G são modelos inerentemente interpretáveis em potencial, tais como modelos lineares, árvores de decisão ou listas de regras. A complexidade da explicação $g \in G$ é definida por $\Omega(g)$, sendo específica para cada tipo de modelo. Ribeiro et al. [41] referem-se a $\Omega(g)$ como sendo a profundidade de uma árvore de decisão, ou o número de pesos diferentes de zero em um modelo linear, para exemplificar.

Por sua vez, o modelo sendo explicado é denotado por *f*, sendo que em uma tarefa de classificação *f(x)* representa a probabilidade de que um exemplo ou instância x pertença a alguma dada classe. Por outro lado, $\pi_x(z)$ representa alguma métrica de distância entre duas instâncias z e x. Ademais, $\mathcal{L}(f, g, \pi_x)$ representa uma medida da infidelidade da explicação g em aproximar o modelo *f*, na localidade definida por π_x.

[1] *Voxel* é a generalização de *pixel* para dimensões maiores

Com base nestas definições, note que a Eq. (3-8) representa a minimização da infidelidade explicativa de um modelo substituto (*surrogate model*) em uma localidade ao redor de uma dada instância a ser explicada.

Ribeiro et al. [41] mencionam ainda que diferentes famílias de explicações G, funções de fidelidade $\mathcal{L}$ e métricas de complexidade Ω podem ser utilizadas, porém ressaltam que o foco da técnica é a escolha de modelos lineares como explicações e a busca do melhor modelo através de perturbações.

3.1.3 Explicações Para Imagens

O equacionamento do LIME apresentado acima é bastante geral, sendo conveniente traduzi-lo para termos práticos associados à aplicação desejada desta tese: classificadores de imagens.

Para entender em detalhes como a implementação do algoritmo funciona é necessário destrinchar o código fonte do repositório público [41], visto que esta informação não está sumarizada em nenhuma outra referência.

A primeira etapa do algoritmo envolve segmentar a imagem, que tipicamente é um *array* com as três dimensões RGB associadas às cores de cada pixel, usando algum dos algoritmos de segmentação descritos anteriormente. O algoritmo de segmentação padrão da biblioteca utilizada é o Quickshift. Os superpixels resultantes são armazenados como índices inteiros de 0 à $n_{superpixels} - 1$, onde $n_{superpixels}$ denota a quantidade total de superpixels.

Em seguida, uma cópia da imagem é criada e coberta por uma cor que será usada na perturbação. A cor padrão é preta, podendo também serem utilizadas outras opções de cores sólidas, ou a média das cores dentro de cada superpixel.

Na sequência, são geradas imagens perturbadas, com base no número de imagens definido previamente pelo usuário, como um parâmetro passado ao LIME. Para cada imagem perturbada é gerado um *array* de zeros e uns, cujo comprimento é igual ao número de superpixels da imagem. Este *array* é gerado aleatoriamente com a função *randint* da biblioteca NumPy [50], que usa uma distribuição uniforme discreta. Neste *array*, valores iguais a zero indicam para o algoritmo quais superpixels da imagem cobrir com a perturbação.

Todos os *arrays* como este, para cada imagem perturbada, ficam armazenados em uma variável, que no código fonte é chamada de *data*, notação esta que será também adotada para facilitar o restante da explicação do algoritmo.

Uma vez feito isso, as imagens perturbadas são apresentadas ao classificador (a CNN), que gera predições na forma de probabilidades *softmax*, armazenadas separadamente em outra variável (no código original [44], *preds*).

O algoritmo então calcula as distâncias entre a imagem original e as imagens perturbadas comparando os *arrays* dentro da variável *data* com um *array* do mesmo tamanho contendo apenas uns, sem zeros, ou seja, sem perturbações. A função de comparação é a *similaridade por cosseno*, de modo que, para dois arrays a e b, tem-se:

$$\cos_{\text{similarity}}(a, b) = \frac{a \cdot b}{\|a\|\|b\|} \tag{3-9}$$

Note que a função *similaridade por cosseno* é dada pelo produto vetorial entre os *arrays* normalizado pela norma *L2*.

O algoritmo então treina um classificador linear, usando os dados da vizinhança local gerados anteriormente (armazenados na variável *data*) e as *(*labels) geradas a partir das predições da CNN nos dados perturbados, ponderados pelas distâncias estimadas com a similaridade por cosseno. O modelo linear adotado é uma regressão linear do tipo *ridge regression* [44].

Finalmente, os coeficientes do modelo linear são retornados e usados para gerar um *heatmap*, onde o coeficiente correspondente a cada superpixel indica se ele contribuiu positivamente ou negativamente para a classificação e quão grande essa influencia foi. Ao longo desta tese, estes coeficientes serão denominados *pesos explicativos*, *w*. Um sumário simplificado deste processo pode ser visto no Algoritmo 1, e na Figura 3.1.

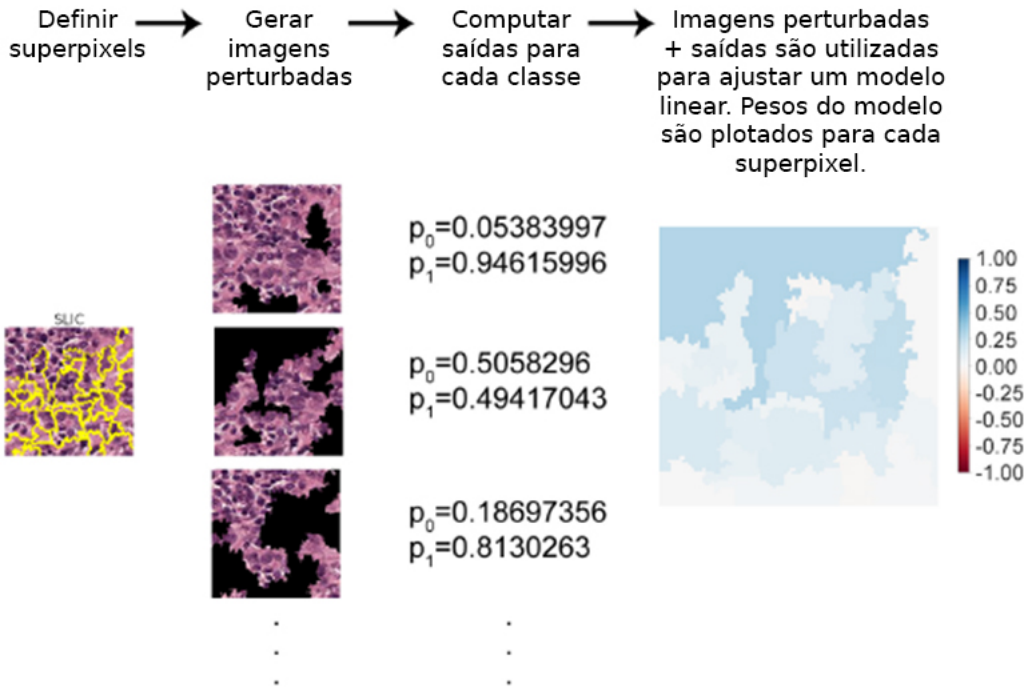


Figura 3.1: Diagrama resumido da técnica LIME. Uma imagem cuja classificação se deseja explicar é dividida em superpixels, e uma distribuição de imagens perturbadas estocasticamente é gerada. Esta distribuição é apresentada ao modelo classificador, cujas probabilidades de classificação são usadas para ajustar um modelo linear. Os pesos deste modelo linear mostram quanto cada superpixel contribui a favor ou contra a classificação, e isso pode ser plotado como um *heatmap*. Adaptado de [51]

Algoritmo 1: Explicar classificação de uma CNN

Dados: Imagem,CNN
Resultado: Gerar um *heatmap* explicativo
segmentar imagem;
gerar vizinhança perturbada da imagem;
para *Imagem com vizinhança perturbada* **faça**
  predizer *Imagem* com CNN;
  armazenar probabilidades *softmax*;
fim
treinar modelo linear substituto;
gerar *heatmap* com pesos do modelo linear;

3.1.4 RISE

Em 2018, Petsiuk et al. [42] publicaram outra abordagem de explicação por perturbação das entradas, chamada *Randomized Input Sampling for Expla-*

nation (RISE). Similarmente à técnica LIME, RISE também é uma abordagem agnóstica a modelo, que pode ser aplicada a qualquer classificador *black-box* de imagens. Entretanto, existem duas diferenças principais entre estas técnicas. Ao invés de perturbar superpixels e usar modelos substitutos para aproximar as explicações localmente, a técnica RISE baseia-se no uso de máscaras geradas aleatoriamente para cobrir a imagem original [42]. Cada máscara $\mathcal{M}$ recebe um peso que corresponde à probabilidade *softmax* da classificação da imagem $\mathcal{I}$ perturbada pela máscara, $\mathcal{I}_{perturbada} = \mathcal{I} \odot \mathcal{M}$, onde o símbolo $\odot$ denota uma multiplicação termo a termo entre os *arrays*.

Dessa forma, o *heatmap* final, $\mathcal{H}$ é dado por [42]:

$$\mathcal{H}_{RISE} \underset{MC}{\approx} \frac{1}{p_1 N} \sum_{i}^{N} f(\mathcal{I} \odot \mathcal{M}_i)\mathcal{M}_i \tag{3-10}$$

onde o somatório é definido sobre todas as N máscaras geradas, f é o modelo *black-box* que se deseja explicar (no caso, a CNN), e p_1 é a probabilidade de que um pixel seja preservado pelas máscaras.

A soma das N máscaras é aproximada via Monte Carlo (MC), com a geração estocástica das máscaras. Na prática, o usuário pode ajustar os valores de p_1 e N conforme desejado.

Petsiuk et al. argumentam que gerar máscaras pixel por pixel, no tamanho da imagem original, é impraticável computacionalmente, podendo acarretar efeitos adversos indesejáveis. Portanto, sugerem gerar máscaras pequenas e fazer *upsampling* bilinear até o tamanho desejado [42]. A Figura 3.2 mostra este processo.

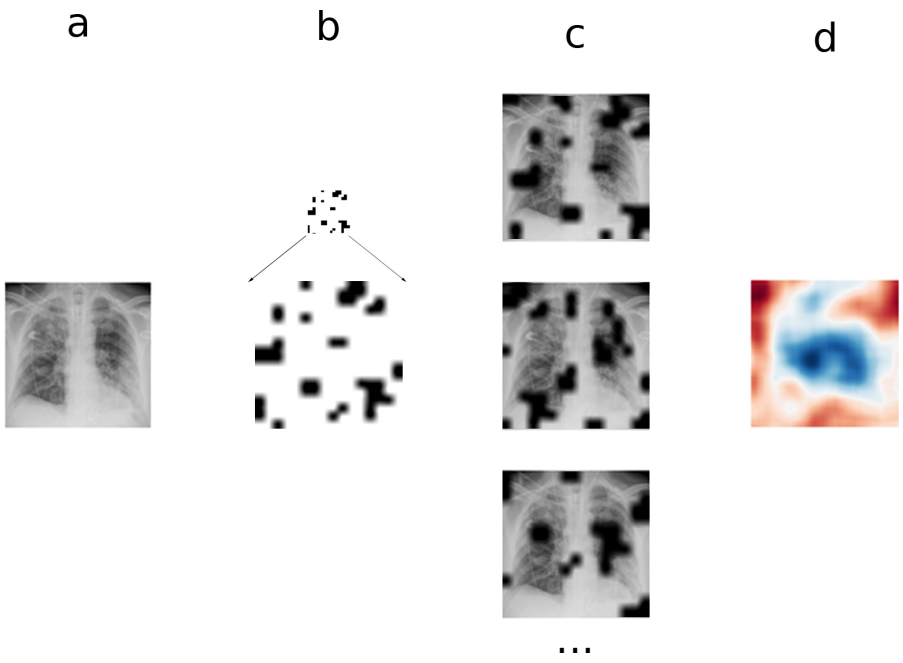


Figura 3.2: Diagrama simplificado da técnica RISE. Começando com uma imagem cuja classificação se deseja explicar (painel (a)), máscaras de perturbação são geradas, primeiro gerando aleatoriamente pequenas máscaras que são então redimensionadas para o tamanho total da imagem, como visto no painel (b). Em seguida as várias imagens perturbadas (painel (c)) são apresentadas ao modelo classificador, e as probabilidades de classificação são utilizadas como pesos em uma soma ponderada das máscaras para obter o *heatmap* final (painel (d)).

3.2 Técnicas baseadas em saliência

Diferente das abordagens discutidas até aqui, baseadas em perturbações, uma outra classe de técnicas, em geral chamadas de técnicas de saliência ou atribuição, visam associar cada pixel da entrada à sua influência na classificação final. Dessa forma, os *heatmaps* produzidos por técnicas de saliência frequentemente têm detalhes mais finos do que aqueles produzidos por técnicas de perturbação.

Para ajudar a selecionar técnicas candidatas para este estudo, utilizaram-se as técnicas de saliência mais citadas e utilizadas na literatura de XAI. Estas estão descritas nas subseções seguintes. Contudo, destaca-se que a nomenclatura referente às diferentes subclasses destas técnicas pode variar sutilmente entre diferentes referências. Vale também ressaltar que a utilização destas técnicas específicas não exclui o estudo futuro de outras variantes de técnicas de Saliência.

A Figura 3.3 mostra um esquema simplificado das técnicas de gradiente discutidas nesta seção. A técnica GradCAM é representada em uma figura separada por partir de um princípio diferente, onde o gradiente não é calculado da camada da camada de saída até a camada de entrada , mas somente até a última camada convolucional.

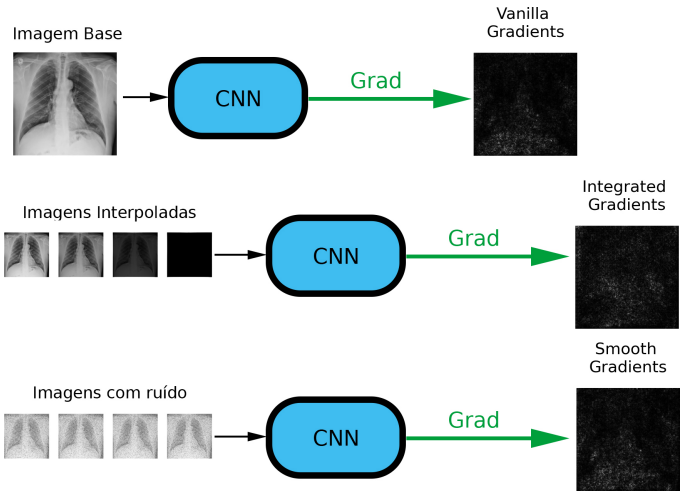


Figura 3.3: Sumário visual das técnicas baseadas em gradiente puro. Todas estas técnicas calculam o gradiente da saída até a entrada da rede neural. A principal diferença é como os dados de entrada são tratados. Na *Vanilla Gradients* se utiliza somente a imagem base. *Smooth Gradients* utiliza versões da imagem perturbadas com ruído, e *Integrated Gradients* utiliza a interpolação da imagem original com uma imagem *baseline* que o classificador não saiba classificar (em geral uma imagem preta). A seta verde representa a computação do gradiente, e o heatmap final é apresentado do lado direito.

3.2.1 GradCAM

A principal técnica de saliência, com mais citações e uso em aplicações de XAI, é conhecida como *Gradient-weighted Class Activation Maps* – GradCAM, tendo sido publicada pela primeira vez em 2016 por Selvaraju et al [52].

A GradCAM é uma técnica de saliência baseada em mapas de ativação, visto que baseia-se na ativação da última camada antes do bloco de classificação, para tentar gerar uma visualização. O princípio básico da GradCAM consiste em calcular o gradiente de uma dada saída de interesse da rede neural (normalmente a classe cuja classificação se deseja explicar), e computar o gradiente desse neurônio de saída específico de volta para a última camada de convolução, pois considera-se que essa é a camada que contém a abstração

mais detalhada dos padrões que formam a imagem, de forma que ela seria a com maior potencial para gerar a visualização mais explicativa [52].

Dessa forma, para uma dada classe c, tem-se que:

$$\alpha_c^k = \frac{1}{Z} \sum \nabla_k y^c \tag{3-11}$$

onde y^c representa a saída específica de uma classe c da rede, k representa um mapa de atributos da última camada convolucional da CNN, Z é o número total de pixels desse mapa de atributos e o somatório é definido sobre as dimensões desse mapa. Por sua vez, α_c^k é um número que representa a influência ou peso de um dos mapas de atributos k sobre o neurônio de saída c.

O somatório sobre todos os pixels do mapa de atributos dividido pelo número de total de tais pixels também é conhecido por *global average pooling*. Segundo Selvaraju et al. [52], a escolha do *global average pooling* ao invés de outras formas de tomar as médias, ou pooling dos dados, foi feita empiricamente por produzir os melhores resultados.

O *heatmap* é então dado pela combinação linear dos mapas de atributo, cujos pesos são os valores α_c^k calculados na Eq. (3-12).

$$\mathcal{H}_{GradCAM} = \text{ReLU}(\sum_k \alpha_c^k A^k) \tag{3-12}$$

Selvaraju et al. usam a ativação ReLU para eliminar contribuições negativas à classe de interesse, mencionando que pixels negativos provavelmente pertencem a outras classes da imagem.

Note que o *heatmap* gerado tem as dimensões do mapa de atributos da última camada convolucional, que em geral é menor do que a imagem original. Consequentemente, o *heatmap* é redimensionado para o tamanho da imagem original via upsampling bilinear [52].

Este processo está representado visualmente na Figura 3.4.

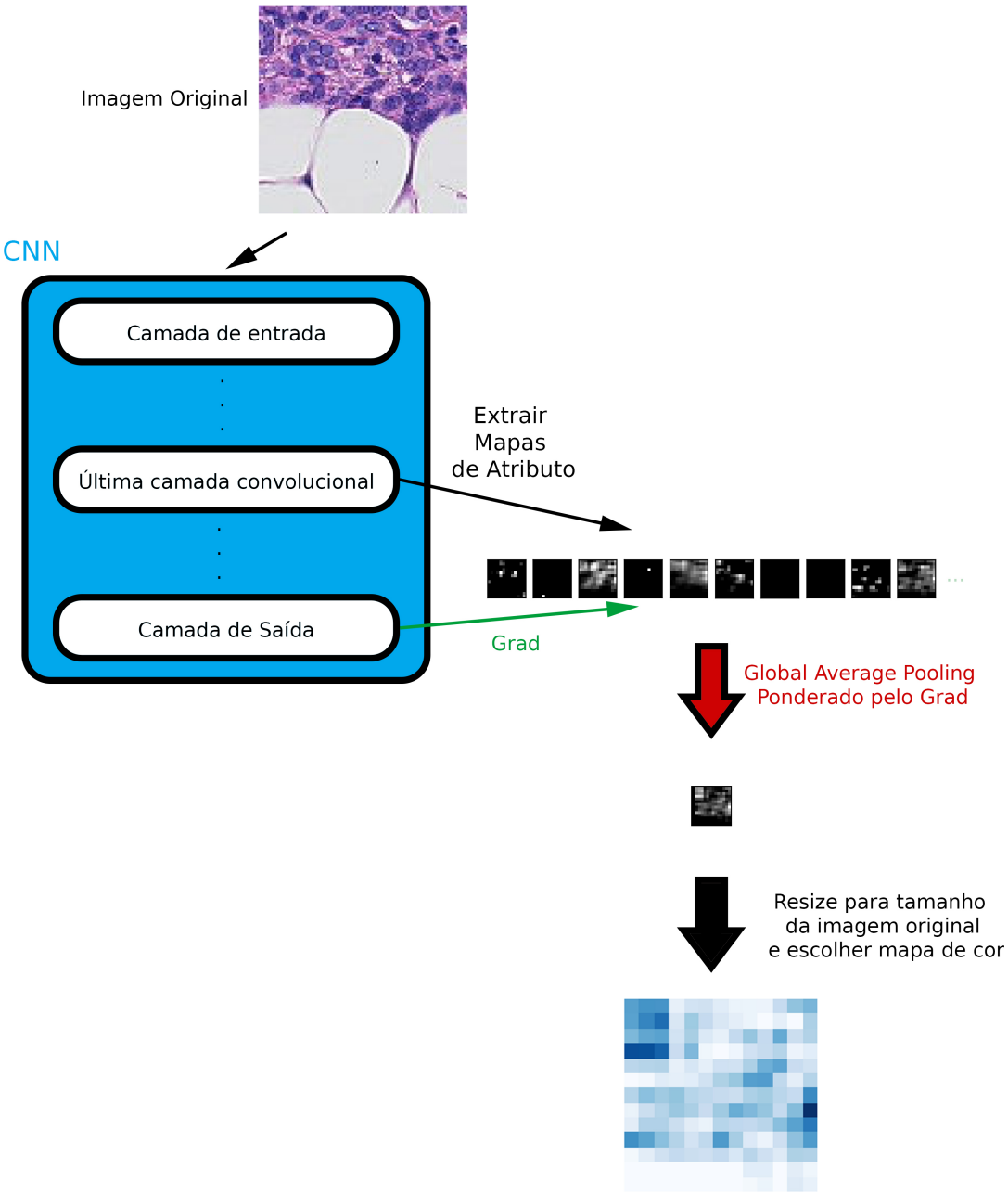


Figura 3.4: Diagrama da técnica GradCAM. Uma imagem é classificada por uma CNN, e os mapas de atributo da última camada convolucional são obtidos. O gradiente da saída em relação à última camada convolucional é computado, e os mapas de atributo são combinados via *Global Average Pooling*, utilizando os gradientes como peso ponderador. O resultado é um heatmap do tamanho dos mapas de atributo, que normalmente são menores que a imagem original. O passo final é redimensionar e colorir este *heatmap*.

3.2.2 Vanilla Gradients

As últimas três técnicas a serem apresentadas aqui também são técnicas de gradientes, mas estas, ao contrário da GradCAM, calculam o gradiente da saída até a camada de entrada, propagando a informação de volta através da arquitetura inteira.

A abordagem mais simples neste âmbito consiste em calcular o gradiente puro da camada de saída, propagado de volta até cada pixel da entrada. Esta ideia é comumente conhecida como *Vanilla Gradients* na literatura atual, tendo sido originalmente publicada por Simonyan et al. em 2013 [53].

Notavelmente, *heatmaps* gerados com esta técnica não permitem discernir se um dado pixel contribuiu a favor ou contra a classificação, sendo possível apenas avaliar se o pixel contribuiu para ela.

3.2.3 Smooth Gradients

Em 2017, uma versão mais sofisticada das explicações por gradiente foi proposta por Smilkov et al. [54]. Estes autores mencionam que, tipicamente, os *heatmaps* gerados via *Vanilla Gradients* apresentam muito ruído, provavelmente causado por flutuações locais nas derivadas das ativações de saída. No intuito de suavizar essas flutuações, propuseram calcular os gradientes para várias versões da mesma imagem, com ruído gaussiano aplicado estocasticamente.

Os resultados obtidos parecem mostrar qualitativamente, dada a falta de uma métrica quantitativa para fazer essa comparação, que as explicações efetivamente passam a apresentar menos ruído usando esta abordagem em relação às obtidas com o uso da *Vanilla Gradients*.

3.2.4 Integrated Gradients

Também em 2017, Sundararajan et al. propuseram outra forma de melhorar o método de *Vanilla Gradients* [55].

