

UNIVERSIDADE ESTADUAL DO PIAUÍ  
CIÊNCIA DA COMPUTAÇÃO

Giovanni Lucas Araújo Moura

**Segmentação de células mitóticas em raiz de *allium cepa* com  
a arquitetura *U-Net***

TERESINA

2023

# **Segmentação de células mitóticas em raiz de *allium cepas* com a arquitetura *U-Net***

Giovanni Lucas Araújo Moura

Monografia submetida à Universidade Estadual do Piauí como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Trabalho Aprovado, Teresina, 26 de Junho de 2023

---

**Prof. Me. José de Anchieta Araújo Marques**  
Orientador

---

**Marcus Vinicius Ribeiro de Carvalho**

---

**Carlos Giovanni Nunes de Carvalho**

M929s Moura, Giovanni Lucas Araújo.

Segmentação de células mitóticas em raiz de *allium cepas* com  
a arquitetura *U-Net* / Giovanni Lucas Araújo Moura. - 2023.  
61 f. : il.

Monografia (graduação) – Universidade Estadual do Piauí – UESPI,  
Curso de Bacharelado em Ciência da Computação, *Campus Poeta  
Torquato Neto*, Teresina - PI, 2023.

“Orientador: Prof. Me. José de Anchieta Araújo Marques.”

1. Detecção de mitose. 2. Segmentação de imagens. 3. Diagnóstico  
médico – Computadores. I. Título.

CDD: 004

# AGRADECIMENTOS

Agradeço principalmente a minha família, que sempre me deram e continuam a dar suporte em todos os momentos da minha vida.

Ao professor José de Anchieta Araújo Marques, não só pelas orientações mas também por todos os auxílios que possibilitaram a execução desse projeto.

A todos os professores da UESPI que me formaram como pesquisador ao longo de todos esses anos de curso.

A todos os cientistas da computação ao longo da história que desenvolveram todas as tecnologias que hoje são integrais a nossas vidas.

Ao professor pesquisador Pedro Marcos de Almeida e as alunas pesquisadoras Yasmin Lorene de Alvarenga Araújo e Janine Midori Figueiredo Watanabe do curso de Medicina da UESPI que forneceram a base de dados e esclareceram detalhes técnicos da parte médica desse projeto.

A todos os amigos que fiz durante o curso.

E por último a mim mesmo por ter conseguido chegar tão longe, independente das dificuldades que surgiram durante o caminho.

*“Happy accidents are real gifts, and they can open the door to a future that didn’t even exist. It’s kind of nice sometimes to set up something to encourage or allow happy accidents to happen.“*

*(David Lynch)*

# RESUMO

O câncer de mama é uma das principais causas de morte em mulheres ao redor do mundo, com o diagnóstico do problema o mais cedo possível sendo a maneira mais efetiva de aumentar a chance de sobrevivência do paciente. Dentre os métodos utilizados para esse diagnóstico, o mais comum é a análise de tecido mamário para identificação de células mitóticas, que é um processo árduo, demorado e propenso a erro humano. O auxílio de computadores para substituir essa necessidade de trabalho humano é uma área bastante ativa de pesquisas no momento, porém mesmo com os ótimos resultados sendo obtidos, o padrão de diagnóstico ainda continua sendo a análise de um histopatologista experiente. Aprendizes de histopatologia fazem seu treinamento em tecidos vindos de raízes de *allium cepa* para identificação de células mitóticas, devido a melhor disponibilidade e mais fácil distinção de células. Neste projeto, é selecionada, implementada e treinada uma arquitetura de segmentação de imagens para a detecção de células mitóticas em tecidos de *allium cepa* em uma base de dados fornecida por pesquisadores. A arquitetura escolhida para esse problema foi a U-Net devido a sua popularidade em segmentação de imagens médicas, mas mesmo depois da anotação do *dataset*, implementação e treinamento os resultados do modelo foram insatisfatórios, mais provavelmente devido a limitações na base de dados disponibilizada.