Neste caso, a proposta dos autores consiste em criar uma imagem que serve como *baseline* (normalmente uma imagem completamente preta), e interpolá-la com a imagem original que se deseja explicar, calculando o gradiente para cada passo da interpolação e integrando ao final. Neste contexto, a interpolação significa gerar várias versões da imagem com diferentes níveis de transparência, indo de uma imagem totalmente preta até a imagem original.

Segundo Sundararajan et al., este processo satisfaz uma série de axiomas matemáticos sobre sensibilidade e invariância, o que justifica a adoção deste método do ponto de vista teórico. Os autores ainda estabelecem que um número de passos de interpolação entre 20 e 300 é tipicamente suficiente [55].

4
Técnicas Propostas

Alguns dos resultados originais desta tese envolvem aplicar as técnicas descritas no capítulo anterior visando a analisar o comportamento de CNNs na classificação de certos *datasets* de imagens médicas. Conforme discutido anteriormente, havia uma lacuna na literatura relacionada a este tema em particular, que esta tese visa a suprir.

Também houve a ideia e motivação de contribuir para o estado da arte das técnicas de XAI, de forma geral. Entre as técnicas existentes, conforme discutido no Capítulo 3, há algumas limitações chave que ficaram progressivamente mais evidentes conforme o projeto foi desenvolvido.

Este foi notavelmente o caso para a técnica LIME, que apesar de ser a mais universalmente acessível e aplicável para diferentes modelos em diferentes *frameworks*, requer uma quantidade bastante significativa de ajustes finos de parâmetros, a fim de se aprimorar a qualidade das explicações geradas. Estes aspectos serão evidenciados e discutidos em detalhes nos resultados, apresentados no próximo capítulo.

Por sua vez, o capítulo atual apresenta e discute as propostas originais feitas neste projeto, para geração de explicações que corrigem alguns dos problemas enfrentados com a técnica LIME, enquanto ainda preservando o aspecto agnóstico a modelo e agnóstico a *framework*.

Especificamente, as técnicas aqui propostas visam solucionar os problemas de instabilidade das explicações de LIME para parâmetros similares, bem como criar metodologias consistentes para ajuste de parâmetros, ou remover a necessidade desse ajuste através de aproximações com técnicas sem parâmetros.

As duas técnicas propostas se chamam *Squaregrid* e Evolved Explanations (EvEx), descritas a seguir.

4.1
Squaregrid

Um dos problemas chave com a aplicação da LIME, de forma geral, é a seleção dos parâmetros para os algoritmos de segmentação, visto que fazer um ajuste manual consome um tempo excessivo. A Figura 4.1 mostra um exemplo

claro deste problema. Supondo que tenhamos uma imagem que se deseja segmentar, visando gerar explicações via LIME, podemos começar usando os parâmetros padrão da técnica FHA. Variando o parâmetro sigma, como visto na figura, há pequenas modificações na forma de alguns dos numerosos superpixels gerados.

Não é trivial tentar prever como serão essas modificações ou os seus efeitos nas explicações LIME. Como contraponto, vemos, à direita da Figura 4.1, um exemplo de segmentação para a mesma imagem, usando parâmetros otimizados via uma abordagem proposta neste capítulo. Não há um caminho trivial ou intuitivo para partir dos parâmetros padrão e chegar nesses parâmetros otimizados manualmente. O espaço de busca tem, como estimativa baixa, milhões de possíveis combinações de parâmetros.

Algo que torna a questão ainda mais complexa é que certos parâmetros podem ter maior ou menor influência dependendo das características da imagem. Em testes preliminares, para um mesmo dataset, havia tanto imagens cujas segmentações eram sensíveis ao parâmetro scale, quanto outras menos sensíveis. Não é trivial dizer que tipo de imagem será sensível a que parâmetro, ou como isso pode variar com as diferentes combinações destes parâmetros.

Adicionalmente, também é necessário definir qual é um bom conjunto de superpixels, ou quais perturbações são mais representativas e permitem gerar melhores explicações.

Na Seção 4.2, abordaremos uma nova técnica proposta nesta tese, que visa a solucionar estes problemas de forma mais abrangente. No entanto, inicialmente, implementou-se uma abordagem mais simples que tentar eliminar a necessidade de ajustar parâmetros de segmentação, abandonando a ideia de superpixels e substituindo-os por um conjunto de grades fixas. Como estas grades (*grids*) são compostas por quadrados, todos com as mesmas dimensão,

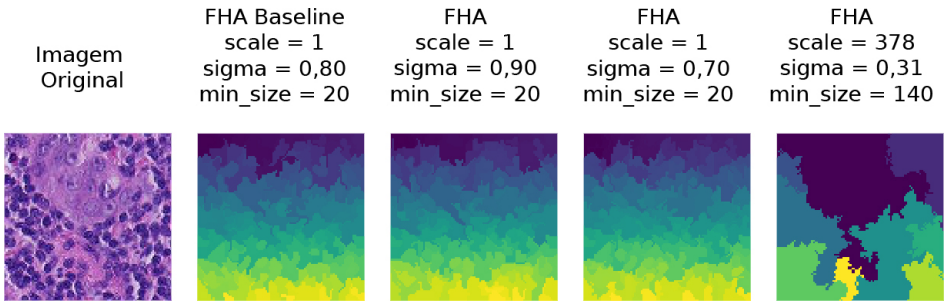


Figura 4.1: Exemplos da influência dos parâmetros na segmentação Felzenszwalb (FHA). Temos uma imagem *baseline* com os parâmetros padrão do FHA, e outras variando o parâmetro sigma, mostrando pequenas mudanças. O último exemplo, da direita, mostra a mesma imagem segmentada com parâmetros otimizados através da técnica EvEx, discutida neste capítulo.

nomeou-se esta técnica como *Squaregrid*.

A ideia básica da técnica *Squaregrid* está representada na Figura 4.2. Inicialmente neste projeto se utilizaram grades cujos quadrados tem tamanhos que são divisores inteiros das dimensões da imagem. Contudo isso depende muito dessas dimensões. Este é o caso visto na imagem.

Para criar uma abordagem mais geral que maximiza o número de quadrados na grade, a imagem é dividida em *grids* com quadrados cujos tamanhos são potências de 2. A escolha da base 2 visa maximizar o número de *grids* que se podem encaixar dentro do espaço da imagem. Estes grids são centralizados com a imagem, de forma que qualquer *offset* nas dimensões fique localizado nas bordas, pois nos *datasets* avaliados, de forma geral, as regiões de interesse estão no centro da imagem. Figuras com este esquema de divisão podem ser vistas no Capítulo 5.

A maior potência de 2 utilizada é a primeira a superar as dimensões da imagem. A menor potência de 2 é limitada pelo custo computacional que se deseje utilizar. Na prática, conforme discutido adiante nos resultados, é desnecessário utilizar grades excessivamente finas pois o poder explicativo de cada quadrado da grade diminui consideravelmente.

Inicialmente, para evitar divisões não inteiras do tamanho da imagem pela grade, pensou-se em usar somente tamanhos de grade que sejam divisores inteiros das dimensões da imagem. Contudo isso se torna muito dependende das dimensões do input, e a abordagem acima descrita maximiza o número de quadrados nas grades de forma mais universal e independente das dimensões da imagem.

Ressalta-se que os quadrados não seguem a ideia dos superpixels, pois não necessariamente dividem a imagem em padrões consistentes ou similares de texturas e cores. No entanto, é possível aplicar esta técnica sem ajuste de parâmetros para qualquer imagem.

Por fim, os *heatmaps* das explicações são gerados via LIME, usando as Squaregrids ao invés dos algoritmos de segmentação convencionais, anteriormente discutidos. Isto pode ser visto na parte inferior da Figura 4.2.

4.2 EvEx

Ao invés de simplesmente dividir a imagem em um *grid* quadrado, outra alternativa proposta envolve utilizar superpixels que sejam úteis ou representativos, porém automatizando o processo de ajuste destes superpixels. Para tanto, é necessário definir algumas métricas relativas à qualidade da

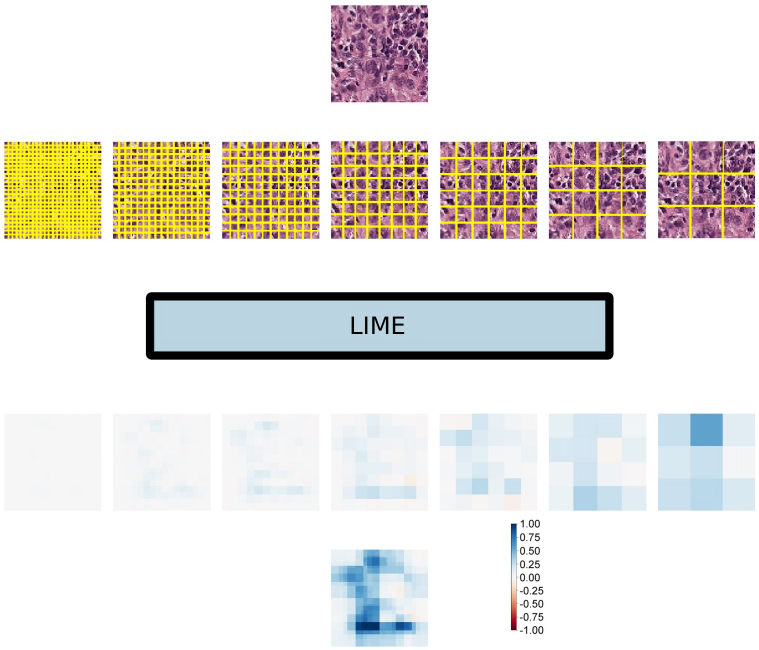


Figura 4.2: Diagrama da técnica *Squaregrid*. Na parte superior da figura é possível ver a imagem original e as diferentes grades quadradas que atuam como *superpixels*. Na parte inferior da figura estão os *heatmaps* correspondentes, gerados via LIME. Adaptado de [51].

segmentação e da explicação, além de escolher um algoritmo de otimização. Nesta seção esses aspectos são descritos em maior detalhe.

Um panorama geral da técnica EvEx proposta nesta tese pode ser sumarizado como: otimizar os parâmetros de segmentação FHA através de um algoritmo genético multiobjetivo (*Multi-Objective Genetic Algorithm* – MOGA), e tomar a explicação final como a média da frente de Pareto contendo as melhores explicações. A fim de melhor ilustra as características desta técnica, a Figura 4.3 mostra um diagrama de blocos composto pelos diversos elementos que compõem a técnica EvEx.

Essencialmente, há nove blocos constituintes da técnica:

1. Um classificador recebe a imagem e gera as probabilidades *softmax* de classificação.

2. Um algoritmo de segmentação divide a imagem em *superpixels*; sementes aleatórias são utilizadas para garantir a consistencia em uma dada rodada do código.

3. A técnica LIME é utilizada para gerar os pesos explicativos.

4. A algoritmo NSGA-II é usado para otimizar os parâmetros da técnica de segmentação.

5. uma frente de Pareto com as melhores explicações a cada geração é formada.

6. Um critério de parada precoce determina se a frente de Pareto não foi modificada acima de um dado limiar de gerações.

7. Um conjunto de métricas e objetivos direciona o processo evolutivo.

8. Um *heatmap* final é gerado com a média da frente de Pareto.

Embora a técnica EvEx permita uma escolha genérica e livre do algoritmo de segmentação, a abordagem FHA foi utilizada para os estudos de caso desta tese. Esta escolha foi feita de maneira empírica, como ficará mais claro na discussão dos estudos de caso abordados no próximo capítulo. Basicamente, para os *datasets* médicos estudados, esta foi a segmentação que forneceu explicações mais próximas das segmentações médicas feitas por humanos, além de ter os pesos explicativos mais altos nos *heatmaps* gerados. Uma vantagem adicional da segmentação FHA, embora não seja o motivo principal de sua escolha, é que esta segmentação é mais rápida que as demais técnicas utilizadas para geração de superpixels (Quickshift e SLIC), bem como a que tem menos parâmetros ajustáveis, conforme descrito na Subseção 3.1.1.1.

4.2.1 Algoritmo Genético Multi-Objetivo

Como visto anteriormente, a segmentação FHA tem três parâmetros ajustáveis: *scale*, *sigma* e *min_size*. Embora sejam apenas três parâmetros, o espaço de busca formado por todas as combinações cria um custo computacional excessivo para uma busca exaustiva. Adicionalmente, estes parâmetros tem influência drástica sobre o *heatmap* resultante.

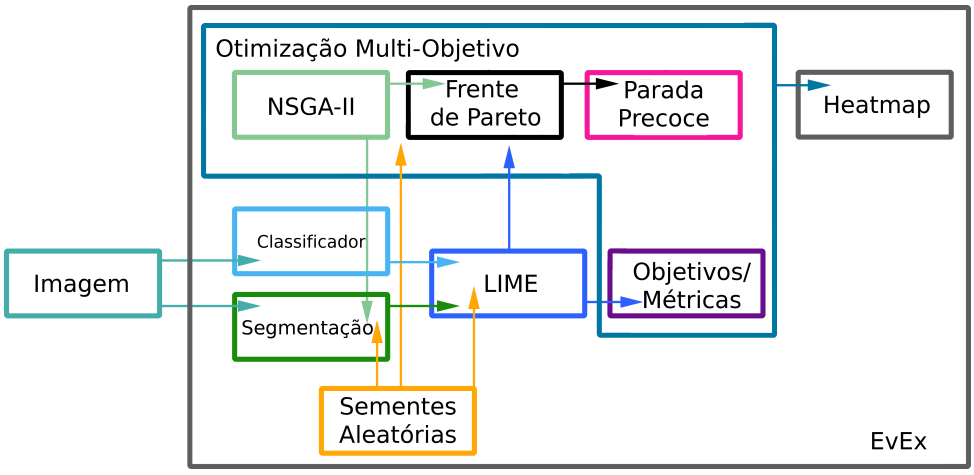


Figura 4.3: Diagrama da técnica EvEx. Adaptado de [56].

Resultados iniciais (ver Seção 5.1) de buscas manuais demonstraram que a segmentação FHA produzia superpixels com área relativa (a_r) grande (ocupando grande parte da imagem) e pesos explicativos (w) altos. Porém, não era possível otimizar os parâmetros apenas buscando maximizar w, dado que isso favorecia explicações com um único superpixel, que cobria toda a imagem. Ressalta-se, contudo, que estes aspectos são discutidos em mais detalhes na Seção 5.1, sendo mencionados aqui apenas para justificar a adoção das métricas abaixo. A quantidade a_r também é definida mais formalmente na equação (4-1), mais adiante.

A tarefa em questão consiste em explicar a classificação de uma imagem $\mathcal{I}$ por um modelo f. Esta imagem é segmentada por um algoritmo com parâmetros ajustáveis θ em N segmentos, ou superpixels, $s_i = [s_1, s_2, ..., s_N]$. Cada superpixel, após gerada a explicação via LIME, tem um peso explicativo associado, $w_i = [w_1, w_2, ..., w_N]$.

Por sua vez, dentre os superpixels criados, o superpixel s_L tem o maior peso explicativo, w_L, entre todos os superpixels. Note que a métrica usada para a otimização deve buscar maximizar w_L. Por sua vez, ressalta-se que considerando-se os coeficientes do ajuste linear descrito na Subseção 3.1.2, os valores de w_L estão no intervalo $[0, 1]$, atingindo tipicamente valores ao redor de 0,6 a 0,8 ao final da otimização.

Entretanto, não se pode avaliar w_L isoladamente, visto que a área ocupada por s_L na imagem também é um aspecto relevante, a fim de se tentar evitar a seleção de superpixels que simplesmente cubram a imagem inteira. De modo a fazer com que o parâmetro associado à área do superpixel também fique contido no intervalo $[0, 1]$, criou-se uma métrica associada à área relativa, a_r, definida por:

$$a_r = \frac{a_{s_L}}{a_{\mathcal{I}}} \tag{4-1}$$

onde a_{s_L} é a área do superpíxel de maior peso explicativo, e $a_{\mathcal{I}}$ a área da imagem.

Dessa forma, o segundo objetivo do MOGA é minimizar a_r.

Ademais, a biblioteca LIME inclui uma medida de qualidade da explicação final [44], chamada de *explanation score*. Uma análise do código fonte revela que este score é o valor R^2 da regressão linear, o qual também está contido no intervalo $[0, 1]$. Dessa forma, o MOGA tem um terceiro objetivo associado à maximização deste parâmetro. Ao longo deste trabalho, o *explanation score* será referido por E_s.

Em resumo, a tarefa de otimização consiste em buscar a combinação de parâmetros *scale*, min_{size} e *sigma* que impliquem simultaneamente na

maximização de w_L e E_s e na minimização de a_r.

4.2.1.1 NSGA-II

O algoritmo selecionado para fazer a otimização multi-objetivo discutida na seção anterior foi o *Non-dominated Sorting Genetic Algorithm* (NSGA-II) [57], por ser um dos algoritmos mais tradicionais para este tipo de problema evolutivo multi-objetivo. Para problemas deste tipo, é difícil ou impossível definir uma única solução como sendo a melhor, porque podem existir diversos *trade-offs* possíveis entre os diferentes objetivos sendo otimizados.

Mais especificamente, para o caso específico aqui discutido, pode-se imaginar uma explicação onde, a princípio, é possível reduzir a_r sem ter prejuízos no valor de w_L. Porém, após certo ponto, qualquer redução adicional de a_r incorrerá na diminuição do valor de w_L. Esta condição, onde um dos objetivos não pode ser otimizado sem incorrer em prejuízos para outros, se chama *eficiência/otimização de Pareto*. O conjunto de pontos nesta condição crítica forma uma "frente" no espaço de soluções do problema, conhecida como frente de Pareto [58].

Na terminologia de algoritmos genéticos, cada solução é chamada de *indivíduo*, e seus atributos são denominados genes. O algoritmo NSGA-II cria frentes de Pareto inicializando os genes dos indivíduos de uma população aleatoriamente, e fazendo um *sorting* que visa atingir a eficiência de pareto para aquela dada população nos multi-objetivos definidos. Adicionalmente, o algoritmo usa uma métrica de distanciamento que cria uma preferência por indivíduos que não sejam muito similares ao grupo e ainda tenham bom desempenho na frente, visando a aumentar a diversidade da população.

Em outras palavras, a saída final do algoritmo NSGA-II é uma frente de soluções, ou, neste caso, explicações/*heatmaps* $\mathcal{H}_i = [\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_{N_P}]$, sendo N_P o número de indivíduos presentes na fronteira de Pareto ao final do processo evolutivo.

Por sua vez, O *heatmap* explicativo final, $\mathcal{H}_{EvEx}$, é definido como a média aritmética dos *heatmaps* associados aos indivíduos presentes na frente de Pareto:

$$\mathcal{H}_{EvEx} = \frac{1}{N_P} \sum_{i=1}^{N_P} \mathcal{H}_i \tag{4-2}$$

A justificativa para o calculo da média dos *heatmaps* presentes na frente de Pareto é que a presença de múltiplas explicações potencialmente válidas ou boas coincide até mesmo com o que ocorre normalmente para explicações humanas. Ademais, buscar a intersecção entre essas explicações igualmente

válidas (ao menos no sentido de eficiência de Pareto) pode revelar áreas de interesse.

4.2.2 Hipervolume

Se cada solução para uma otimização multi-objetivo (*Multi Objective Optimization* – MOO) for considerada um ponto em um espaço n-dimensional, um indicador comum de desempenho do MOO é o volume n-dimensional contido no conjunto de soluções [59]. Ou seja, o espaço contido na região delimitada pelos pontos de solução. Essa métrica é chamada de hipervolume (HV).

Dessa forma, como o MOO aqui tratado possui 3 objetivos, tem-se que o HV será o volume tridimensional contido nos conjuntos de solução, que neste caso são as frentes de Pareto. Esta métrica foi adotada para acompanhar o desempenho do GA, de modo que o HV é calculado para as frentes de Pareto encontradas em cada geração.

Neste intuito, ressalta-se que a função de computação HV descrita por Fonseca et al. [60] foi utilizada neste trabalho. Entretanto, esta função HV assume implicitamente que o MOO visa a minimizar todos os objetivos. Para a_r não há mudança necessária, pois a formulação do problema aqui tratado já objetiva sua minimização. No entanto, note que E_s e w_L devem ser maximizados. Portanto, a formulação do problema de otimização foi ajustada, fazendo com que as funções MOO e HV considerem o problema de minimização de $1 - E_s$, $1 - w_L$ e a_r. Dessa forma, o HV é calculado usando o ponto (1,1,1) como referência, o que significa que (1, 1, 1) corresponderia à pior solução possível, e (0,0,0) à melhor, para todos os objetivos. A Figura 4.4 mostra este processo de forma mais visual, para um exemplo simplificado bidimensional.

Note que o hiper-volume é usado apenas para observar como a frente de Pareto está evoluindo, não sendo usado para interferir ou influenciar no processo de evolução de nenhuma forma. Ao se monitorar HV ao longo do processo evolutivo, espera-se observar o valor de HV aumentando conforme o processo de evolução ocorre, com a frente de Pareto se distanciando gradualmente do ponto (1,1,1) em direção à vizinhança de (0,0,0).

4.2.3 Early Stop

Durante os primeiros testes com o EvEx, os GAs foram configurados para rodar por várias gerações (neste caso 200), mas frequentemente convergiam

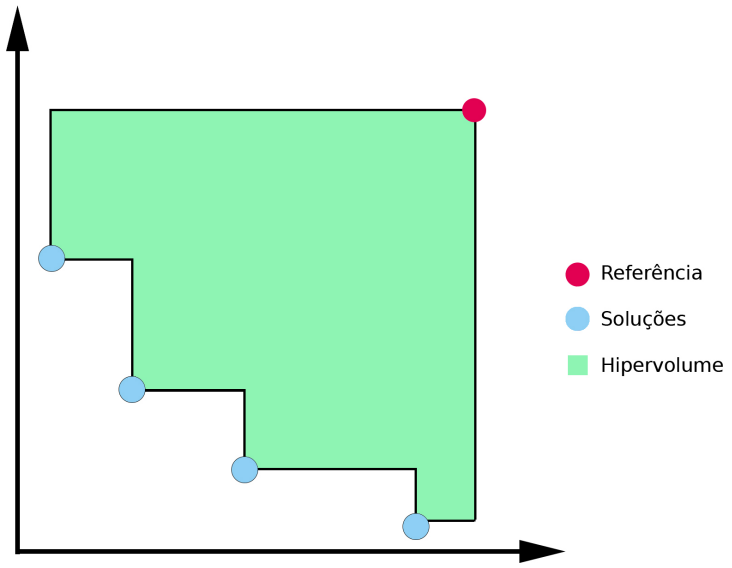


Figura 4.4: Esquema visual do cálculo do hipervolume definido por um conjunto de soluções em relação a um ponto de referência. Este exemplo é bidimensional, mas pode ser expandido equivalentemente para qualquer número maior de dimensões.

antes disso. Dessa forma, a fim de reduzir os tempos computacionais, um critério de parada precoce (*early stopping*) passou a ser adotado.

Para cada geração, a frente de Pareto atual é salva em uma lista. Se não forem encontradas novas frentes de Pareto, em comparação com as já existentes nesta lista, um contador é incrementado em 1, a partir de 0. Se esta variável de controle atingir um valor determinado (no caso deste estudo, 70), o processo evolutivo é então encerrado.

4.2.4 Reprodutibilidade

Um dos principais aspectos a serem aqui destacados é o fato de que as explicações LIME apresentam alguma variação inerente ao método. Ou seja, gerar várias explicações LIME para a mesma imagem pode criar explicações ligeiramente diferentes, em decorrência da geração aleatória de imagens perturbadas dentro do algoritmo LIME. Este aspecto estocástico foi discutido na Subseção 3.1.2.

Mesmo sempre usando o mesmo algoritmo de segmentação, com os mesmos parâmetros, nas mesmas imagens, os pesos explicativos resultantes para cada superpixel ainda podem variar, dependendo de quais superpixels foram cobertos nas imagens perturbadas geradas aleatoriamente. Note que isso pode ser um problema para o GA, uma vez que cada indivíduo na população não terá uma avaliação única. Portanto, o processo evolutivo poderia terminar

com vários indivíduos idênticos com avaliações ligeiramente diferentes.

A fim de se mitigar este tipo de problema, duas abordagens foram avaliadas e adicionadas ao algoritmo EvEx, como componentes de reprodutibilidade. A primeira consistiu em aumentar o número de imagens perturbadas usadas no LIME, visto que isso diminui a variabilidade nas métricas de otimização, embora não consiga eliminá-la completamente. Testes efetuados indicaram que, conforme esperado, essa abordagem ainda retorna populações com indivíduos idênticos e avaliações diferentes, embora essa diferença se torne quase desprezível.

Assim, o número de imagens perturbadas foi ajustado de modo que seja baixo o suficiente para permitir um cálculo rápido, e alto o suficiente para diminuir a variabilidade para um número controlado de casas decimais, para as métricas de objetivo.

A outra abordagem avaliada consistiu em definir sementes aleatórias, de modo que as explicações do LIME sempre usem as mesmas distribuições de imagens perturbadas. Isso torna os resultados consistentes quando o LIME é chamado várias vezes para uma determinada imagem. No entanto, para sementes diferentes, os resultados também podem ser um pouco diferentes.

Dessa forma, na prática, foi usada uma combinação de ambas as abordagens. Várias execuções do código foram realizadas, cada uma com diferentes sementes aleatórias, de modo que os resultados do LIME sejam consistentes, dentro de cada rodada do código, além de comparáveis entre diferentes rodadas.

Essas diferenças foram calculadas como *heatmaps* de desvio padrão, os quais serão descritos nas próximas seções. Em outras palavras, com as abordagens adotadas garante-se que o mesmo indivíduo sempre tenha a mesma avaliação dentro de uma determinada execução e avaliações suficientemente semelhantes entre diferentes execuções.

Esta abordagem combinada gerou explicações que em grande parte alcançam o mesmo consenso entre as frentes de Pareto evoluídas, tornando a geração de explicações totalmente determinísticas para um determinado conjunto de parâmetros e sementes aleatórias, o que é crucial para a reprodutibilidade das explicações.

4.2.5 Comparando *Heatmaps*

Como forma de avaliar as variações entre explicações geradas via EvEx, foi necessário propor uma metodologia de comparação.

O desvio padrão relativo (*Relative Standard Deviation* – RSD) foi utilizado, calculado pixel a pixel. Esta quantidade, também conhecida como "coeficiente de variação", é a razão entre o desvio padrão e a média da população, neste caso pixel a pixel.

Os resultados são plotados como *heatmaps* em tons de cinza, com cores na escala de 0 a 1. Entretanto, note que os valores de RSD podem ser maiores ou iguais a 1, indicando alta variância, mostrando que o desvio padrão é maior do que a média. Dessa forma, o limite superior da escala de cores foi arbitrado em 1, pois considera que qualquer RSD acima deste limite já seja excessivamente alto.

4.3 Novos tipos de perturbações

Conforme discutido na Subseção 3.1.2, as perturbações disponíveis no repositório público LIME incluem apenas cores sólidas e a média interna de cada superpixel. Ambas cobrem cada superpixel uniformemente com uma cor sólida.

Entretanto, ao longo dos estudos aqui efetuados questionou-se se esta cor influencia os *heatmaps* explicativos resultantes, bem como se outros tipos de perturbação poderiam produzir resultados diferentes, ou até mesmo potencialmente melhores. Adicionalmente, em certas aplicações, como análises de imagens de raio X, tomografias ou outras imagens em *grayscale*, é intuitivo cogitar que usar perturbações pretas, brancas ou cinzas possa introduzir artefatos nas explicações geradas. Consequentemente, nestes casos em particular, torna-se ainda mais importante estudar outras alternativas para cobrir os superpixels.

Neste contexto, identificou-se na literatura precedentes [43] para o uso de *blurs* e ruído como perturbações, embora não em técnicas como LIME ou RISE. Ademais, as implementações existentes não incluiam formas de fazer isso de maneira simples.

Portanto, parte deste projeto também envolveu propor e implementar novos tipos de perturbações, tanto para a técnica LIME quanto para a técnica RISE. São eles:

1. *Blur* – Um *blur* gaussiano com desvio padrão ajustável (o valor padrão é 4);

2. Ruído – Ruído com distribuição gaussiana, centrado na cor cinza (127, 127, 127) e com desvio padrão ajustável (o valor padrão é (51,51,51);.

3. Ruído preto e branco – Similar ao anterior, mas convertido para o espaço *grayscale*.

4.4 RISE para Tensorflow

O repositório público da técnica RISE contém apenas definições para um tipo de modelo, usando um código compatível com Tensorflow 1.0. Dessa forma, neste trabalho, também buscou-se expandir este código, para aceitar qualquer tipo de arquitetura escrita em Tensorflow 1.0 ou 2.0.

5
Estudos de Caso

Neste capítulo, serão sumarizados os estudos de caso onde foram aplicadas as técnicas e abordagens discutidas ao longo dos capítulos anteriores. Mais especificamente, os estudos apresentados e discutidos ao longo das subseções seguintes focam em três *datasets* distintos:

1. *Dataset* P-CAM, com imagens de metástases em linfonodos (Seção 5.1);
2. *Dataset* de Malária do National Institute of Health (NIH) (Seção 5.2);
3. *Dataset* de raios X de tórax (*Chest X-Ray* – CXR), relacionados à COVID-19 (Seção 5.4).

De forma geral, estes *datasets* foram escolhidos como os melhores candidatos para este estudo por ter um grande número de imagens, serem de um tamanho que permitia o treino sem exceder a capacidade de computação do *hardware disponível*, e por apresentar alguma característica desejável adicional.

Por exemplo, o *dataset* P-CAM inclui anotações médicas que permitiam uma comparação mais fácil com as explicações. Por sua vez, o *dataset* de malária não representava um caso um pouco mais realista, sem anotações disponíveis, mas ainda bastante controlado por ser um *dataset* balanceado e com padrões facilmente observáveis que diferenciam as duas classes.

O *dataset* CXR foi escolhido por ser o dataset mais extenso, disponível publicamente no momento da escritura desta tese, para COVID-19. Ademais, representa uma situação ainda mais realista de aplicação, onde não há anotações médicas, não há padrões óbvios para um não especialista, além do desbalanceamento das classes.

Todos os experimentos foram rodados em um ambiente virtual remoto, com uma CPU Intel ®Xeon ®(2,30 GHz) e uma GPU NVIDIA P100.

A GPU foi utilizada sempre que possível para treino e inferência das CNNs, com a única exceção sendo os estágios iniciais da experimentação com LIME, onde estava se verificando se parte da instabilidade das explicações advinha de operações inerentemente estocásticas de CUDA, linguagem utilizada pelos *frameworks* de *Deep Learning* para utilizar a GPU.

A estrutura do capítulo está dividida em três seções, uma para cada estudo de caso (Metástases, Malária e Covid CXR). Cada seção é subdividida em metodologia e resultados, com subseções adicionais para descrever elementos específicos de cada caso.

5.1 Metástases

O *dataset* Patch Camelyon (P-CAM) foi desenvolvido por Veeling et al. [3], sendo derivado de *Whole Slide Images* (WSI) marcadas com hematoxilina e eosina, utilizadas no desafio Camelyon16. Este *dataset* foi montado a partir de imagens originais de slides inteiros adquiridas e digitalizadas com uma lente objetiva de 40× (correspondendo a uma resolução de pixel de 0,243 μm) [3] e, posteriormente, sub-amostradas em 10× para aumentar o campo de visão para o P-CAM.

Na sequência, Veeling et al. extraíram *patches* de 96 por 96 pixels das WSIs e converteram estes slides para o formato hue-saturation-value (HSV), seguido da aplicação de *blur* e filtragem de *patches* com linhas de saturação abaixo de 0,07, o que foi feito para excluir *patches* irrelevantes, contendo apenas o fundo claro da imagem. Por sua vez, cada imagem foi então rotulada de acordo com uma classificação binária, correspondendo à presença ou ausência de pelo menos um pixel de tecido tumoral no quadrado central de 32 por 32 pixels de cada *patch* (0 significa ausência, 1 significa presença).

Note que a eventual existência de tecido tumoral fora deste quadrado central não influencia o rótulo binário. No entanto, essa região externa foi mantida nas imagens, tanto para dar contexto relevante para algoritmos de classificação, quanto para possibilitar o uso de certos tipos de modelos do tipo "fully convolutional" que usam *zero-padding* [61].

Por sua vez, um balanceamento de classes próximo a uma divisão 50/50 foi alcançado por um esquema de mineração negativa rígida estocástica. Na sequência, o conjunto de dados resultantes foi disponibilizado em duas versões.

A primeira foi um repositório GitHub [61], contendo o conjunto de dados completo com 327680 *patches*, subdivididos em conjuntos de treinamento, validação e teste, seguindo a mesma divisão dos dados originais do Camelyon16.

A segunda versão é similar à primeira, porém com a remoção de imagens duplicadas que porventura estavam presentes no P-CAM original, devido à amostragem probabilística dos *patches*. Esta versão foi apresentada como um conjunto de dados para uma competição Kaggle [62]. Além da remoção das duplicatas, não há diferenças entre as divisões ou outros aspectos dos dados. Há 220026 imagens rotuladas no total para esta versão.

No restante desta tese, sempre que os conjuntos de dados P-CAM ou seus *patches* forem mencionados, eles se referem a esta segunda versão. A Figura 5.1 mostra exemplos de algumas imagens deste conjunto de dados.

5.1.1 Anotações médicas

Além da classificação binária, o *dataset* Camelyon16 original também contém anotações manuais, feitas por especialistas, sobre quais partes da imagem correspondem às metástases.

Essas anotações/segmentações foram feitas e verificadas por estudantes e patologistas especialistas de dois hospitais diferentes na Holanda (Radboud University Medical Center e University Medical Centre Utrecht) [3], sendo posteriormente mapeadas para os *patches* correspondentes por Veeling et al. e disponibilizados no repositório PCAM GitHub [61].

Ao longo deste texto, as anotações médicas são referidas como "Seg" ou "Segmentação" nas figuras e são visualizadas como uma sobreposição transparente verde, colocada sobre as figuras originais e *heatmaps* explicativos, gerados pelas técnicas de XAI. *Pixels* verdes indicam metástases e pixels brancos indicam tecido normal.

5.1.2 Metodologia

O *dataset* P-CAM foi o principal foco de estudo desta tese de forma geral, sendo que os desenvolvimentos se deram em três principais etapas, descritas em detalhes nas subseções seguintes.

5.1.2.1 Primeira Etapa

Esta etapa foi mais exploratória, sendo constituída pelos primeiros testes realizados no decorrer do projeto. Foram utilizados os algoritmos de segmen-

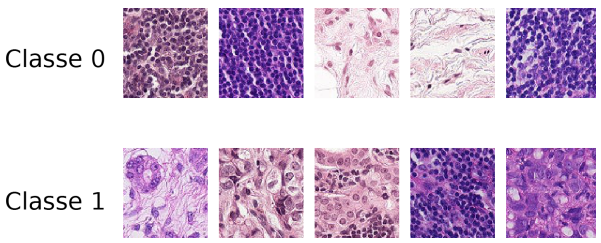


Figura 5.1: Exemplos do dataset P-CAM. Adaptado de [51].

tação FHA, SLIC, Quickshift e Squaregrid para gerar explicações via LIME e comparação dos resultados advindos destes diferentes algoritmos.

Dois modelos de CNNs foram estudados para a tarefa de classificação: VGG19 e Model1, descritos anteriormente nas subseções 2.4.5 e 2.4.7.1.

Os parâmetros para os algoritmos de segmentação foram ajustados manualmente até que gerassem o mesmo número de superpixels para um dado *patch*. Somente *patches* onde os três algoritmos (FHA, Quickshift, SLIC) conseguiram números idênticos de superpixels foram adicionados aos resultados, para que se pudesse comparar como estes algoritmos se comportam em situações similares. O valor padrão de *sigma* para o algoritmo FHA é 0,8, tendo também sido usado para os demais algoritmos que também têm este mesmo parâmetro.

5.1.2.2 Segunda Etapa

Após a análise dos resultados iniciais obtidos durante a primeira etapa, um novo conjunto de *patches* foi analisado, porém utilizando apenas o algoritmo de segmentação FHA associado à metodologia EvEx (ver Seção 4.2).

Neste caso, focou-se em *patches* correspondendo a verdadeiros positivos (TP), onde a segmentação médica era uma sub-região do *patch*, ou seja, não cobria o *patch* inteiro. Dessa forma, pode-se comparar mais facilmente os resultados dos *heatmaps* explicativos gerados com a marcação médica feita para as imagens em questão.

Os limites do espaço de busca para cada um dos três parâmetros que compõem o FHA foram definidos como:

- *scale*: *float* de 3 decimais, intervalo = [1,1000];
- *sigma*: *float* de 2 decimais, intervalo = [0,5];
- *min_size*: *int*, intervalo = [15,500].

Os limites inferiores e superiores para estes intervalos são baseados em resultados preliminares obtidos na primeira etapa. Valores fora destes intervalos não parecem gerar explicações significativas, aumentando o espaço de busca desnecessariamente com superpixels pequenos que não contribuem para as explicações e aumentam o custo computacional.

Por sua vez, os hiperparâmetros do MOGA implementado foram ajustados para:

- Tamanho da população = 80
- Número máximo de gerações = 200

– Probabilidade de mutação = 20%

– Probabilidade de *crossover* (uniforme) = 50%

– Early Stopping após 70 gerações sem melhora.

Estes parâmetros não são adaptativos, permanecendo fixos durante o processo evolucionário. Seus valores foram determinados empiricamente após testes iniciais, até que se observou convergência.

5.1.2.3 Terceira Etapa

Já tendo ganhado maior prática com a técnica LIME e implementado as técnicas EvEx e Squaregrid, a questão do tempo computacional começou a se fazer mais relevante. Em especial, testes com *datasets* contendo imagens maiores mostraram que o tempo de processamento computacional demandado para aplicação do EvEx se tornava impraticavelmente longo, com o *hardware* disponível.

Consequentemente, nesta terceira etapa, a fim de se reduzir o tempo de processamento computacional, pensou-se em uma alternativa de aproximações onde se usa uma segmentação com uma única região de interesse. Ou seja, ao invés de superpixels ou grids, se usa apenas um único segmento quadrado, passado para o LIME antes de gerar um *heatmap*. Embora seja uma aproximação extremamente grosseira, desejava-se saber se era possível obter algum tipo de informação relevante ou aproximada, em um tempo computacional significativamente menor.

Esta técnica, chamada de "Explicações Aproximadas", ao invés de usar algoritmos de segmentação complexos, simplesmente divide a instância a ser explicada em dois “segmentos” ou regiões: um quadrado e toda a área ao redor dele. Obviamente, essa divisão simples não conterá tantas informações contextuais de cor/textura quanto um superpixel segmentado de forma mais precisa. No entanto, usar essa aproximação com apenas um quadrado cria uma simplificação substancial da próxima etapa do processo.

A Figura 5.2 mostra um exemplo dessa segmentação quadrada. Nota-se que, para uma segmentação tão simples, existem apenas 4 opções possíveis em relação às imagens perturbadas. Ou a imagem é deixada intacta, como visto na parte mais à direita da Figura 5.2, ou podem haver três possíveis perturbações onde: (1) o quadrado é coberto, (2) a área fora do quadrado é coberta, ou (3) toda a imagem é coberta.

Note que não é trivial inferir se o uso de uma aproximação tão simples realmente poderia gerar explicações significativas, especialmente porque agora

Figura 5.2: Exemplo de segmentação quadrada e imagens perturbadas. O quadrado amarelo é sobreposto em um *patch* de amostra P-CAM. As 4 imagens representam o original (à direita) e as três possíveis perturbações. Adaptado de [63].

o modelo substituto linear é ajustado com base em apenas 4 imagens, ao invés de uma grande distribuição de perturbações com centenas ou milhares de exemplos. Entretanto, como será visto nos resultados apresentados na próxima seção, esta aproximação potencialmente produz resultados úteis, enquanto também reduz drasticamente os custos computacionais e, consequentemente, o tempo de processamento.

A redução massiva no número de imagens perturbadas significa que a CNN também realiza menos avaliações, o que impacta diretamente na redução do custo computacional. Dessa forma, a técnica de explicações aproximadas, apesar de levar a explicações menos detalhadas, possibilita uma redução substancial no tempo de processamento, chegando a reduzi-lo em algumas ordens de grandeza.

No entanto, a fim de se gerar as explicações mais significativas possíveis com esta aproximação, deve-se identificar o tamanho dos quadrados que resulta nas melhores explicações. Quadrados menores tendem a melhorar a resolução espacial, pois podem discernir de forma mais pontual características menores, presentes em um determinado *patch*, que contribuam mais significativamente para uma determinada classe. Entretanto, normalmente, quadrados muito pequenos fazem com que a área fora do quadrado contenha a maior parte das informações contextuais e, consequentemente, possua pesos de explicação maiores, como visto em cenários semelhantes onde grandes superpixels dominam as explicações (vide Seção 4.2). Por outro lado, espera-se também, por motivos análogos, que quadrados maiores tenham pesos explicativos maiores, uma vez que cobrem uma área maior da imagem, apesar de produzirem resoluções espaciais piores. Dessa forma, é de suma importância definir o tamanho do quadrado que otimize a qualidade das explicações geradas.

Neste intuito, foram feitos testes preliminares, onde todos os tamanhos possíveis de quadrados foram avaliados, para todas as posições na grade. Consequentemente, para os *patches* P-CAM estudados, foram geradas explicações usando esta segmentação quadrada, para quadrados com tamanhos de 1×1

pixels a 95×95 *pixels*, varrendo estes quadrados ao longo da imagem, do canto superior esquerdo até o inferior direito.

Além de computar os pesos de cada quadrado, para todos os seus possíveis tamanhos, a fim de analisar quais melhor explicam as classificações, também foram criados *heatmaps* para visualização das explicações, para cada tamanho de quadrado, seguindo o seguinte procedimento:

- Seleciona-se um determinado tamanho de quadrado;
- Os pesos explicativos de cada quadrado são obtidos via LIME
- Selecionam-se os quadrados desse tamanho com pesos explicativos positivos (contribuindo a favor da classificação);
- Somam-se os pesos explicativos dos quadrados selecionados no passo anterior, para criar um *heatmap*;
- calcula-se a média dos quadrados adicionados, dividindo-se cada pixel do *heatmap* pelo número de quadrados que estão contribuindo para ele.

Dessa forma, o *heatmap* gerado para visualização das explicações é a média ponderada em *pixels* dos quadrados de peso positivo. Este procedimento emula e aproxima o resultado final do EvEx, que produz o *heatmap* médio das melhores explicações encontradas. Consequentemente, espera-se que possivelmente esta aproximação quadrada possa encontrar áreas semelhantes às identificadas pelo EvEx, embora com muito menos detalhes. Por outro lado, a aproximação pode encontrá-los em minutos, em comparação com as cerca de 4 a 8 horas demandadas para uma execução do EvEx ser concluída, para um *patch* P-CAM. Vale notar, contudo, que esse tempo de execução do EvEx ocorreu para um ambiente sem uso de GPU, não otimizado para paralelização com as CNNs classificadoras utilizadas.

Ressalta-se que, para estes experimentos, 95 mapas de calor foram gerados para cada *patch* P-CAM avaliado (um para cada tamanho de quadrado).

5.1.3 Resultados

5.1.3.1 Primeira Etapa

O conjunto de dados P-CAM possui mais de 200 mil imagens, sendo que qualquer uma delas pode ser analisada sob a ótica da geração de explicações por meio dos procedimentos descritos nos capítulos 3 e 4. Os resultados aqui apresentados mostram o comportamento geral observado após gerar e examinar manualmente cerca de quarenta imagens de amostra do conjunto de dados.

Inicialmente, tentou-se segmentar os *patches* em um grande número de superpixels, pensando-se em capturar mais detalhes da imagem. Entretanto, a tentativa de gerar um grande número de superpixels (centenas ou milhares) mostrou rapidamente que cada superpixel corresponderia a uma região muito pequena da imagem, com poucos *pixels*, que não contém informações visuais relevantes para a classificação. Consequentemente, eles compreensivelmente foram associados a pesos de explicação insignificantes, fazendo com que os *heatmaps* gerados fossem quase que inteiramente brancos, indicando que não era necessário aumentar os tempos de computação a fim de se gerar grades quadradas excessivamente finas ou ajustar manualmente os parâmetros de segmentação para produzir milhares de superpixels.

Algumas imagens no conjunto de dados compartilham características semelhantes de cor e textura, tendo sido observado que o comportamento geral das explicações foi consistente para elas. Embora não seja possível garantir que os algoritmos de segmentação utilizados sempre tenham dividido as imagens no mesmo número de superpixels, a cada ajuste manual, ressalta-se que os *patches* analisados sempre foram segmentados entre 20 e 40 superpixels, para faixas semelhantes de parâmetros. Tais intervalos estão na vizinhança de escalas FHA entre 200 e 600, número de segmentos ($n\_segments$) do SLIC entre 20 e 70 e $sigma$ de 0,8 para todos os algoritmos.

A Figura 5.3 mostra as características básicas observadas para exemplos de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, para classificações feitas pelo Modelo1, que tem AUC de 0.9528 para esta tarefa. Em cada linha, a imagem mais à esquerda refere-se a imagem do *patch* original, seguida pela imagem com a segmentação médica sobreposta à original e dos *heatmaps* explicativos obtidos usando as técnica de segmentação SLIC, FHA, Quickshit, AVG e Squaregrid, respectivamente, todos com a segmentação médica sobreposta. Ressalta-se que o caso AVG refere-se a um *heatmap* gerado a partir da média dos *heatmaps* gerados para as segmentações SLIC, FHA, Quickshit.

Conforme mostrado na Figura 5.3, as classificações positivas (onde a classe predita era 1), verdadeiras ou falsas positivas, geralmente apresentaram explicações com grandes pesos positivos e pesos negativos desprezíveis, que tornam-se virtualmente invisíveis nos *heatmaps* ao se utilizar uma escala de cores simétrica. Por outro lado, os verdadeiros / falsos negativos apresentaram explicações com pesos explicativos muito pequenos, sejam eles positivos ou negativos, produzindo mapas quase que inteiramente brancos ao se adotar os mesmos limites da escala de cores usada para os casos verdadeiro e falso positivos.