**Palavras-chaves:** detecção de mitose, aprendizado profundo, segmentação de imagens

# ABSTRACT

Breast cancer is one of the leading causes of death for women around the world, with early diagnosis being the most effective way of increasing the chance of survival. Among all current methods of diagnosis, the most common is breast tissue analysis in search of mitotic cells, an arduous, slow process that is also prone to human error. The use of computer-aided diagnostics to substitute the need for human effort is a very active field of research, however, even with good results being obtained the golden standard for diagnostics is still analysis by an experienced histopathologist. Trainees for histopathology of mitotic cells train with tissue extracted from the roots of *allium cepa* due to more availability and bigger cells that provide better visibility. This project proposes the selection, implementation, and training of an image segmentation architecture for mitosis detection on *allium cepa* tissue on a database provided by researchers. The selected segmentation architecture ended up being U-Net due to its popularity in medical image segmentation, however after preparing the data, implementing, and training the model the results ended up being unsatisfactory, most likely due to limitations on the provided database.

**Key-words:** mitosis detection, deep learning, image segmentation

# LISTA DE ILUSTRAÇÕES

Figura 1 – Imagem microscópica de tecido celular de uma raiz de <i>allium cepa</i> , tirada pela câmera do celular de uma das estudantes relacionadas ao projeto . . . . .	15
Figura 2 – Representações de um neurônio biológico(à esquerda) e um <i>perceptron</i> (à direita) . . . . .	22
Figura 3 – Representação de uma rede neural simples com uma camada de entrada de tamanho 3, duas camadas escondidas com 5 e 4 perceptrons respectivamente, e uma camada de saída de tamanho 1 para resposta binária . . . . .	24
Figura 4 – Estrutura da CNN <i>LeNet</i> . . . . .	25
Figura 5 – Representação do funcionamento de um filtro. O segmento da matriz de entrada é processado com o resultado sendo a soma ponderada da aplicação do filtro, indicando a ativação do segmento filtrado. . . . .	25
Figura 6 – Aplicação de <i>max pooling</i> em uma imagem 4x4 utilizando um filtro 2x2. A redução do tamanho da imagem consequentemente também reduz o processamento das próximas camadas. . . . .	26
Figura 7 – Exemplo de detecção, com cachorro, pessoa e bicicleta sendo as classes detectadas . . . . .	28
Figura 8 – Exemplos de segmentação de pessoas, com segmentação semântica na esquerda e segmentação de instância na direita . . . . .	28
Figura 9 – Visualização da estrutura da arquitetura U-Net . . . . .	30
Figura 10 – Diagrama do processo de desenvolvimento desse projeto . . . . .	32
Figura 11 – Exemplo de anotação fornecida para cada uma das imagens da base de dados, com a imagem original na esquerda e a anotada na direita . . . . .	33
Figura 12 – Exemplos de imagens da base de dados . . . . .	34
Figura 13 – Visualização do uso da ferramenta <i>labelme</i> para anotação da base de dados fornecida . . . . .	35
Figura 14 – Diagrama de componentes das bibliotecas utilizadas em cada arquivo do projeto	37
Figura 15 – Visualização do fluxo de funcionamento arquivo <i>inferencia.py</i> . . . . .	41
Figura 16 – Visualização das camadas de segmentação retornadas pelo modelo . . . . .	45
Figura 17 – Visualização das camadas individuais para cada classe do modelo, com a imagem original na esquerda . . . . .	47

# LISTA DE TABELAS

Tabela 1 – String de busca resultante . . . . .	17
Tabela 2 – Total de resultados retornados e selecionados . . . . .	17
Tabela 3 – Amostras aleatórias de resultados de IoU por combinação de parâmetros tentados . . . . .	44

# LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
CAD	Computer Aided Diagnostics
RNA	Rede Neural Artificial
CNN	Convolutional Neural Network
SVM	Support Vector Machine
AG	Attention Gate
IRRCNN	Inception Recurrent Residual Convolutional Neural Network
DBN	Deep Belief Network
HKH-ABO	Hybrid Krill Herd African Buffalo Optimization
R-CNN	Regions with CNN Features
IoU	Intersection Over Union

# SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>13</b>
<b>1.1</b>	<b>Contextualização . . . . .</b>	<b>13</b>
<b>1.2</b>	<b>Motivação e Justificativa . . . . .</b>	<b>14</b>
<b>1.3</b>	<b>Objetivo . . . . .</b>	<b>15</b>
<b>1.3.1</b>	Geral . . . . .	15
<b>1.3.2</b>	Específico . . . . .	15
<b>2</b>	<b>REVISÃO BIBLIOGRÁFICA . . . . .</b>	<b>16</b>
<b>2.1</b>	<b>Revisão Sistemática . . . . .</b>	<b>16</b>
<b>2.1.1</b>	Seleção de repositórios científicos . . . . .	16
<b>2.1.2</b>	Refinamento da busca . . . . .	16
<b>2.1.3</b>	Palavras-chave( <i>keywords</i> ) e <i>String</i> de busca . . . . .	17
<b>2.1.4</b>	Critérios de exclusão . . . . .	17
<b>2.1.5</b>	Seleção dos artigos retornados . . . . .	17
<b>2.2</b>	<b>Trabalhos Relacionados . . . . .</b>	<b>18</b>
<b>2.3</b>	<b>Conclusões . . . . .</b>	<b>21</b>
<b>3</b>	<b>FUNDAMENTAÇÃO TEÓRICA . . . . .</b>	<b>22</b>
<b>3.1</b>	<b><i>Perceptrons</i> e Redes Neurais Artificiais . . . . .</b>	<b>22</b>
<b>3.2</b>	<b>Redes Neurais Convolucionais . . . . .</b>	<b>23</b>
<b>3.2.1</b>	Camada de Convolução . . . . .	25
<b>3.2.2</b>	Camada de <i>pooling</i> . . . . .	26
<b>3.2.3</b>	Camadas completamente conectadas . . . . .	27
<b>3.3</b>	<b>Paradigmas de resposta . . . . .</b>	<b>27</b>
<b>3.4</b>	<b>U-Net . . . . .</b>	<b>29</b>
<b>4</b>	<b>PROCEDIMENTO METODOLÓGICO . . . . .</b>	<b>31</b>
<b>5</b>	<b>DESENVOLVIMENTO . . . . .</b>	<b>32</b>
<b>5.1</b>	<b>Obtenção da base de dados . . . . .</b>	<b>32</b>
<b>5.2</b>	<b>Preparação dos dados . . . . .</b>	<b>33</b>
<b>5.2.1</b>	Anotação . . . . .	34
<b>5.2.2</b>	Conversão . . . . .	35
<b>5.2.3</b>	Separação . . . . .	36
<b>5.3</b>	<b>Implementação . . . . .</b>	<b>36</b>
<b>5.3.1</b>	Dataset . . . . .	38

5.3.2	UNet . . . . .	38
5.3.3	Utils . . . . .	39
5.3.4	Treinamento . . . . .	39
5.3.5	Inferência . . . . .	41
<b>5.4</b>	<b>Treinamento</b> . . . . .	<b>42</b>
<b>5.5</b>	<b>Avaliação</b> . . . . .	<b>43</b>
<b>6</b>	<b>RESULTADOS E DISCUSSÕES</b> . . . . .	<b>44</b>
<b>6.1</b>	<b>Resultados iniciais</b> . . . . .	<b>44</b>
<b>6.2</b>	<b>Investigação</b> . . . . .	<b>44</b>
<b>6.3</b>	<b>Discussões</b> . . . . .	<b>47</b>
<b>7</b>	<b>CONCLUSÃO</b> . . . . .	<b>49</b>
<b>8</b>	<b>TRABALHOS FUTUROS</b> . . . . .	<b>50</b>
	<b>REFERÊNCIAS</b> . . . . .	<b>51</b>
<b>A</b>	<b>APÊNDICE</b> . . . . .	<b>54</b>
<b>A.1</b>	<b>dataset.py</b> . . . . .	<b>54</b>
<b>A.2</b>	<b>unet.py</b> . . . . .	<b>56</b>
<b>A.3</b>	<b>utils.py</b> . . . . .	<b>57</b>
<b>A.4</b>	<b>treinamento.py</b> . . . . .	<b>59</b>
<b>A.5</b>	<b>inferencia.py</b> . . . . .	<b>60</b>

# 1 INTRODUÇÃO

## 1.1 Contextualização

Câncer de mama é uma das formas de câncer mais comumente diagnosticadas, sendo uma das principais ameaças para saúde feminina no mundo. No Brasil houve mais de 73.610 novos casos diagnosticados em 2022 e 18.032 óbitos em 2020 (INCA, 2022).