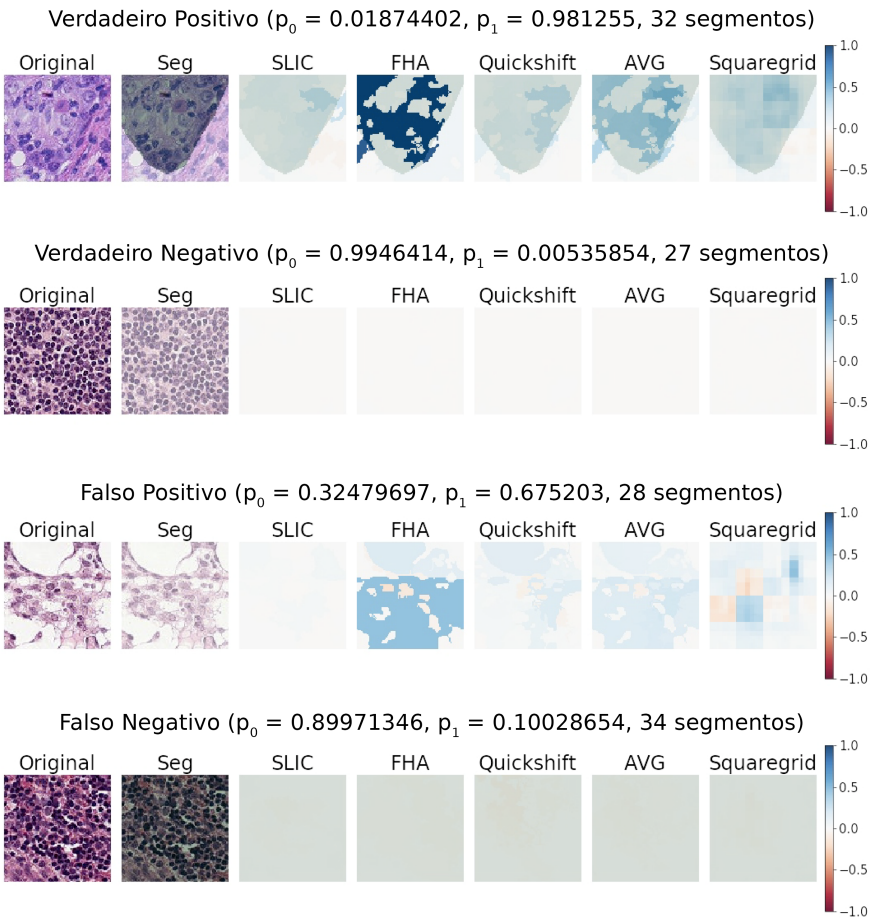


Figura 5.3: Exemplos de explicações LIME para TP, FP, TN, FN do dataset P-CAM. Adaptado de [51].

Esse comportamento foi muito consistente em classificações positivas e negativas, parecendo indicar que a CNN não identificou nenhuma estrutura como um forte indicativo de tecido normal; caso contrário, grandes pesos positivos (áreas em azul forte) seriam perceptíveis em algum lugar nos *heatmaps* negativos verdadeiro / falso. Entretanto, nota-se a ausência de tais estruturas indicativas nos mapas de calor de explicação correlacionada com a previsão da classe 0 (negativa).

Todavia, os resultados sugerem que, de fato, a CNN aprendeu a identificar estruturas associadas à presença de tumores no centro da imagem, conforme pode ser observado pelas áreas azuis presentes nos *heatmaps* explicativos dos casos positivos (classe 1).

De certa forma, o comportamento observado segue a intuição humana. Um patologista normalmente examina uma imagem em busca de indicações de estruturas anormais / tecido tumoral, classificando-a como saudável se nada anormal é encontrado.

Comparando SLIC, FHA e Quickshift Como as previsões positivas verdadeiras apresentavam as explicações com os maiores pesos, indicando que o LIME estava encontrando superpixels relevantes, mais explicações positivas verdadeiras foram geradas.

A Figura 5.4 contém 10 exemplos de explicações geradas para o Modelo1, com os respectivos *heatmaps* para cada algoritmo de segmentação. Similarmente, a Figura 5.5 mostra as explicações geradas para as classificações da rede VGG19. Para ambas as figuras, note que, em cada linha, a imagem mais a esquerda refere-se à imagem do *patch* original, seguida pela imagem com a segmentação médica sobreposta à original e dos *heatmaps* explicativos obtidos usando as técnica de segmentação SLIC, FHA, Quickshit, AVG e Squaregrid, respectivamente, todos com a segmentação médica sobreposta. Ressalta-se que o caso AVG refere-se a um *heatmaps* gerado a partir da média dos *heatmaps* gerados para as segmentações SLIC, FHA, Quickshit.

Todos os *heatmaps* são plotados com a mesma escala de cores (simétrico, vermelho-azul, com limites [-1,1]). A justificativa para esta escolha é de que uma escala simétrica de cor permite uma interpretação mais fácil da contribuição positiva ou negativa dos superpíxels, em que estas ficam claramente marcadas como positivas (azul) ou negativas (vermelho). Embora em muitos casos as contribuições negativas sejam pequenas, é preferível usar esta escala de cor simétrica para evitar ambiguidades na interpretação do *heatmap*.

Vale também ressaltar que os superpixels usados para gerar as explicações tanto para o modelo VGG19 quanto para o Model1 foram os mesmos.

Embora as imagens variassem consideravelmente em texturas e cores, alguns aspectos importantes tornaram-se aparentes. Quando os algoritmos SLIC, Quickshift e FHA foram forçados a produzir o mesmo número de segmentos, três situações principais foram evidenciadas para verdadeiros positivos:

1. **Casos em que a anotação médica (correspondente ao tecido metastático) cobriu quase ou totalmente o *patch* e as texturas e cores eram muito homogêneas em todo o *patch***:

   Nesses casos, nenhum superpixel individual pode explicar a classificação, pois todas as regiões do *patch* eram semelhantes, resultando em pesos baixos, visualizados nos *heatmaps* explicativos como áreas brancas. As únicas exceções para este comportamento ocorreram ao se utilizar o algoritmo FHA, que, às vezes, agrupava todas as regiões semelhantes em textura e cor em um superpixel gigante.

   Como esse superpixel cobria a maior parte do tecido metastático, ele acabava tendo um peso muito alto na explicação do LIME (geralmente

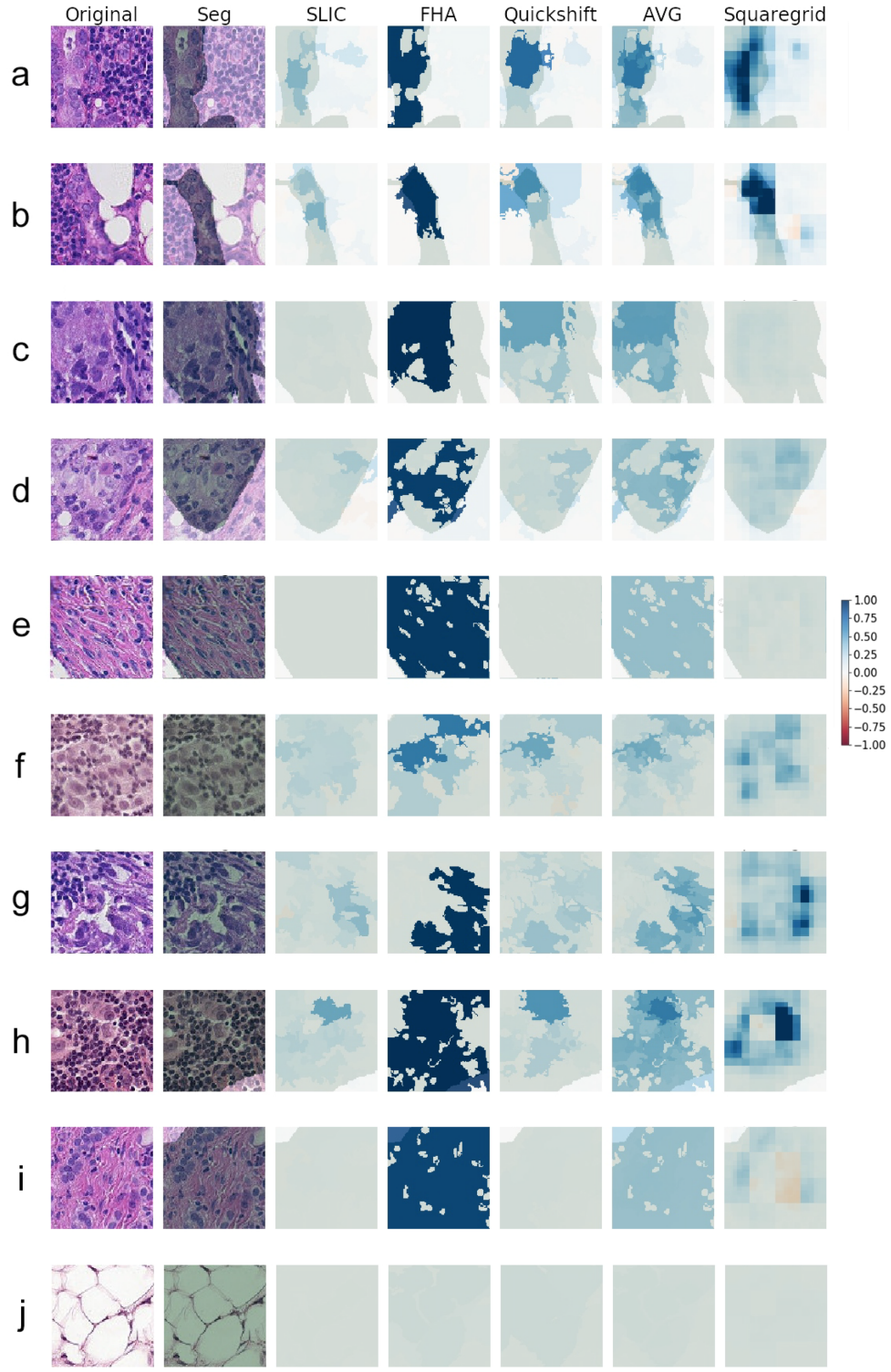


Figura 5.4: Resultados do model1. Adaptado de [51].

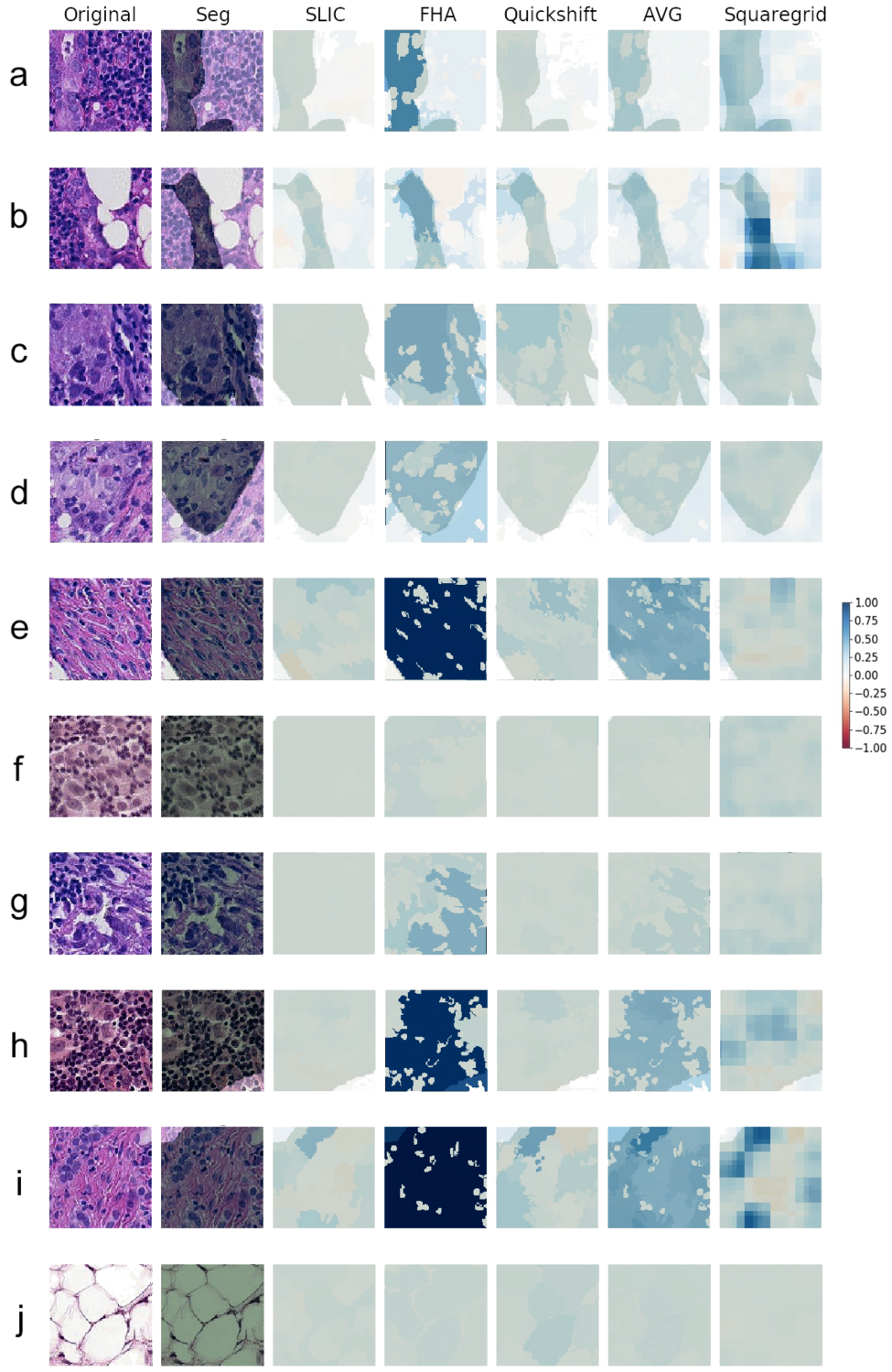


Figura 5.5: Resultados do modelo VGG19. Adaptado de [51].

acima de 0,95).

Exemplos desse comportamento podem ser vistos no "painel j"das Figuras 5.4 e 5.5. O "painel e"também apresenta comportamento semelhante, embora as explicações geradas para o VGG19 tenham dado importância a algumas sub-regiões que não ocorreram para as explicações do Model1.

2. **Casos em que a anotação médica cobriu quase ou totalmente o *patch*, mas haviam várias texturas e cores diferentes**:

   Nesses casos, as explicações geralmente mostravam sub-regiões do *patch* que tinham pesos maiores e, muitas vezes, correspondiam a certas formas, cores ou texturas. A Figura 5.4 (Painéis f, g, h, i) e a Figura 5.5 (Painéis g, h, i) mostram este comportamento particular.

   Dessa forma, pode-se inferir que, embora as informações visuais de todo o *patch* fossem relevantes para os patologistas, as redes neurais fizeram suas classificações influenciadas principalmente por regiões e características específicas menores.

3. **Casos em que a anotação médica era uma sub-região menor do que o *patch***:

   As Figuras 5.4 e 5.5, Painéis a – d, mostram exemplos deste caso. Nestes exemplos, as explicações geradas destacaram áreas de maior peso, sempre contidas dentro da área correspondente à segmentação dos patologistas, mostrando concordância entre as explicações do LIME e as anotações do patologista.

   Conforme também destacado nos casos discutidos anteriormente ((a) a (d)), o comportamento das explicações sugerem que a rede geralmente faz uma classificação / decisão com base em menos informações visuais do que o patologista, já que os *heatmaps* gerados possuem superpixels com pesos de explicação elevados, indicando que essas pequenas áreas são as principais responsáveis pela classificação feita pela CNN, tanto para o Model1 quanto para o VGG19.

   Ademais, também observa-se que, dentre os algoritmos de segmentação avaliados, no geral, os segmentos encontrados pela metodologia SLIC possuem os menores pesos explicativos. Este comportamento é perceptível ao se observar os resultados decorrentes de ambos os modelos da CNN, como visto, por exemplo, nos Painéis b, e, f, g, h, e i da Figura 5.4.

   Entretanto, as segmentações SLIC, FHA e Quickshift parecem concordar aproximadamente sobre quais áreas de cada imagem são importantes, na

maioria dos casos, visto que a área azul mais escura de cada *heatmap* parece sempre ter pelo menos alguma sobreposição.

Ademais, conforme pode ser observado ao se analisar a coluna associada ao caso AVG, presente nas Figuras 5.4 e 5.5, os *heatmaps* gerados pelo cálculo da média dos três algoritmos ajuda a detalhar melhor as áreas relevantes, com maior detalhe.

Por outro lado, o "painel j"da Figura 5.4 mostra uma imagem com uma textura muito diferente, que ocorre raramente no conjunto de dados, com várias manchas brancas grandes. Embora estas manchas brancas não fossem incomuns em pequenas quantidades (como, por exemplo, na imagem do "painel b"), a imagem do "painel j"em particular destaca-se por ser quase que completamente coberta por estas manchas.

Para este caso específico, os superpixels gerados por todos os algoritmos de segmentação foram associados a explicações com pesos muito pequenos, visualmente expressos por *heatmaps* quase que inteiramente brancos, apesar dos modelos das CNNs terem classificado corretamente este *patch*, com uma saída *softmax* acima de 0,98. Entretanto, destaca-se que casos como este são minoria no *dataset* estudado.

Squaregrid Os resultados apresentados nas Figuras 5.4 e 5.5 mostram que os *heatmaps* gerados pela abordagem do squaregrid, apesar de menos detalhados, apresentam boa concordância, nas áreas mais relevantes da imagem, com aqueles gerados com base nos outros métodos de segmentação, especialmente quando faz-se a comparação com os *heatmaps* médios (AVG).

Uma análise minuciosa dos resultados obtidos mostrou que, individualmente, cada quadrado das grades de tamanhos diferentes não tem muita relevância por si só. Isso era esperado, uma vez que, diferentemente do que ocorre para os superpixels, estes quadrados não são gerados com base em texturas ou mudanças de cor presentes nas imagens. Ou seja, eles não possuem nenhuma informação de contexto especial, não havendo razão para esperar que, isoladamente, expliquem bem uma determinada previsão.

No entanto, fazendo-se a sobreposição de grades de tamanhos diferentes, de modo a somar os pesos positivos e negativos, construtivamente ou destrutivamente, observou-se que o perfil do *heatmap* gerado muitas vezes concordava com os obtidos a partir de superpixels, em um grau notável. Em particular, os painéis b, e, f, h e i da Figura 5.4 permitem que se observe este comportamento. Dessa forma, nota-se que esta proposta de abordagem de segmentação

simples, desprovida de parâmetros, foi capaz de gerar resultados interpretáveis por meio da metodologia LIME.

Ademais, comparando os *heatmaps* gerados via Squaregrid e SLIC/FHA/QUickshift, notam-se algumas diferenças. Há uma grande inconsistência nas segmentações das técnicas de superpixels, possivelmente causadas por sua sensibilidade aos parâmetros de segmentação, fazendo com que a forma dos superpixels seja altamente variável, o que influencia significativamente os *heatmaps* resultantes.

A abordagem squaregrid, por outro lado, não apresentou esses problemas. Por exemplo, os painéis h e i das Figuras 5.4 e 5.5 mostram casos onde a explicação quadrada parece destacar núcleos aumentados, com formas / cores / texturas inconsistentes, enquanto os outros três algoritmos não destacavam estas regiões com a mesma intensidade (embora, nesses casos, o modelo VGG19 pareça identificar regiões relevantes mais do que o Model1).

Significado das explicações A comparação entre as explicações geradas, para os modelos da CNN, e as anotações feitas pelos patologistas mostrou que, de modo geral, as explicações costumam estar contidas nas áreas definidas pelas anotações, conforme pode ser visto em todos os exemplos das Figuras 5.4 e 5.5.

Conforme discutido anteriormente, isso sugere que ambos os modelos de CNN estudados parecem estar geralmente se concentrando em áreas que efetivamente concordam com as opiniões de especialistas médicos / patologistas.

Embasamento biológico/médico Um dos aspectos mais importantes desses resultados é que eles derivam de um rótulo binário muito simples. A marcação do conjunto de dados não continha nenhuma informação detalhada da localização do tumor, além da indicação da presença de pelo menos um pixel do tumor em uma região de 32 por 32 *pixels*, localizada no centro do *patch*.

Notavelmente, no entanto, os modelos da CNN aprenderam como identificar estruturas biológicas relevantes. Embora já tenha sido destacado que as explicações, em geral, estão contidas nas anotações médicas, os modelos da CNN parecem muitas vezes focar em algumas características ainda mais específicas destes tecidos.

Por exemplo, uma estrutura comum nas manchas eram pequenas bolhas roxas escuras com formas circulares, geralmente presentes em grande número. Elas são visíveis em vários *patches* já mostrados nas figuras 5.3, 5.4 e 5.5, mas raramente são o foco principal das explicações de LIME. As áreas contendo tais círculos púrpura escuros normalmente aparecem como branco, ou azul claro,

nos *hetamaps* médios (AVG) e quadriculados (gerados com base na técnica Squaregrid), sugerindo que esse tipo de estrutura não parece ser indicativo de tumores. A instância verdadeira positiva na Figura 5.3, bem como os painéis a, b, e, f e g da Figura 5.4 são bons exemplos do comportamento aqui descrito.

Por outro lado, as áreas com azuis escuros dos *hetamaps* explicativos estão geralmente associadas a mudanças na textura e cor, para tons mais claros de rosa / roxo e muitas vezes para manchas roxas claras muito grandes e de formato menos consistente, presentes nas imagens originais dos *patches*. Este comportamento é ainda mais evidente para os *heatmaps* gerados pelo Squaregrid. Os painéis b, f, g, h e i da Figura 5.4 exemplificam este comportamento, embora o mesmo também possa ser observado na maioria dos outros *patches*.

A presença de bolhas roxas alargadas deformadas parece indicar a presença de tecido tumoral para o modelo CNN. Em algumas imagens, esses núcleos aumentados se sobrepõem, e isso parece afetar o peso dado para as explicações, especialmente para a squaregrid (painéis b e h de Figura 5.4).

Além disso, ressalta-se que a escala de Elston-Ellis, usada para graduação histológica e diferenciação de cânceres de mama [64], concentra-se em algumas características semiquantitativas que seguem linhas semelhantes a esta discussão. Por exemplo, uma das características consideradas é o pleomorfismo nuclear, isto é, o grau em que os núcleos são uniformes em tamanho ou variam consideravelmente, caso onde recebem pontuação mais alta. Definitivamente, este também parece ser um dos aspectos mais relevantes para explicações geradas neste trabalho.

A contagem mitótica é outra característica considerada na escala de Elston-Ellis. Mais especificamente, se os núcleos sobrepostos aumentados forem indicativos de atividade mitótica, isso pode justificar porque eles parecem ser considerados relevantes nas explicações do LIME.

5.1.3.2 Segunda Etapa

Frentes de Pareto e Hipervolume A Figura 5.6 mostra um exemplo de frente de Pareto e do comportamento do HV ao longo da evolução de uma dada explicação, gerada com base na segmentação FHA. Mais especificamente, esta figura refere-se ao resultado gerado para a geração do *heatmap* explicativo associado à imagem destacada no "painel a"da Figura 5.7. A população inicial (cruzes laranjas) e as avaliações finais para a semente 45 (círculos azuis) da frente de Pareto são apresentadas no "painel a"; já a evolução do HV é mostrada no "painel b". Note que a forma da frente de Pareto otimizada torna-se aparente

nas proximidades do ponto ótimo (0,0,0).

A maioria dos 80 indivíduos da população inicial, em laranja, está agrupada em dois grupos. O primeiro composto por indivíduos em torno do ponto (1,1,1), que correspondem a segmentações onde o superpixel mais explicativo cobre uma grande área da imagem, mas não consegue explicar satisfatoriamente a classificação, visto que tanto o *explanation score* quanto o peso do superpixel com o maior peso explicativo são muito pequenos (vide definições estabelecidas na Seção 4.2.3). Por outro lado, o segundo grupo de indivíduos está agrupado próximo à zero nos eixos x e y, e ao longo do eixo Z positivo, correspondendo a indivíduos com grandes *explanation scores* e pesos do superpixel com o maior peso explicativo, pelo simples fato de possuírem o superpixel mais explicativo cobrindo uma grande área da imagem, uma vez que, conforme indicado no "painel a"da Figura5.6, as áreas relativas se situam entre 0,3 e 1,0. Conforme discutido anteriormente, isso é causado pela segmentação FHA, e esse comportamento é o que a otimização busca explorar.

Por sua vez, o painel (b) da Figura 5.6 mostra que o hipervolume da frente de Pareto aumenta ao longo das gerações até convergir em torno de aproximadamente 0,88, demonstrando que o algoritmo é capaz de encontrar indivíduos que otimizam os objetivos definidos.

Ao comparar a frente de Pareto da população inicial com a alcançada para a população final, nota-se que indivíduos de baixo desempenho foram progressivamente substituídos nas populações por contrapartes melhores. A frente final inclui indivíduos com superpixels cobrindo regiões muito menores da imagem, com elevados *explanation scores* e grandes pesos do superpixel com

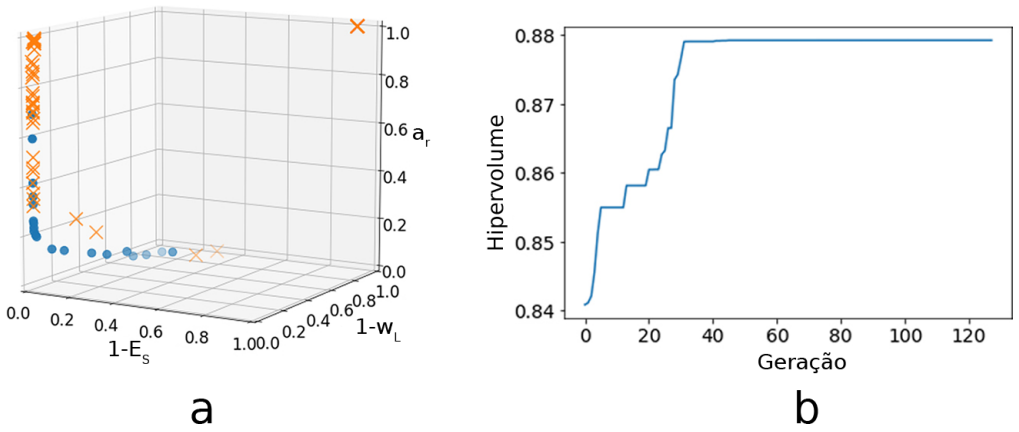


Figura 5.6: Frente de Pareto e gráfico de hiper-volume. O "painel a"mostra a população inicial (em laranja) e a Frente de Pareto final (em azul), para o *patch* correspondente ao "painel a"da Figura 5.7, semente 45). O painel (b) mostra como a métrica de hiper-volume se comporta ao longo do processo de evolução. Adaptado de [56].

o maior peso explicativo.

EvEx No total, considerando todas as imagens estudadas, e todas as sementes, houve 32 execuções da técnica EvEx. O tamanho médio das frentes de Pareto finais observadas foi de 37 indivíduos, sendo que a maior frente teve 61 indivíduos e a menor 15.

A Figura 5.7 mostra os resultados dessas 32 execuções, visto que testou-se 4 sementes para cada um dos 8 *patches* avaliados. Os *heatmaps* de saída consistem na explicação média gerada com base nos indivíduos da frente de Pareto obtida ao final do processo evolutivo, ou seja, na última geração do AG.

Nota-se que, para os casos representados nos painéis a, b, c, d, f, g, h, as áreas destacadas nas imagens em azul escuro, correspondendo aos maiores pesos explicativos, estão contidas dentro da segmentação médica (exibida como uma sobreposição verde transparente). Por sua vez, a imagem do "painel e"tem áreas com pesos explicativos não desprezíveis espalhadas por todo o *patch*, mas observa-se que as regiões em tons azuis mais escuros ainda estão contidas principalmente na segmentação médica, estendendo-se para fora dela até o canto superior esquerdo do *patch*.

Ressalta-se ainda que, devido à abordagem determinística adotada na metodologia deste estudo, os resultados para cada semente são sempre idênticos, mesmo em diferentes execuções. Ademais, mesmo comparando os resultados de diferentes sementes, percebe-se que eles também parecem indicar concordância quanto a quais áreas das imagens são mais relevantes, uma vez que, apesar de cada Frente de Pareto final poder ter um número diferente de indivíduos, com foco em diferentes partes das imagens, a agregação dessas explicações individuais leva a explicações médias consistentes. Essa média também contribui para uma maior robustez, atuando contra a aleatoriedade esperada para explicações LIME, tornando esta técnica interessante para tais aplicações.

Variabilidade e Reprodutibilidade Para testar e quantificar de forma ainda mais detalhada essa concordância entre as sementes, o desvio padrão relativo (RSD) entre as imagens associadas a um mesmo *patch* foi computado, conforme descrito na Seção 4.2.5. A Figura 5.8 mostra esses resultados.

A coluna de desvio padrão (SD) mostra o desvio padrão em *pixels*, calculado entre os *heatmaps* relacionados a cada uma das 4 sementes avaliadas para cada *patch*, conforme mostrado na Figura 5.8. O SD máximo (aproximadamente 0,175) ocorre em uma sub-região do *patch* mostrado no "painel h"da Figura 5.8. No entanto, a região com SDs próximos a este máximo compreende

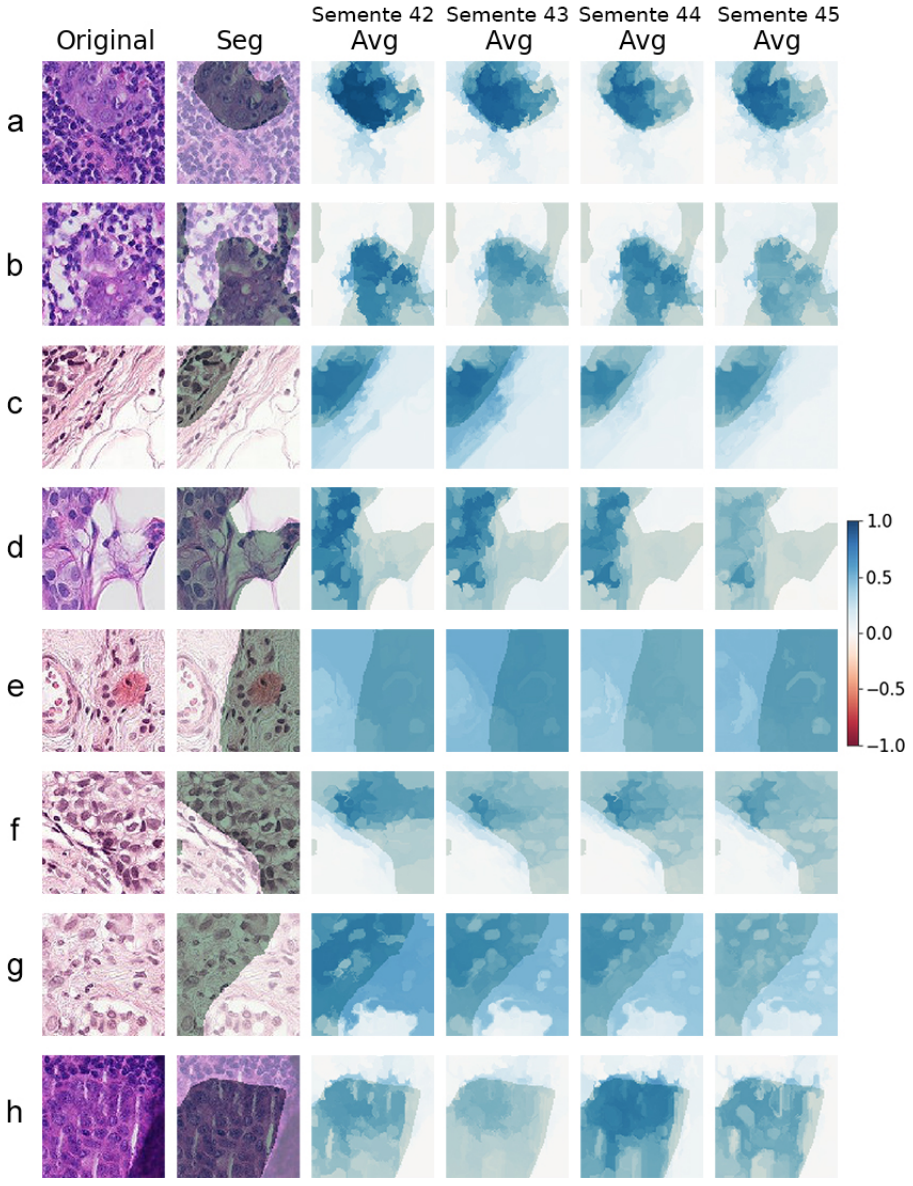


Figura 5.7: Painéis (a) - (h): Explicações evoluídas para as sementes 42 a 45. Avg representa a média das explicações associadas aos indivíduos presentes na Frente de Pareto obtida no final do processo de evolução. Uma escala de cores azul-vermelho é usada para representar pesos explicativos. Seg apresenta a segmentação do médico especialista, para cada imagem, representada por uma sobreposição verde. Adaptado de [56].

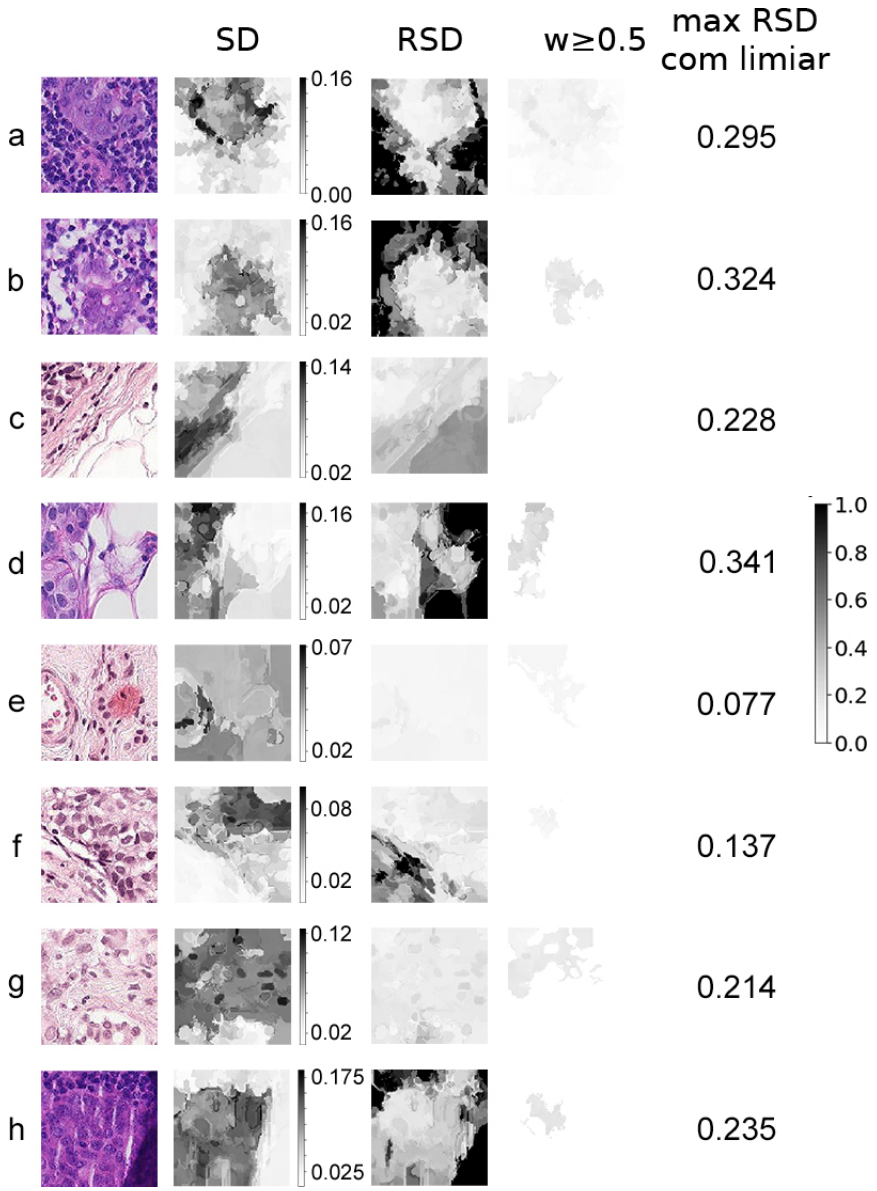


Figura 5.8: Painéis (a) - (h): Desvio padrão em pixels (SD) e Desvio padrão relativo (RSD), para as 4 sementes estudadas. O RSD e o RSD com limiar usam a escala de cores mostrada na extrema direita. A coluna SD usa escalas de cores individuais apresentadas à direita de cada *heatmap* correspondentes. Adaptado de [56].

muito poucos pixels da imagem.

Observa-se que, para a maioria dos casos, os desvios padrão assumem valores baixos. No entanto, é difícil tirar conclusões apenas dessa quantidade, pois diferentes *pixels* têm diferentes pesos explicativos médios nos *heatmaps* azul-vermelho originais. Por exemplo, um desvio padrão de 0,1 é drasticamente mais expressivo para um pixel com peso explicativo médio de 0,12 do que para outro com 0,9.

Consequentemente, a fim de se visualizar melhor este efeito, conforme explicado na Seção 4.2.5, *heatmaps* RSD foram também gerados, usando uma escala de cinza indo de 0 a 1, para todos os casos analisados. Nessa escala, valores iguais ou maiores que 1 indicam que o desvio padrão é igual ou maior que a média, evidenciando uma alta variabilidade. Dessa forma, o limite superior da escala foi feito igual a 1, pois considera-se valores de RSD acima de 1 excessivamente altos. Consequentemente, os pontos com $RSD > 1$ aparecerão em preto nesses *heatmaps*.

Observando-se o painel (a) da Figura 5.8, por exemplo, pode-se notar que existem de fato regiões com RSD de 1 ou mais. No entanto, também se observa que essas regiões de alta variabilidade são áreas associadas a pesos explicativos extremamente baixos e desprezíveis. Este resultado está, de fato, de acordo com observações feitas na Seção 5.1.3.1 de que regiões de baixo peso explicativo tendem a variar mais significativamente entre diferentes execuções de LIME, provavelmente por não serem relevantes para a explicação.

Embora estes pesos flutuem, eles oscilam entre valores absolutos muito pequenos, com muitas casas decimais. Assim, espera-se que essas áreas sejam altamente variáveis em relação ao RSD, porém também altamente irrelevantes para a explicação. Portanto, estas regiões podem ser desconsideradas da explicação com segurança. Esse padrão é, de fato, observado para a maioria das imagens estudadas, embora o painel (e) e também, de forma menos evidente, os painéis (c) e (g) da Figura 5.8 pareçam apresentar alta concordância (baixo RSD), entre os *heatmaps* explicativos respectivamente associados.

Além disso, conforme esperado, as áreas com altos pesos explicativos (azul escuro na Figura 5.7) parecem coincidir com áreas de baixo RSD na Figura 5.8, o que significa que, em geral, as explicações geradas pela aplicação desta metodologia, com diferentes sementes, concordam sobre quais áreas do *patch* contêm as informações mais relevantes para explicar a classificação. Isso concorda com a discussão anterior, porém, a análise RSD ajuda a quantificar e solidificar essa interpretação.

Para melhor visualizar esse comportamento, um terceiro conjunto de*heatmaps* RSD foi gerado, excluindo-se dos *heatmaps* quaisquer *pixels* com

pesos explicativos w médios (média entre as 4 sementes) abaixo de um dado limiar (ver Figura 5.8). Isto é, só foram plotadas regiões de peso explicativo alto, de acordo com o limiar escolhido.

Em particular, o limite usado foi 0,5, o que em geral não é um peso explicativo alto. Percebe-se que, mesmo para este limite baixo, há uma concordância notável entre diferentes sementes, com os RSDs máximos observados sendo no máximo em torno de 0,3, conforme destacado na coluna "RSD máximo com limiar", inserida na Figura5.8. Além disso, nota-se que esses RSDs máximos (tons de cinza mais fortes) são observados apenas em pequenas sub-regiões dos *heatmaps* correspondentes, com a maioria das áreas tendo valores de RSD ainda mais baixos.

Em outras palavras, regiões com pesos explicativos altos apresentam desvio padrão relativo menor. Pesos explicativos mais intensos parecem indicar menor variabilidade.

Notavelmente, observando uma imagem como a mostrada no painel (a) da Figura 5.8, que originalmente tinha regiões com pesos explicativos elevados, acima de 0,8, variou-se este limiar a fim de se avaliar como isso afeta o RSD. Neste caso, verificou-se que o RSD máximo de 0,295, para um limiar de $w = 0,5$, é consideravelmente reduzido para 0,073, usando-se um limiar de $w = 0,8$. Esse comportamento indica que, para um dado *patch*, quanto maior os pesos explicativos associados a uma dada região da imagem, melhor a concordância desta região entre explicações geradas utilizando-se sementes diferentes. Consequentemente, parece razoável esperar que áreas de alto peso explicativo, acima de 0,7 ou mais, concordem consideravelmente entre diferentes sementes. Ademais, é encorajador constatar que mesmo áreas com w moderadamente baixo mostrarem valores RSD muito menores do que 1.

Observando-se os *heatmaps* de "RSD com limiar"da Figura 5.8, nota-se que, conforme esperado, as sub-regiões associadas a valores baixos de RSD são justamente aquelas que apresentam a maior similaridade entre os *heatmaps* gerados a partir de sementes diferentes, para cada imagem original, mostrados na Figura 5.7.

Consequentemente, verifica-se que a metodologia aqui empregada parece ser totalmente determinística para sementes individuais, bem como concordar consideravelmente para sementes diferentes. Dessa forma, conclui-se que esta abordagem do GA MOO possibilita a implementação de uma alternativa automatizada robusta para o ajuste dos parâmetros de segmentação e geração de explicações. O usuário da abordagem EvEx não precisa ajustar manualmente os parâmetros de segmentação, uma vez que a própria técnica já tem suas métricas internas para escolher os mesmos.

Portanto, este parece ser um resultado promissor no intuito de gerar sistemas mais explicáveis e confiáveis em ambientes de IA médica. Note que a reprodutibilidade dos resultados também é bastante aprimorada pela adoção da metodologia aqui proposta, permitindo a obtenção de explicações mais facilmente rastreáveis.

Caso especialistas médicos assim desejassem, poder-se-ia detalhar uma dada explicação para uma semente específica, analisando cada uma das explicações individuais associadas aos indivíduos pertencentes à Frente de Pareto final evoluída para esta semente, entre outras ideias que possam ajudar na visualização de resultados. Por sua vez, a tarefa mais complicada, demorada e não trivial, de encontrar parâmetros apropriados para segmentar os *patches* e gerar explicações pode ser deixada para o GA multi-objetivo aqui apresentado.

Outras considerações A principal desvantagem do EvEx é seu custo computacional. Cada execução do GA com 200 gerações levou entre 6 e 8 horas para gerar um *heatmap* explicativo para um *patch*. Porém, após análise, foi constatado que o fator que mais contribui para este alto tempo computacional é a função LIME. É possível que implementações mais eficientes computacionalmente de LIME possam ser desenvolvidas no futuro, ou que essa mesma metodologia possa ser empregada em um hardware muito mais poderoso do que os *kernels* da nuvem Kaggle, usados neste estudo, permitindo um processamento mais rápido dos *patches*.

De forma semelhante aos resultados obtidos na primeira etapa, uma característica notável das explicações aqui também obtidas é que, partindo de um rótulo binário muito simples no conjunto de dados P-CAM, as explicações geradas foram capazes de produzir *heatmaps* ricos em informações e com alto grau de concordância com as segmentações de especialistas médicos.

No entanto, uma diferença fundamental é que ao usar o ajuste manual de parâmetros, como em [62], pode-se usar a experiência humana ou segmentações de especialistas, para orientar a decisão de quais explicações são melhores, conforme feito na primeira etapa. Por outro lado, no caso desse algoritmo evolutivo, avaliado na segunda etapa, os únicos fatores que norteiam as explicações são as métricas de objetivo a serem otimizadas. As segmentações médicas são utilizadas apenas para comparação, após o término do processo de evolução. No entanto, os *heatmaps* explicativos gerados mostram uma concordância notável com as respectivas segmentações médicas, como mostrado na Figura 5.7.

Dessa forma, verifica-se que a técnica proposta configura-se como uma metodologia valiosa e reprodutível, podendo ser aplicada posteriormente em

outros conjuntos de dados de imagens médicas. Essa abordagem pode ser extremamente valiosa, especialmente para casos onde as segmentações de especialistas médicos não estejam disponíveis, ou na busca de explicações para problemas de imageamento médico, onde ainda não se saiba quais regiões de uma dada imagem devem ser mais relevantes para a classificação.

5.1.3.3 Terceira Etapa

Os testes preliminares, descritos na Seção 5.1.2.3, do algoritmo Squaregrid com tamanhos de quadrados variados, mostraram que, em geral, os quadrados com pesos explicativos altos tendem a se concentrar nas mesmas regiões do *patch*, conforme o tamanho diminui de 95 pixels para 1.

Inicialmente, os quadrados são muito grandes e, consequentemente, ocupam a maior parte do *patch*, fazendo com que tenham altos valores de explicação. Porém, à medida que o tamanho destes diminui, os pesos explicativos associados também tendem a diminuir. Dessa forma, os pesos explicativos associados a quadrados muito pequenos assumem valores desprezivelmente pequenos (perto de zero ou ligeiramente abaixo de zero).

Estas observações iniciais sugerem que não seja necessário gerar *heatmaps* explicativos associados a todos os tamanhos de quadrados. Ao invés disso, para cada *patch* particular, pode-se gerar apenas *heatmaps* associados a quadrados com alguns tamanhos diferentes e se observar as regiões para as quais convergem. Para os *patches* de 96x96 pixels aqui estudados, quadrados com tamanhos de aresta abaixo de 30 pixels, geralmente, já são muito pequenos para gerar explicações satisfatórias, enquanto tamanhos acima de 80 geralmente já são muito grandes. A Figura 5.9 mostra um exemplo desta metodologia aplicada a dois *patches* (a), incluindo também a informação das respectivas segmentações médicas (b) e dos *Heatmaps* explicativos obtidos pela aplicação do EvEx (c). Esta figura também explicita: (d) os pesos explicativos gerados para cada posição no *patch* de quadrados com arestas de 30, 40 e 50 pixels; (e) o quadrado com maior peso explicativo sobreposto ao *patch* original; e (f) o *Heatmap* médio, calculado conforme a métrica estabelecida na seção 5.1.3.3.

Os gráficos da Figura 5.9 permitem que se observe as semelhanças e diferenças entre as aproximações quadradas e as saídas do EvEx, as quais são muito mais detalhadas. Note que os dois *heatmaps* gerados pelo EvEx, mostrados no "painel C"da Figura 5.9, destacam claramente, em tons azuis mais escuros, algumas regiões dentro das respectivas segmentações médicas, onde os pesos explicativos médios são muito maiores que nos arredores. Essas regiões seguem os contornos de forma, textura e cor de certas estruturas celulares do

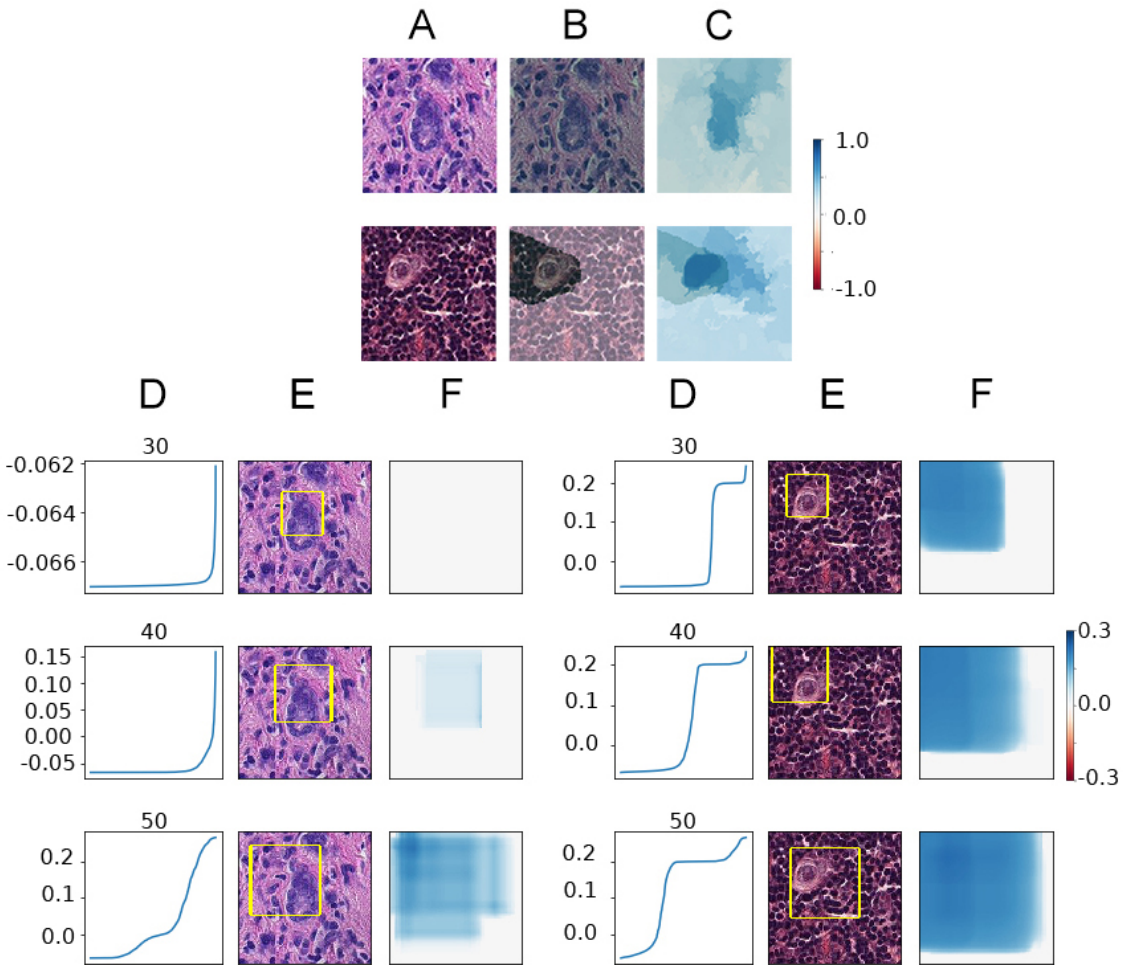


Figura 5.9: Resultados para dois *patches* diferentes. (A) - *Patches* originais; (B) - Patches com Segmentação Médica sobreposta em verde; (C) - *Heatmaps* explicativos obtidos pela aplicação do EvEx; (D) Pesos explicativos gerados para cada posição no *patch* de quadrados com arestas de 30, 40 e 50 pixels. Os pesos explicativos sempre foram ordenados de forma crescente em seus respectivos gráficos; (E) - Quadrado com maior peso explicativo sobreposto ao *patch* original; (F) - *Heatmap* médio, calculado conforme descrito na seção 5.1.3.3. Observe que os *Heatmaps* obtidos com o EvEx têm uma escala de cores de -1 a 1, enquanto os *Heatmaps* quadrados aproximados variam de -0,3 a 0,3. Adaptado de [63].

tecido.

Por sua vez, também pode-se notar que as regiões destacadas em tons azuis mais escuros nos *heatmaps* gerados pelo EvEx coincidem aproximadamente com as regiões destacadas pela aproximação quadrada, conforme visto pelos sub-painéis E e F da Figura 5.9, para ambos os *patches*. Os quadrados com maiores pesos explicativos (painel E) e *heatmaps* (painel F) parecem ser capazes de, pelo menos, aproximar a posição das áreas mais relevantes do *patch*.

Outra diferença importante observada diz respeito aos valores absolutos dos pesos explicativos. Normalmente, os maiores pesos encontrados por explicações do EvEx alcançam valores em torno de 0,6 a 0,9, enquanto os pesos explicativos dos quadrados da técnica de "explicações aproximadas"não produziu valores acima de 0,3, para nenhum dos *patches* estudados. Entretanto, note que, até certo ponto, este comportamento era esperado já que os quadrados contêm muito menos informações contextuais do que os superpixels otimizados pelo EvEx, que são muito mais detalhados. Esta foi também a razão pela qual optou-se por ajustar a escala de cores dos resultados mostrado no "painel F"da Figura 5.9 para uma faixa menor do que a utilizada para o "painel C". Consequentemente, pode-se inferir que os pesos explicativos obtidos pela técnica de "explicações aproximadas"tendem a ser mais baixos do que os convencionalmente alcançados pela aplicação do EvEx, mas ainda conseguem delinear as regiões mais relevantes da imagem.

Outra diferença importante, e talvez a principal vantagem dessa nova abordagem, está na significativa redução dos tempos de computação. A Figura 5.10 mostra o tempo necessário, em segundos, para gerar os *heatmaps* explicativos associados a cada tamanho de aresta do quadrado, variando de 1 a 95 pixels, em passos de 1 pixel. Os resultados aqui apresentados referem-se a análises efetuadas com o *patch* mostrado na Figura 5.9. Note que, para as arestas de 30, 40 e 50 pixels usadas nos exemplos mostrados na Figura 5.9, os tempos foram respectivamente de cerca de 5 minutos, 3 minutos e 42 segundos e 2 minutos e 31 segundos. Por sua vez, o maior tempo observado, para quadrados de 1×1 pixel, foi de cerca de 10 minutos. Por outro lado, para fins de comparação, ressalta-se que os *heatmaps* explicativos superior e inferior, mostrados no painel (c) da Figura 5.9, levaram cerca de 4 horas e 7 horas, respectivamente, para serem gerados pelo EvEx. Consequentemente, conclui-se que a técnica de "explicações aproximadas"possibilita uma diminuição considerável do tempo de processamento computacional, em muitas ordens de grandeza.

A Figura 5.11 mostra uma comparação, lado a lado, de dois *heatmaps* explicativos. O primeiro, à esquerda, corresponde a um indivíduo com alto

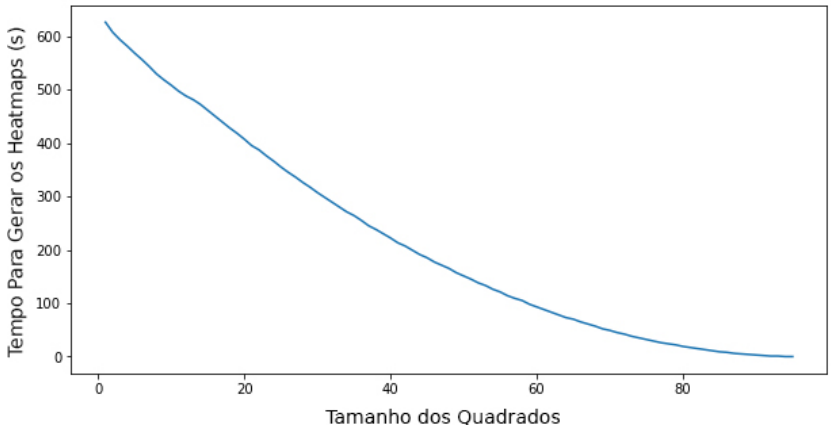
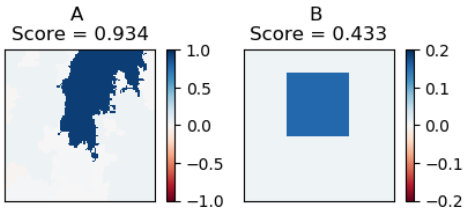


Figura 5.10: Tempo, em segundos, usado para gerar *heatmaps* explicativos para cada tamanho de aresta dos quadrados, variando de 1×1 a 95×95 pixels. Adaptado de [63]

explanation score, presente na Fronteira de Pareto resultante da otimização pelo EvEx, usado para gerar o *heatmap* médio mostrado na linha superior do "painel C"da Figura 5.9. Por sua vez, O segundo *heatmap* explicativo, à direita, corresponde ao quadrado, com tamanho de aresta de 40 pixels, gerado pela técnica de "explicações aproximadas"que obteve o maior peso explicativo, para a mesma imagem. Além da diferença de pesos explicativos já discutida, observando-se os resultados mostrados na Figura 5.11, nota-se que o *explanation score* também é significativamente menor (cerca de 2,16 vezes) para as explicações geradas pela técnica de "explicações aproximadas", em relação aos resultados obtidos via EvEx. Conforme mencionado anteriormente, o *explanation score* corresponde ao R-quadrado do ajuste do modelo linear substituto.

Porém, vale lembrar que, para o caso aqui estudado, cada indivíduo da



.

Figura 5.11: Comparação lado a lado das explicações dos *heatmaps* gerados por ambos os métodos analisados, para o mesmo patch (Figura 5.9, linha superior da coluna A), com seus respectivos *explanation scores*. O painel A mostra o *heatmap* associado a um dos indivíduos da Frente de Pareto otimizada via EvEx, enquanto o painel B mostra o *heatmap* associado ao quadrado com maior peso explicativo, produzido pela técnica de "explicações aproximadas", para quadrados com arestas de 40 pixels. Adaptado de [63]

frente de Pareto do EvEx é uma explicação gerada com 200 perturbações, enquanto as geradas pela técnica de "explicações aproximadas"utilizam apenas 4, o que certamente afeta a métrica R^2 e contribui para o resultado observado. É importante ressaltar que, apesar do R^2 mais baixos da técnica de "explicações aproximadas", os *heatmaps* de ambas as técnicas parecem destacar regiões semelhantes.

Os resultados obtidos indicam que, embora as explicações geradas via LIME e EvEx sejam mais precisas, as explicações geradas pela técnica de "explicações aproximadas"podem ser usadas para gerar perspectivas mais rápidas sobre quais são as áreas mais relevantes de uma imagem, nas quais a CNN foca para efetuar as classificações.

5.2 Malária

5.2.1 Metodologia

5.2.1.1 Dataset

O *dataset* de imagens de malária do NIH [65] possui 27558 imagens de microscopia, com tamanhos variados, contendo uma hemácia cada. Por sua vez, o fundo de cada uma destas imagens é excluído através de pré-processamento e convertido em *pixels* pretos.

As imagens que compõem este *dataset* foram coletadas de 150 pacientes infectados por *Plasmodium falciparum* e 50 não-infectados, usando uma câmera de celular acoplada a um microscópio no Chittagong Medical College de Bangladesh. A base de dados é balanceada entre duas classes com exatamente o mesmo número de instâncias. A classe das imagens foi manualmente anotada por um especialista em leitura de lâminas de extensão sanguínea da Mahidol-Oxford Tropical Research Unit, em Bangkok, sendo que a classe 0 corresponde a hemácias saudáveis e a classe 1 a hemácias infectadas. A Figura 5.12 mostra alguns exemplos de imagens de ambas as classes desta base de dados.

Como as imagens da base de dados têm dimensões variadas, foi necessário redimensioná-las para um tamanho uniforme, antes de efetuar o treinamento da CNN. Dessa forma, no presente trabalho, as imagens foram redimensionadas para 64 por 64 *pixels*. A escolha destas dimensões permite reduzir o custo computacional associado ao treinamento da rede, enquanto possibilita a obtenção de acurácias comparáveis com às explicitadas na literatura [65].

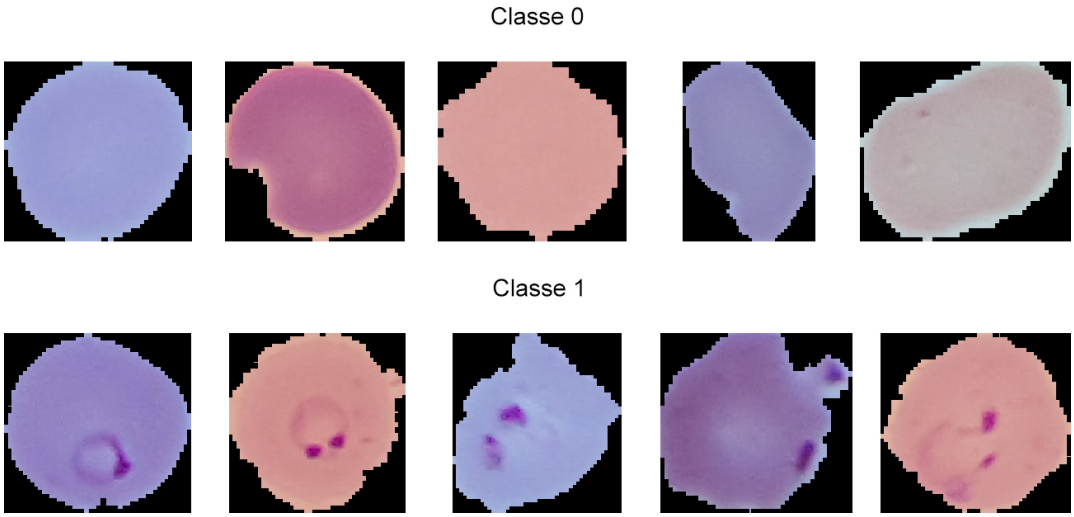


Figura 5.12: Exemplos de imagens da base de dados do NIH. A classe 0 corresponde à células não-infectadas e a classe 1 à células infectadas. Adaptado de [66].

Este *dataset* representou um passo interessante como contraponto ao P-CAM. Por um lado, a identificação das regiões da imagem correspondendo à malária parece ser mais simples, mas, por outro lado, não há uma segmentação médica oficial que sirva para comparação, como havia no caso das metástases presentes no P-CAM. Geralmente, em aplicações reais, bases de dados com segmentações médicas no estilo das presentes no P-CAM são escassas. Dessa forma, estudos com esta nova base de dados pareceram uma forma interessante de testar o desempenho das abordagens vistas até aqui, em casos mais convencionais.

5.2.2 Modelo

O classificador escolhido para este estudo é uma rede neural residual, mais especificamente o modelo ResNet50, introduzido na Seção 2.4.6. Ressalta-se que a única adaptação necessária para o emprego da arquitetura base da ResNet50 no problema aqui tratado consistiu na substituição do bloco de classificação por uma camada densamente conectada com 2 neurônios (correspondendo às duas classes), com ativação *softmax*.

5.2.3 Treinamento

A base de dados descrita na subseção anterior foi dividida em três subconjuntos: treino/validação/teste. Essa divisão foi feita em porcenta-

gens de 80/10/10 do total de imagens, respectivamente, correspondendo a 22046/2756/2756 imagens em cada um desses subconjuntos.

Após testes preliminares, foi notado que a acurácia de validação da rede converge rapidamente (em geral já na terceira época de treino) para valores ao redor de 95%. Ainda assim, o treinamento foi estendido até 20 épocas, usando um *checkpoint* baseado na acurácia do conjunto de validação. Na sequência, o melhor modelo obtido ao final das 20 épocas de treinamento foi usado para avaliar as imagens do conjunto de teste, que não foram apresentadas para a rede durante o treino/validação.

Destaca-se que o tamanho do *batch* utilizado foi 128, a função de custo aplicada foi a "*categorical crossentropy*" e o otimizador utilizado foi o SGD, pelos motivos discutidos na Seção 2.2.5.4.

5.2.4 Resultados

O procedimento de treino, descrito na seção anterior, foi adotado, sendo observada a melhor acurácia de validação na época de treino 16. Portanto, os pesos dessa época foram utilizados para implementação da ResNet50, que obteve acurácia de aproximadamente 96% para as imagens do conjunto de teste. Este valor é comparável àqueles apresentados na literatura [65, 67], parecendo ser próximo ao máximo esperado para o desempenho desta arquitetura na classificação de imagens contidas na base de dados analisada. Por sua vez, a Figura 5.13 mostra a matriz de confusão resultante da aplicação do modelo implementado, para os dados do conjunto de teste.

Em seguida, foram geradas explicações utilizando as técnicas LIME e

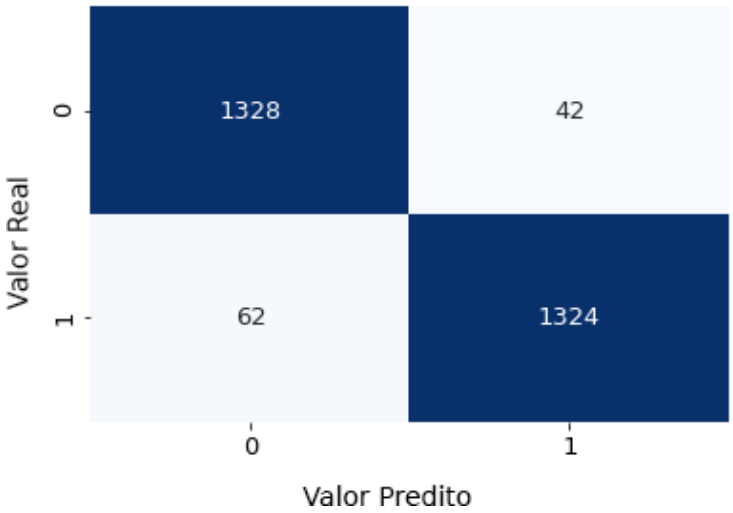


Figura 5.13: Matriz de confusão resultante da aplicação da ResNet50 treinada às imagens do conjunto de teste. Adaptado de [66].

SquareGrid, para diversas imagens do conjunto de teste, conforme descrito na seção anterior. A Figura 5.14 ilustra alguns exemplos dos resultados obtidos, para 6 imagens classificadas como infectadas, mostrando que as regiões com manchas roxas de fato correspondem a superpixels com valor explicativo mais alto, conforme esperado. Percebe-se claramente que a coloração nessas regiões do *heatmap* apresenta tons mais escuros de azul.

O algoritmo de segmentação FHA foi utilizado para geração dos superpixels presentes nos *heatmaps* das explicações obtidas via LIME, apresentadas na segunda coluna da Figura 5.14. O fato destas explicações realçarem justamente as estruturas mais características das células infectadas indica que a heurística usada para a segmentação FHA também mostra-se capaz de produzir bons resultados para este caso. Adicionalmente, é encorajador notar que, na grande maioria dos casos, as explicações geradas pela da metodologia *SquareGrid*, mostradas na quarta coluna da Figura 5.14, parecem também destacar as mesmas regiões realçadas pelo FHA em tons azuis escuros, ou seja, aquelas com pesos explicativos mais altos. Dessa forma, assim como já observado na aplicação do *SquareGrid* para o estudo de caso das metástases, nota-se que as explicações geradas neste novo estudo de caso com a técnica *SquareGrid* também conseguem aproximar, mesmo que de forma mais grosseira, as explicações mais refinadas obtidas por técnicas de segmentação baseadas em superpixes (como a FHA), conseguindo dar indícios coerentes de quais são as regiões mais relevantes da imagem original.

Por sua vez, a terceira coluna da Figura 5.14 circunda, com contornos brancos, as regiões da imagem original onde os *heatmaps* gerados pela aplicação do FHA+LIME, explicitados na segunda coluna, possuem pesos explicativos (xw) superiores a um dado *threshold* arbitrário, que neste caso foi feito igual a 0,2. Esta terceira coluna apresenta uma maneira alternativa de se visualizar os resultados, podendo ser potencialmente interessante para profissionais médicos usando esta técnica. Ademais, ao visualizar diretamente quais são as áreas da imagem original com maior peso explicativo, percebe-se que se tratam das áreas de coloração roxa mais acentuada, conforme esperado.

Outro aspecto importante sobre definir *thresholds* de pesos explicativos é que, como já notado no estudo de caso das metástases, áreas com pesos explicativos muito baixos apresentam maior variabilidade entre execuções diferentes do LIME, para uma mesma imagem. Dessa forma, são menos confiáveis para interpretação dos resultados. Por outro lado, é encorajador que, conforme mostrado na Figura 5.14, justamente as áreas com pesos mais altos nos *heatmaps* explicativos correspondam a estruturas biológicas claramente demarcadas nas imagens de infecções.

Observando-se especificamente os resultados obtidos pelo *SquareGrid*, nota-se que a imagem no painel (e) da Figura 5.14 parece ser a única para a qual a explicação obtida por esta técnica não conseguiu gerar regiões com pesos explicativos satisfatoriamente altos. Destaca-se ainda que, apesar da explicação via FHA+LIME ter sido exitosa para a imagem do "painel e", pode-se perceber que ela atribuiu pesos explicativos consideravelmente menores para a mancha roxa presente nesta imagem do que os pesos atribuídos nas demais explicações.

Com base no comportamento apresentado pelo conjunto de resultados obtidos, é razoável cogitar que a maior dificuldade para a geração de explicações para a imagem do painel (e) possa estar associada à mancha roxa nela presente ser mais difusa, apresentando menor contraste em relação ao restante da imagem. Ademais, no caso específico do *SquareGrid*, este problema também pode ter sido agravado pelo fato da mancha localizar-se justamente entre alguns dos quadrados que compõem a grade, de modo que nenhum deles foi capaz de capturar totalmente o peso explicativo daquela região. Para casos como este, poderia ser potencialmente benéfica uma varredura de parâmetros mais minuciosa, objetivando gerar segmentações mais aprimoradas.

Por sua vez, em termos de tempo de processamento computacional, observou-se que a geração de uma explicação via LIME, com segmentação FHA, leva cerca de 16 segundos, para uma única combinação específica de parâmetros. Dessa forma, caso centenas ou milhares de combinações de parâmetros fossem testadas, é possível notar que o tempo computacional aumentaria muitas ordens de grandeza. Por outro lado, as explicações obtidas via segmentação *SquareGrid* são geradas em cerca de 80 segundos. Embora este seja um tempo maior do que o demandado por uma única explicação FHA+LIME comum, deve-se destacar que frequentemente é necessário fazer vários testes de parâmetros com a técnica LIME, implicando em tempos computacionais muito maiores do que os demandados pela adoção do Squaregrid. Em particular, para o estudo de caso aqui apresentado, vários testes foram realizados a fim de encontrar uma heurística boa para o parâmetro *scale* do FHA.

Os resultados obtidos sugerem uma possível estratégia para geração de explicações em aplicações clínicas de XAI, onde um sistema caixa-preta de classificação, como a ResNet50 aqui estudada, precise ser auditada. Inicialmente, as explicações poderiam ser geradas com a técnica *SquareGrid*, que é rápida e não requer ajuste de parâmetros por especialista. Em seguida, caso explicações mais detalhadas sejam necessárias, poderia ser utilizada alguma técnica de segmentação mais sofisticada, como o algoritmo FHA com alguma heurística tal como a encontrada neste trabalho, para esta base de dados específica. Considerando os demais resultados da tese também poderia se utilizar

a abordagem EvEx para este propósito.

Por fim, caso ainda mais detalhes sejam necessários, é possível realizar uma investigação do desempenho de outros algoritmos de segmentação, como o Quickshift e o SLIC, com o intuito de encontrar as segmentações que gerem explicações com maior peso explicativo e relevância.

5.3 Covid CXR

5.3.1 *Dataset*

A última parte das investigações conduzidas no âmbito desta tese surgiu com o advento da pandemia de COVID-19. Ao longo de 2020, imagens de radiografia de COVID-19 começaram a ser disponibilizadas para estudos de *machine learning*. A maioria dos estudos atualmente disponíveis na literatura usam combinações dos mesmos *datasets* de Covid, com diferentes operações de *augmentation* ou diferentes esquemas de validação e teste [68].

De forma geral, mesmo os *datasets* mais completos têm apenas algumas centenas de radiografias de Covid. Por sua vez, um dos *datasets* mais completos neste estilo foi disponibilizado para a competição/*hackathon* DLAI3 [69]. Deste ponto do texto em diante, o nome dessa competição será usado sempre que se desejar fazer menção a este *dataset*.

O DLAI3 é composto por um conjunto de imagens provenientes de vários outros *datasets*:

- Radiografias pediátricas de pneumonia (não causada por coronavírus) [2];
- Radiografias de COVID-19 de adultos, obtidas por *crawling* da literatura [68, 70];
- Imagens saudáveis e de pneumonia em adultos [71].

No total, o *dataset* DLAI3 contém:

- 1408 radiografias saudáveis (classe 0);
- 3736 radiografias de doenças torácicas, excluindo Covid-19 (classe 1; e
- 363 radiografias de Covid-19 (classe 2).

Este *dataset* representa um desafio maior que os discutidos anteriormente, pois além de ter uma quantidade muito menor de dados, também é extremamente desbalanceado e possui poucos exemplos da classe de maior interesse, que é justamente a de Covid-19, representando menos de 7% dos dados totais.

5.3.2 Metodologia

Como na época em que este estudo foi iniciado não haviam exemplos de CNNs publicamente disponíveis para esta tarefa de classificação em 3 classes, de raios X de Covid, decidiu-se utilizar uma arquitetura da família ResNet, a ResNet50, por ter apresentado bons resultados para a classificação de malária, discutida anteriormente na Seção 5.2.2, e por já ter sido usada na literatura com sucesso para classificações de imagens de tomografias de Covid-19 [68] e radiografias de tórax para outras doenças [71, 72].

As ResNets foram inicializadas com pesos pré-treinados provenientes do *dataset* Imagenet, adotando-se uma abordagem de *transfer learning*. Como os modelos pré-treinados de ResNets convencionam usar imagens de entrada com tamanho de 224×224 pixels, optou-se por usar esse tamanho para este caso também. Entretanto, observou-se que esse tamanho aumenta o custo computacional das explicações EvEx além do limite do hardware disponível. Porém, exceto pelo EvEx, ainda seria possível utilizar todas as outras abordagens e técnicas de XAI até aqui discutidas.

Contudo, somente as camadas de classificação foram efetivamente treinadas, ficando os blocos convolucionais atuando apenas como extratores de *features*. Este procedimento passou a ser adotado, pois em todos os testes preliminares, onde buscou-se treinar todos os pesos, verificou-se que a rede falhava em convergir nas métricas de validação e apresentava *overfit* consistentemente.

O otimizador escolhido foi Adam e a função de custo foi a entropia categórica cruzada (vide Seção 2.2). Por sua vez, foi utilizada uma combinação de *oversampling* das classes menos representativas, com *data augmentation* de todas as classes, a fim de se balancear as classes presentes neste *dataset*.

Para a *augmentation*, foram utilizadas transformações do tipo:

1. *Flip*
2. *Rotate*
3. *Zoom*

Adicionalmente, a métrica de interesse para a competição DLAI3 era o F-score, de modo que monitorou-se esta métrica no *dataset* de validação, a fim de se escolher o melhor conjunto de pesos a ser usado na configuração final da rede destinada a avaliação dos dados de teste. O *dataset* de teste era controlado pelos organizadores da competição, sendo dividido entre um conjunto público (contendo 70% dos dados de teste) e um privado (contendo os 30% restantes). Ressalta-se ainda que o número máximo de submissões por dia, a fim de se

avaliar o F-Score da rede para o conjunto público de testes, era limitado. O desempenho da rede para os dados do *dataset* de teste privado, que definiria o resultado da competição, só foi fornecido ao final do evento.

A fim de se fomentar a capacidade de generalização do modelo, o procedimento de *test time augmentation* (TTA) (vide Seção 2.3.2.1) também foi empregado. Nos casos em que essa técnica foi utilizada, se fez a predição para 10 versões *augmented* de cada imagem. Ademais, todas as técnicas de XAI discutidas no Capítulo 3 foram aplicadas para o modelo treinado, após o fim da competição.

O comportamento da técnica LIME com a segmentação por superpixels e *Squaregrid* foi avaliado quando submetido às novas perturbações (*blur* e *gaussian noise* com e sem cores), a fim de se comparar estes resultados com os provenientes de uma perturbação preta, usada nos estudos de caso anteriormente discutidos nesta tese.

Adicionalmente, os pulmões foram segmentados de forma a separá-los das demais regiões das imagens, pois se espera que eles contenham informações relevantes para este problema em particular. A segmentação FHA não obteve sucesso nesta tarefa, apesar de ter sido a melhor para imagens coloridas de histopatologia. Por outro lado, a segmentação SLIC, com pequenos ajustes manuais, conseguiu atingir este objetivo, ajustando-se o parâmetro $n_{segments} = 15$ e usando-se os valores padrão dos demais parâmetros.

Dessa forma, para as imagens deste *dataset*, optou-se por utilizar o algoritmo de segmentação SLIC, ao invés do FHA, para geração dos superpixels. Entretanto, por ser uma segmentação mais lenta que a FHA, o tempo de processamento computacional demandado torna-se ainda maior, tornando impraticável uma busca por segmentações melhores via EvEx com a infraestrutura de hardware disponível.

5.3.3 Resultados

O desempenho dos modelos ResNet50, com e sem TTA, treinados com a metodologia descrita acima permitiu que se alcança-se a quarta posição na competição DLAI3, para o *dataset* público de teste, e a terceira posição, para o *dataset* privado de teste. Os respectivos F-scores de teste estão sumarizados na Tabela 5.1.

Tabela 5.1: F-scores dos modelos treinados.

F-score	ResNet50	ResNet50+TTA
Público	0,90219	0,90442
Privado	0,91436	0,91340

A Figura 5.15 contém uma comparação dos *heatmaps* resultantes de diferentes tipos de perturbação, obtidos via LIME, usando as segmentações SLIC e Squaregrid.

As perturbações testadas foram *blur*, *gaussian noise*, com (*noise*) e sem ($noise_bw$) cores, e preta sólida (*black*); conforme indicado à direita de cada linha da Figura 5.15.

O parâmetro *max* informado acima de cada *heatmap* refere-se ao maior valor de peso explicativo contido naquele mapa.

De forma geral, observou-se que, usando a perturbação tipo *blur*, a segmentação *Squaregrid* produziu pesos explicativos mais elevados na região dos pulmões e menos elevados no fundo da imagem e demais áreas. Por outro lado, para as demais perturbações avaliadas, nota-se que os pesos explicativos associados aos *heatmap* obtidos via segmentação squaregrid são notavelmente mais baixos do que os obtidos com a perturbação *blur*. Ao se observar os *heatmaps* obtidos via segmentação SLIC, percebe-se que as explicações com *blur* e *noise* colorido tiveram pesos igualmente altos, porém é possível ver que a perturbação *blur* leva a pesos menores para a região do fundo da imagem, assim como ocorreu na Squaregrid.

Ademais, para este estudo de caso, ao contrário dos anteriores, nota-se que, independentemente do tipo de perturbação aplicada, os pesos explicativos máximos obtidos pela segmentação squaregrid foram consistentemente maiores do que os alcançados pelos *heatmaps* gerados por meio da segmentação SLIC, baseada em superpixels. Apesar de se poder argumentar que as explicações obtidas com a segmentação SLIC ainda poderiam ser mais otimizadas, a fim de se tentar produzir pesos maiores, caso fosse efetuada uma busca mais minuciosa pela combinação ótima dos parâmetros da segmentação SLIC, os resultados obtidos evidenciam a utilidade de se ter algum tipo de abordagem inicial com menos parâmetros e menos custo computacional, para explorar um problema desconhecido; ressaltando a importância do Squaregrid. Em um problema onde não haja conhecimento prévio sobre as nuances dos classificadores ou dos padrões das diferentes classes, um método sem parâmetros, agnóstico, como o Squaregrid representaria uma vantagem como abordagem de linha de frente.

Por sua vez, a Figura 5.16 apresenta exemplos de *heatmaps* explicativos gerados para imagens da base de dados de COVID-19, obtidos por diferentes técnicas: GradCAM, *Integrated Gradients*, *Vanilla Gradients*, *Smooth Gradients*, RISE, Squaregrid e LIME, Utilizando-se a segmentação SLIC associada à perturbação do tipo *blur*.

De forma geral, existem muitas imagens do *dataset* que contêm artefatos que podem "distrair" a rede, advindos do processo de *crawling*. Estas imagens,

retiradas de publicações, também contêm letras ou números referentes a *labels* inseridos nas publicações, que não deveriam impactar a classificação. Entretanto, os estudos aqui conduzidos permitiram verificar que a rede neural parece usar essa informação para fazer a classificação. Os *heatmaps* explicativos gerados pelas técnicas GradCAM, Squaregrid, LIME e RISE parecem coincidir em casos nos quais realçam estes artefatos espúrios, presentes em certas imagens, conforme mostrado na Figura 5.16, painel (c) por exemplo. É possivel notar uma anotação (uma letra "d" na imagem) que fica realçada pelo GradCAM, por exemplo.

A linha inferior de cada painel da Figura 5.16 contorna as regiões da imagem original cujos pesos explicativos, presentes no *heatmap* correspondente, ficaram acima de um dado limiar. Para a maioria das técnicas avaliadas ajustou-se o limiar para 60% dos respectivos valores máximos do *heatmap*, exceto para o LIME, Squaregrid e RISE, cujos limiares foram de 70%, 40% e 80%, respectivamente. Dessa forma, facilita-se a visualização das regiões da imagem que produzem os maiores pesos explicativos no *heatmap*. Ressalta-se que estes limiares podem ser variados arbitrariamente, conforme a preferência do usuário das técnicas.

As técnicas de gradiente, em geral, parecem focar na região do pulmão mais do que no fundo da imagem, mas de forma bastante difusa, não permitindo identificar uma única sub-região que influencie mais a classificação. Consequentemente, torna-se mais difícil de interpretar os *heatmaps* gerados com estas técnicas, até mesmo quando usando os limiares para circundar apenas as regiões mais relevantes. Por sua vez, os resultados mostram uma grande concordância entre as técnicas de *Vanilla* e *Smooth Gradients*, porém não tanto destas com a técnica *Integrated Gradients*.

De forma geral, parece muito mais difícil interpretar os mapas gerados via gradiente puro. Porém, a técnica GradCAM, embora ainda oferecendo menos detalhes do que aquelas baseadas em perturbações, parece ser mais interpretável. O fato desta técnica computar o gradiente só até a última camada convolucional, ao invés de segui-lo até a camada de entrada, parece facilitar a interpretação.

Em alguns casos, mesmo com a presença de artefatos espúrios, as redes parecem focar em uma combinação do artefato com o pulmão, como no caso mostrado no "painel d"da Figura 5.16. Curiosamente, também observa-se que as técnicas de gradiente puro não realçam os artefatos diretamente.

De forma geral, estes resultados mostram como técnicas de diferentes estilos podem resultar em *heatmaps* focando em regiões diferentes. Talvez, ao invés de se encarar este aspecto como um problema ou fraqueza das técnicas

de XAI, deva-se encará-los como evidenciando a necessidade de cada vez mais se utilizarem combinações destas diferentes abordagens em estudos futuros, visando criar panoramas mais completos da explicação de uma classificação.

O *dataset* de Covid estudado também demonstra, importantemente, que técnicas de XAI podem ser muito valiosas para detectar vieses causados por características da base de dados. Embora fosse intuitivo cogitar que a presença de artefatos espúrios pudesse afetar as classificações, somente com o uso dessas técnicas obtêm-se de fato indícios mais concretos da real extensão da influência desses artefatos.

Embora a maioria dos estudos com estes *datasets* de COVID-19 estejam ainda em estágio muito inicial, e o objetivo não seja aplicá-los imediatamente para diagnósticos formais, os resultados aqui discutidos suscitam a necessidade de rever a construção desses *datasets*, preferencialmente usando um método mais padronizado de obtenção das imagens e, certamente, evitando a presença de quaisquer artefatos espúrios.

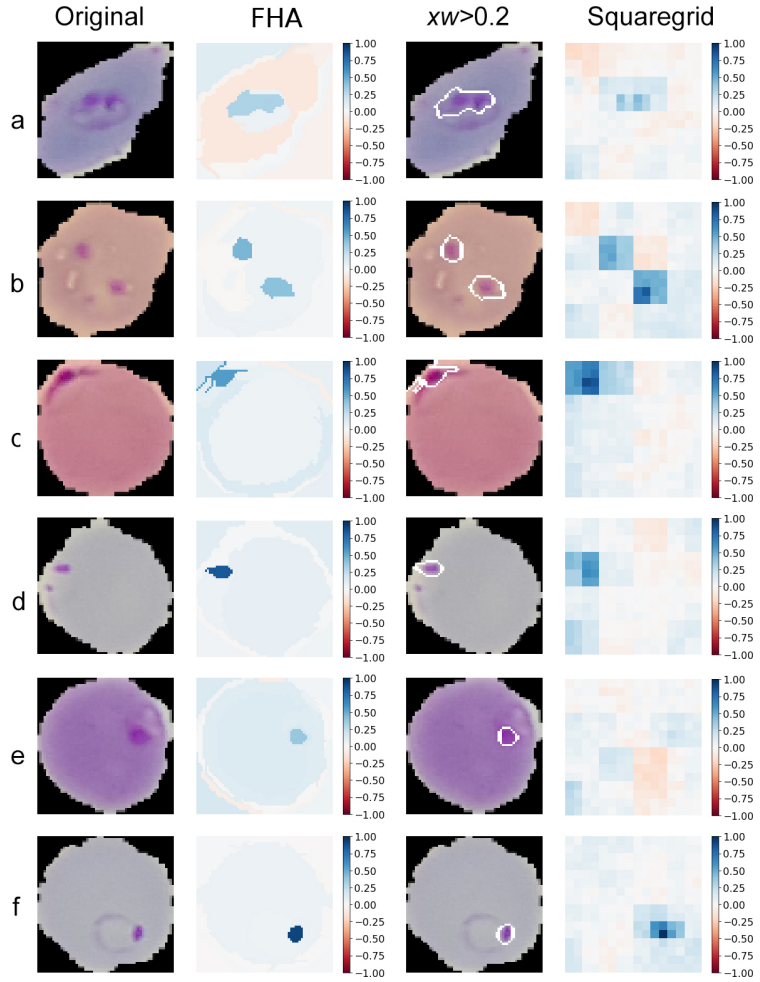


Figura 5.14: Painéis (a) até (f): Explicações para diversas imagens da classe 1 (infectadas) do conjunto de teste. Da esquerda para a direita, as colunas representam, respectivamente: (1) as imagens originais, (2) as explicações obtidas por FHA+LIME, (3) as regiões da imagem original com pesos explicativos acima de 0,2 e (4) a explicação obtida por *SquareGrid*+LIME. A escala de cores dos *heatmaps* se encontra ao lado de cada mapa, indo de -1 a 1 (vermelho à azul). Adaptado de [66].

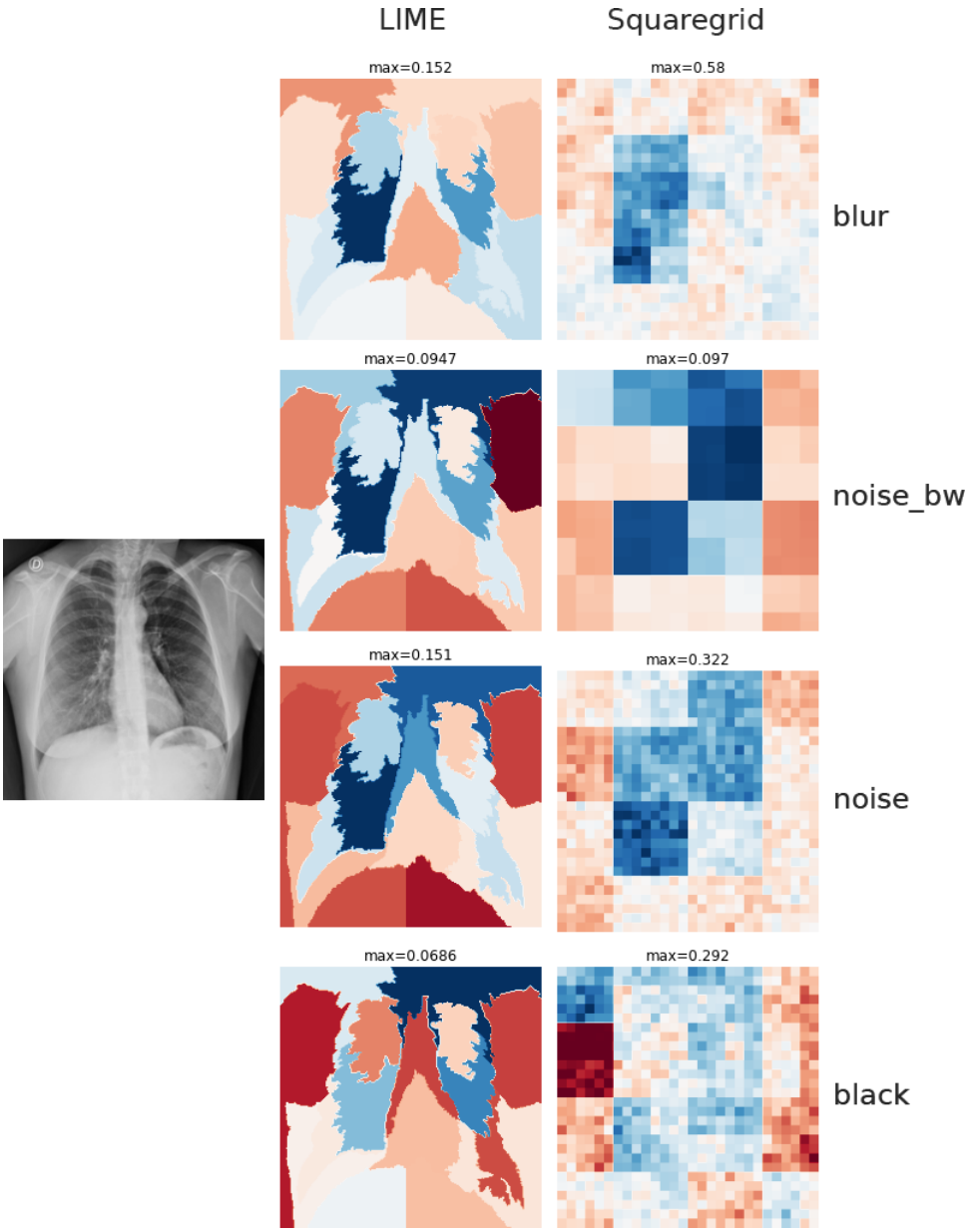


Figura 5.15: *Heatmaps* explicativos gerados para radiografias de COVID-19 via LIME, usando as segmentações SLIC e SquareGrid, para diferentes tipos de perturbação.

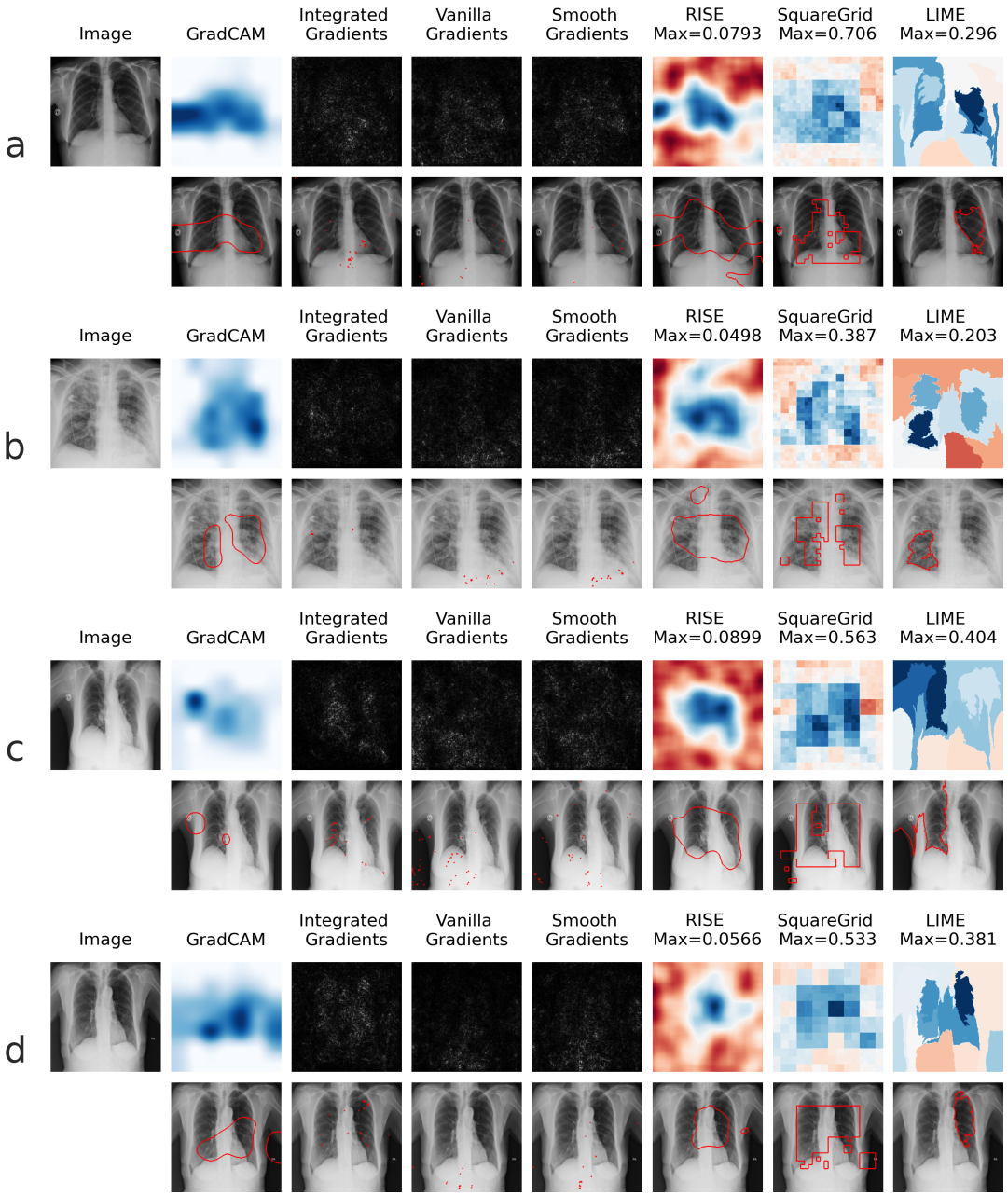


Figura 5.16: *Heatmaps* explicativos gerados para radiografias de COVID-19, por diferentes técnicas de gradiente e perturbação.

6
Conclusão e Trabalhos Futuros

Nesta tese, foram apresentadas uma série de aplicações de técnicas de XAI em problemas de classificação de imagens médicas. Houve um foco maior em técnicas de perturbação inspiradas na metodologia LIME, motivadas especialmente pela facilidade de aplicação desta técnica em diferentes sistemas e *frameworks* de *Deep Learning*.

Duas técnicas principais foram propostas: Squaregrid e EvEx, que contribuiram para resolver o problema de inconsistência e sensibilidade a parâmetros apresentados pela técnica LIME. Notavelmente, a natureza modular da técnica EvEx também permite aplicação futura a diferentes estratégias de XAI. Ambas as técnicas preservam as características desejáveis de ser agnósticas ao modelo, com consistência e usabilidade superior.

Aprimoramentos possíveis para perturbações também foram propostos e investigados. Novas formas de perturbação também foram implementadas e disponibilizadas em repositórios públicos.

Nesse sentido, o objetivo específico de propor e aprimorar o estado da arte das abordagens de XAI foi atingido. Especificamente no caso da técnica EvEx, um conjunto de métricas para qualificar as explicações de forma objetiva também foi proposto com sucesso, permitindo a otimização automática na busca por parâmetros de segmentação.

Quanto ao objetivo específico de aplicar estas técnicas em dados médicos, esta meta também foi largamente atingida. Em relação ao estudo de caso das metástases em linfonodos, os resultados serviram tanto para demonstrar a maior utilidade da segmentação FHA comparada com as demais, bem como o potencial de gerar explicações que coincidem com segmentações médicas. O *dataset* de malária, que não continha anotações além das labels, reforçou a utilidade da técnica Squaregrid como abordagem de linha de frente.

Por sua vez, o estudo de caso de COVID-19 demonstrou que as diferentes técnicas de XAI mostraram a importância da escolha de cores para perturbação na técnica LIME, bem como o fato de que diferentes técnicas de XAI podem gerar explicações que focam em diferentes aspectos da classificação. Dado que estas técnicas fazem "perguntas" diferentes ao classificador, pode ser compreensível que gerem respostas distintas.

A técnica SquareGrid, por sua vez, representa um passo importante para o objetivo de gerar explicações sem a necessidade de ajuste fino de parâmetros, aind que haja um *trade-off* no nível de detalhe do *heatmap* final.

Dessa forma, é possível dizer que, em geral, os objetivos definidos no Capítulo 1 foram atingidos. Contudo, neste processo novos problemas e questões surgiram, que podem e devem ser abordadas por estudos futuros:

1. Técnicas de perturbação têm custo computacional elevado e precisam ser otimizadas. Este aspecto torna-se ainda mais relevante em casos onde deseje-se utilizar o EvEx, que demandou tempos computacionais elevados no *hardware* disponível, para *datasets* com imagens muito grandes.

   Neste intuito, sugere-se fazer alguma combinação das diferentes abordagens de XAI, a fim de se encontrar regiões de interesse via gradiente e então realizar perturbações só nestes trechos da imagem.

2. Sugere-se criar *datasets* artificiais especialmente direcionados para desafiar ou testar aspectos específicos das técnicas de XAI, a fim de se avaliar como elas reagem a certos padrões pré-definidos. Em outras palavras, seria interessante criar um conjunto de dados onde o padrão esperado, a "("resposta certa), seja conhecida em mais detalhes do que nos casos dos *datasets* de imagens médicas aqui estudados.

3. Outro aspecto relevante a ser desenvolvido é o estabelecimento de pesquisas que combinem XAI e com opiniões de especialistas médicos, de forma mais próxima e direta, usando-se métodos de perturbação, tais como LIME. Uma colaboração multidisciplinar beneficiaria muito as análises desta área.

4. Explorar novos *datasets* e modelos. O principal fator limitante nesse caso foi tempo e poder computacional, mas equipes com mais pesquisadores podem aplicar as técnicas aqui propostas a novas frentes de pesquisa, em estudos futuros.

Os resultados obtidos mostram que estas técnicas podem servir não só à especialistas médicos tentando auditar sistemas de classificação, mas também, e talvez especialmente, aos desenvolvedores dos classificadores. Estas abordagens de XAI podem vir a se tornar ferramentas comuns e fundamentais para *debugging* e teste de códigos e algoritmos relacionados a este tipo de problema.

Também ficou evidente que o conjunto de parâmetros necessários para rodar a técnica EvEx é menos sensível e de mais fácil interpretação que os parâ-

metros requeridos pelos algoritmos de segmentação diretamente, configurando um *trade-off* vantajoso.

Muitos dos resultados aqui descritos já foram publicados ao longo dos últimos quatro anos. Alguns destes foram considerados pioneiros, sendo estado da arte para a área de XAI em medicina [73], que apenas agora está começando a ter mais estudos e, consequentemente, publicações. No Apêndice A estão listadas as publicações realizadas ao longo dos trabalhos de pesquisa conduzidos no escopo desta tese [51, 56, 66].

Referências bibliográficas

[1] MIOTTO, R.; WANG, F.; WANG, S.; JIANG, X. ; DUDLEY, J. T.. **Deep learning for healthcare: review, opportunities and challenges**. Briefings in bioinformatics, 19:1236–1246, 2017.

[2] KERMANY, D. S.; GOLDBAUM, M.; CAI, W.; VALENTIM, C. C.; LIANG, H.; BAXTER, S. L.; MCKEOWN, A.; YANG, G.; WU, X.; YAN, F. ; OTHERS. **Identifying medical diagnoses and treatable diseases by image-based deep learning**. Cell, 172(5):1122–1131, 2018.

[3] VEELING, B. S.; LINMANS, J.; WINKENS, J.; COHEN, T. ; WELLING, M.. **Rotation equivariant cnns for digital pathology**. In: INTERNATIONAL CONFERENCE ON MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION, p. 210–218. International Conference on Medical image computing and computer-assisted intervention, 2018.

[4] CHOUGRAD, H.; ZOUAKI, H. ; ALHEYANE, O.. **Deep convolutional neural networks for breast cancer screening**. Computer methods and programs in biomedicine, 157:19–30, 2018.

[5] MENEGOLA, A.; FORNACIALI, M.; PIRES, R.; BITTENCOURT, F. V.; AVILA, S. ; VALLE, E.. **Knowledge transfer for melanoma screening with deep learning**. In: 2017 IEEE 14TH INTERNATIONAL SYMPOSIUM ON BIOMEDICAL IMAGING (ISBI 2017), p. 297–300. IEEE, 2017.

[6] LIANG, Z.; POWELL, A.; ERSOY, I.; POOSTCHI, M.; SILAMUT, K.; PALANIAPPAN, K.; GUO, P.; HOSSAIN, M. A.; SAMEER, A.; MAUDE, R. J. ; OTHERS. **Cnn-based image analysis for malaria diagnosis**. In: 2016 IEEE INTERNATIONAL CONFERENCE ON BIOINFORMATICS AND BIOMEDICINE (BIBM), p. 493–496. IEEE, 2016.

[7] VARSHNI, D.; THAKRAL, K.; AGARWAL, L.; NIJHAWAN, R. ; MITTAL, A.. **Pneumonia detection using cnn based feature extraction**. In: 2019 IEEE INTERNATIONAL CONFERENCE ON ELECTRICAL, COMPUTER AND COMMUNICATION TECHNOLOGIES (ICECCT), p. 1–7. IEEE, 2019.

[8] ADADI, A.; BERRADA, M.. **Peeking inside the black-box: A survey on explainable artificial intelligence (xai)**. IEEE Access, 6:52138–52160, 2018.

[9] GUIDOTTI, R.; MONREALE, A.; RUGGIERI, S.; TURINI, F.; GIANNOTTI, F. ; PEDRESCHI, D.. **A survey of methods for explaining black box models**. ACM computing surveys (CSUR), 51(5):1–42, 2018.

[10] ARRIETA, A. B.; DÍAZ-RODRÍGUEZ, N.; DEL SER, J.; BENNETOT, A.; TABIK, S.; BARBADO, A.; GARCÍA, S.; GIL-LÓPEZ, S.; MOLINA, D.; BENJAMINS, R. ; OTHERS. **Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai**. Information Fusion, 58:82–115, 2020.

[11] **Dimensions ai**. `https://app.dimensions.ai/discover/publication`, 2021.

[12] LEGG, S.; HUTTER, M. ; OTHERS. **A collection of definitions of intelligence**. arXiv preprint arXiv:0706.3639, 2007.

[13] CHOLLET, F.; OTHERS. **Deep learning with Python**, volumen 361. Manning New York, 2018.

[14] GOODFELLOW, I.; BENGIO, Y. ; COURVILLE, A.. **Deep Learning**. MIT Press, 2016. `http://www.deeplearningbook.org`.

[15] HOSSIN, M.; M.N, S.. **A review on evaluation metrics for data classification evaluations**. International Journal of Data Mining and Knowledge Management Process, 5:01–11, 03 2015.

[16] DE BIÈVRE, P.. **The 2012 international vocabulary of metrology:"vim"**. Accreditation and Quality Assurance, 17(2):231–232, 2012.

[17] ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANÉ, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIÉGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y. ; ZHENG, X.. **TensorFlow: Large-scale machine learning on heterogeneous systems**, 2015. Software available from tensorflow.org.

[18] RUDER, S.. **An overview of gradient descent optimization algorithms**. arXiv preprint arXiv:1609.04747, 2016.

[19] QIAN, N.. **On the momentum term in gradient descent learning algorithms**. Neural networks, 12(1):145–151, 1999.

[20] KINGMA, D. P.; BA, J.. **Adam: A method for stochastic optimization**. arXiv preprint arXiv:1412.6980, 2014.

[21] KESKAR, N. S.; SOCHER, R.. **Improving generalization performance by switching from adam to sgd**. arXiv preprint arXiv:1712.07628, 2017.

[22] SIMONYAN, K.; ZISSERMAN, A.. **Very deep convolutional networks for large-scale image recognition**. arXiv preprint arXiv:1409.1556, 2014.

[23] MURTAGH, F.. **Multilayer perceptrons for classification and regression**. Neurocomputing, 2(5-6):183–197, 1991.

[24] HORNIK, K.; STINCHCOMBE, M.; WHITE, H. ; OTHERS. **Multilayer feedforward networks are universal approximators.**

[25] NWANKPA, C.; IJOMAH, W.; GACHAGAN, A. ; MARSHALL, S.. **Activation functions: Comparison of trends in practice and research for deep learning**. arXiv preprint arXiv:1811.03378, 2018.

[26] GLOROT, X.; BORDES, A. ; BENGIO, Y.. **Deep sparse rectifier neural networks**. In: PROCEEDINGS OF THE FOURTEENTH INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND STATISTICS, p. 315–323, 2011.

[27] LECUN, Y.; BOTTOU, L.; BENGIO, Y. ; HAFFNER, P.. **Gradient-based learning applied to document recognition**. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[28] KRIZHEVSKY, A.; SUTSKEVER, I. ; HINTON, G. E.. **Imagenet classification with deep convolutional neural networks**. Communications of the ACM, 60(6):84–90, 2017.

[29] DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K. ; FEI-FEI, L.. **Imagenet: A large-scale hierarchical image database**. In: 2009 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 248–255. Ieee, 2009.

[30] YANN; LECUN, Y. B.. **LeCun, Yann, and Yoshua Bengio. "Convolutional networks for images, speech, and time series**. 1995.

[31] IOFFE, S.; SZEGEDY, C.. **Batch normalization: Accelerating deep network training by reducing internal covariate shift**. arXiv preprint arXiv:1502.03167, 2015.

[32] HE, K.; ZHANG, X.; REN, S. ; SUN, J.. **Deep residual learning for image recognition**. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[33] **Cnn - how to use 160,000 images without crashing**. https://www.kaggle.com/vbookshelf/cnn-how-to-use-160-000-images-without-crashing/data.

[34] KHAN, S.; ISLAM, N.; JAN, Z.; DIN, I. U. ; RODRIGUES, J. J. C.. **A novel deep learning based framework for the detection and classification of breast cancer using transfer learning**. Pattern Recognition Letters, 125:1–6, 2019.

[35] PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DEVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L. ; LERER, A.. **Automatic differentiation in pytorch**. 2017.

[36] QUINLAN, J. R.. **Induction of decision trees**. Machine learning, 1(1):81–106, 1986.

[37] BREIMAN, L.. **Random forests**. Machine learning, 45(1):5–32, 2001.

[38] CHEN, T.; GUESTRIN, C.. **Xgboost: A scalable tree boosting system**. In: PROCEEDINGS OF THE 22ND ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, p. 785–794, 2016.

[39] HEARST, M. A.; DUMAIS, S. T.; OSUNA, E.; PLATT, J. ; SCHOLKOPF, B.. **Support vector machines**. IEEE Intelligent Systems and their applications, 13(4):18–28, 1998.

[40] GILPIN, L. H.; BAU, D.; YUAN, B. Z.; BAJWA, A.; SPECTER, M. ; KAGAL, L.. **Explaining explanations: An overview of interpretability of machine learning**. In: 2018 IEEE 5TH INTERNATIONAL CONFERENCE ON DATA SCIENCE AND ADVANCED ANALYTICS (DSAA), p. 80–89. IEEE, 2018.

[41] RIBEIRO, M. T.; SINGH, S. ; GUESTRIN, C.. **Why should i trust you?: Explaining the predictions of any classifier**. In: PROCEEDINGS

OF THE 22ND ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, p. 1135–1144. Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016.

[42] PETSIUK, V.; DAS, A. ; SAENKO, K.. **Rise: Randomized input sampling for explanation of black-box models**. arXiv preprint arXiv:1806.07421, 2018.

[43] FONG, R. C.; VEDALDI, A.. **Interpretable explanations of black boxes by meaningful perturbation**. In: PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, p. 3429–3437, 2017.

[44] RIBEIRO, M. T.. **Lime**. `https://github.com/marcotcr/lime`, 2016.

[45] FELZENSZWALB, P. F.; HUTTENLOCHER, D. P.. **Efficient graph-based image segmentation**. International journal of computer vision, 59:167–181, 2004.

[46] ACHANTA, R.; SHAJI, A.; SMITH, K.; LUCCHI, A.; FUA, P. ; SÜSSTRUNK, S.. **Slic superpixels compared to state-of-the-art superpixel methods**. IEEE transactions on pattern analysis and machine intelligence, 34:2274–2282, 2012.

[47] VEDALDI, A.; SOATTO, S.. **Quick shift and kernel methods for mode seeking**. In: EUROPEAN CONFERENCE ON COMPUTER VISION, p. 705–718. European Conference on Computer Vision, 2008.

[48] VAN DER WALT, S.; SCHÖNBERGER, J. L.; NUNEZ-IGLESIAS, J.; BOULOGNE, F.; WARNER, J. D.; YAGER, N.; GOUILLART, E.; YU, T. ; THE SCIKIT-IMAGE CONTRIBUTORS. **scikit-image: image processing in Python**. PeerJ, 2:e453, 6 2014.

[49] VEDALDI, A.; FULKERSON, B.. **VLFeat: An open and portable library of computer vision algorithms**. `http://www.vlfeat.org/`, 2008.

[50] HARRIS, C. R.; MILLMAN, K. J.; VAN DER WALT, S. J.; GOMMERS, R.; VIRTANEN, P.; COURNAPEAU, D.; WIESER, E.; TAYLOR, J.; BERG, S.; SMITH, N. J.; KERN, R.; PICUS, M.; HOYER, S.; VAN KERKWIJK, M. H.; BRETT, M.; HALDANE, A.; DEL R'IO, J. F.; WIEBE, M.; PETERSON, P.; G'ERARD-MARCHANT, P.; SHEPPARD, K.; REDDY, T.; WECKESSER,

W.; ABBASI, H.; GOHLKE, C. ; OLIPHANT, T. E.. **Array programming with NumPy**. Nature, 585(7825):357–362, Sept. 2020.

[51] DE SOUSA, I. P.; VELLASCO, M. M. B. R. ; DA SILVA, E. C.. **Local interpretable model-agnostic explanations for classification of lymph node metastases**. Sensors (Basel, Switzerland), 19(13), 2019.

[52] SELVARAJU, R. R.; COGSWELL, M.; DAS, A.; VEDANTAM, R.; PARIKH, D. ; BATRA, D.. **Grad-cam: Visual explanations from deep networks via gradient-based localization**. In: PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, p. 618–626, 2017.

[53] SIMONYAN, K.; VEDALDI, A. ; ZISSERMAN, A.. **Deep inside convolutional networks: Visualising image classification models and saliency maps**. arXiv preprint arXiv:1312.6034, 2013.

[54] SMILKOV, D.; THORAT, N.; KIM, B.; VIÉGAS, F. ; WATTENBERG, M.. **Smoothgrad: removing noise by adding noise**. arXiv preprint arXiv:1706.03825, 2017.

[55] SUNDARARAJAN, M.; TALY, A. ; YAN, Q.. **Axiomatic attribution for deep networks**. arXiv preprint arXiv:1703.01365, 2017.

[56] DE SOUSA, I. P.; VELLASCO, M. M. B. R. ; DA SILVA, E. C.. **Evolved explainable classifications for lymph node metastases**, 2020.

[57] DEB, K.; PRATAP, A.; AGARWAL, S. ; MEYARIVAN, T.. **A fast and elitist multiobjective genetic algorithm: Nsga-ii**. IEEE transactions on evolutionary computation, 6(2):182–197, 2002.

[58] MAS-COLELL, A.; WHINSTON, M. D. ; GREEN, J. R.. **Microeconomic Theory**. Oxford University Press, New York, 1995.

[59] BRADSTREET, L.. **The hypervolume indicator for multi-objective optimisation: calculation and use**. 2011.

[60] FONSECA, C.; PAQUETE, L. ; LÓPEZ-IBÁÑEZ, M.. **An improved dimension-sweep algorithm for the hypervolume indicator**. p. 1157 – 1163, 08 2006.

[61] VEELING, B.. **The patchcamelyon (pcam) deep learning classification benchmark.** https://github.com/basveeling/pcam. Last accessed: 13 march 2019.

[62] **Histopathologic cancer detection**. https://www.kaggle.com/c/histopathologic-cancer-detection/data, Apr. 2020. Last accessed: 13 march 2019.

[63] DE SOUSA, I. P.; VELLASCO, M. M. ; DA SILVA, E. C.. **Approximate explanations for classification of histopathology patches**. In: JOINT EUROPEAN CONFERENCE ON MACHINE LEARNING AND KNOWLEDGE DISCOVERY IN DATABASES, p. 517–526. Springer, 2020.

[64] ELSTON, C. W.; ELLIS, I. O.. **Pathological prognostic factors in breast cancer. i. the value of histological grade in breast cancer: experience from a large study with long-term follow-up**. Histopathology, 19(5):403–410, 1991.

[65] RAJARAMAN, S.; ANTANI, S. K.; POOSTCHI, M.; SILAMUT, K.; HOSSAIN, M. A.; MAUDE, R. J.; JAEGER, S. ; THOMA, G. R.. **Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images**. 2018.

[66] SOUSA, I.; VELLASCO, M. ; SILVA, E.. **Classificações explicáveis para imagens de células infectadas por malária**. In: ANAIS DO XVII ENCONTRO NACIONAL DE INTELIGêNCIA ARTIFICIAL E COMPUTACIONAL, p. 47–57, Porto Alegre, RS, Brasil, 2020. SBC.

[67] SAYYED, A. Q. M. S.; SAHA, D.; HOSSAIN, A. R. ; SHAHNAZ, C.. **Effectiveness of convolutional and capsule network in malaria parasite detection**. IEEE International Conference on Signal Processing, Information, Communication & Systems(SPICSCON), 2019.

[68] WANG, L.; LIN, Z. Q. ; WONG, A.. **Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images**. Scientific Reports, 10(1):1–12, 2020.

[69] CHAN, J.. **Dlai3 hackathon phase3 covid-19 cxr challenge**. `https://www.kaggle.com/c/dlai3-phase3/overview`, 2020.

[70] COHEN, J. P.; MORRISON, P.; DAO, L.; ROTH, K.; DUONG, T. Q. ; GHASSEMI, M.. **Covid-19 image data collection: Prospective predictions are the future**. arXiv 2006.11988, 2020.

[71] WANG, X.; PENG, Y.; LU, L.; LU, Z.; BAGHERI, M. ; SUMMERS, R. M.. **Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of**

common thorax diseases. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 2097–2106, 2017.

[72] XU, S.; WU, H. ; BIE, R.. **Cxnet-m1: Anomaly detection on chest x-rays with image-based deep learning**. IEEE Access, 7:4466–4477, 2018.

[73] POCEVIČIŪTĖ, M.; EILERTSEN, G. ; LUNDSTRÖM, C.. **Survey of xai in digital pathology**. In: ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FOR DIGITAL PATHOLOGY, p. 56–88. Springer, 2020.

A
Artigos Publicados

Conforme mencionado no final do Capítulo 6, este apêndice menciona as publicações realizadas com os dados provenientes de pesquisas conduzidas nesta tese. Foram quatro publicações diretamente relacionadas com estes resultados, sendo que uma está em processo de publicação, não tendo sido ainda disponibilizada online. São elas:

1. Uma publicação na revista Sensors MDPI intitulada "*Local Interpretable Model Agnostic Explanations for Classification of Lymph Node Metastases*"

2. Uma publicação no XVI Encontro Nacional de Inteligência Artificial e Computacional, intitulada "Classificações Explicáveis para Imagens de Células Infectadas por Malária".

3. Uma publicação no arXiV intitulada "Evolved Explanations for Classification of Lymph Node Metastases".

4. Uma publicação pelo ECML PKDD XKDD 2020 intitulada "*Approximate Explanations for Classification of Histopathology Patches*".

Adicionalmente, também foram feitas três outras publicações indiretamente associadas com esta pesquisa, por terem aplicado conceitos vistos aqui de *Deep Learning*, entretanto sem o uso de XAI até o momento, São elas:

1. Uma publicação na revista PeerJ Computer Science intitulada "*Convolutional ensembles for Arabic Handwritten Character and Digit Recognition*"

2. Uma publicação no INNSBDDL 2020 intitulada "*Deep Regression Counting: Customized Datasets and Inter-Architecture Transfer Learning*".

3. Uma publicação na revista Research on Biomedical Engineering intitulada "*Classification of mechanisms underlying cardiac arrhythmias by deep learning*".

O repositório Github da técnica Squaregrid pode ser encontrado no link:

– `https://github.com/palatos/squaregrid`

Contribuições adicionais, tais como a implementação da técnica RISE para Tensorflow, bem como as implementações de perturbações novas para a técnica LIME, encontram-se em repositórios no mesmo perfil.