O diagnóstico do câncer ainda nas fases iniciais aumenta significativamente a chance de sucesso da terapia, e consequentemente, a chance de sobrevivência (DEWANGAN et al., 2022), o que impulsiona as pesquisas na busca por novos ou melhores métodos para a detecção e diagnóstico precoce.

Essa forma de câncer é causada pela divisão irregular de células provenientes dos tecidos mamários, que resulta na formação do tumor (SOHAIL et al., 2021). Por isso, a forma mais efetiva de diagnóstico é a contagem de núcleos celulares no estágio de mitose, conhecido como índice de atividade mitótica (SOHAIL et al., 2021).

Essa contagem ocorre durante a análise de tecido celular em um microscópio por um especialista, em que os núcleos celulares no estágio de mitose são contados (SOHAIL et al., 2021), com o tipo de especialista que se responsabiliza por esse tipo de análise sendo os patologistas, profissionais especializados na análise de amostras de fluido ou tecido.

Contudo, esse método tem uma dificuldade: células mitóticas e não-mitóticas são difíceis de distinguir pelas suas similaridades, e graças a isso a análise termina sendo um processo demorado, que requer muita atenção e ainda sim é suscetível ao erro humano (MAN; YANG; XU, 2020), com mesmo patologistas experientes tendo diferenças em análise do mesmo tecido celular (AKHTAR et al., 2019).

Devido a essas dificuldades o diagnóstico auxiliado pela computação na medicina, denominada *Computer Aided Diagnostics* (CAD) se tornou uma área bastante ativa na histopatologia de células mitóticas, especialmente com os avanços recentes em aprendizado de máquina e visão computacional que já obtêm resultados de alta precisão (HOUSSEIN; EMAM; ALI, 2022).

Porém, mesmo com esses avanços a patologia manual não se tornou obsoleta e continua sendo o padrão para diagnóstico (AKHTAR et al., 2019), com métodos baseados em aprendizado de máquina mais frequentemente sendo usados apenas de maneira auxiliar.

Com essa informação vem a tópico a importância do treinamento de novos patologistas, que geralmente ocorre com a análise microscópica de raízes de *allium cepa*, nome científico das cebolas, devido ao tamanho relativamente grande dos seus cromossomos, a estabilidade das células e a constante disponibilidade com baixo custo (NEFIC et al., 2013).

Neste projeto será proposto a implementação e treinamento de uma arquitetura de segmentação existente, baseada em aprendizado profundo, para auxiliar na detecção de células mitóticas em imagens microscópicas de raízes de *allium cepa*, com objetivo de auxiliar estudantes de histopatologia durante seu aprendizado.

## 1.2 Motivação e Justificativa

Câncer de mama é uma das causas mais comuns de morte feminina, sendo a causa de aproximadamente 1 a cada 40 falecimentos entre mulheres (HAGOS; MERIDA; TEUWEN, 2018), com a chance de sobrevivência e sucesso do tratamento aumentando proporcionalmente com o quanto cedo acontecer o diagnóstico.

Em países em que o diagnóstico e tratamento são suficientemente disponíveis e avançados resultando em até 80% de chance de remissão do tumor (MASUD; RASHED; HOSSAIN, 2022).

No estado da arte de CADs para esse problema a grande maioria das pesquisas inclui 2 fatores: A detecção de células mitóticas e o uso de *deep learning*, com estudos conseguindo ótimos resultados que também constantemente evoluem a medida que novos métodos e técnicas são desenvolvidas. Porém, nenhum desses procedimentos até agora se tornou o padrão para diagnóstico, com a resposta final ainda necessitando da análise manual de um patologista treinado (AKHTAR et al., 2019).

Seguindo essa ideia, existe uma área relacionada em que inteligência artificial não é frequentemente aplicada: o treinamento desses patologistas. Esse treinamento ocorre com uma simulação da tarefa de identificação das células nos vários estágios da mitose, porém devido à melhor visualização e disponibilidade, são analisados tecidos vindos de raízes de *allium cepa* (NEFIC et al., 2013).

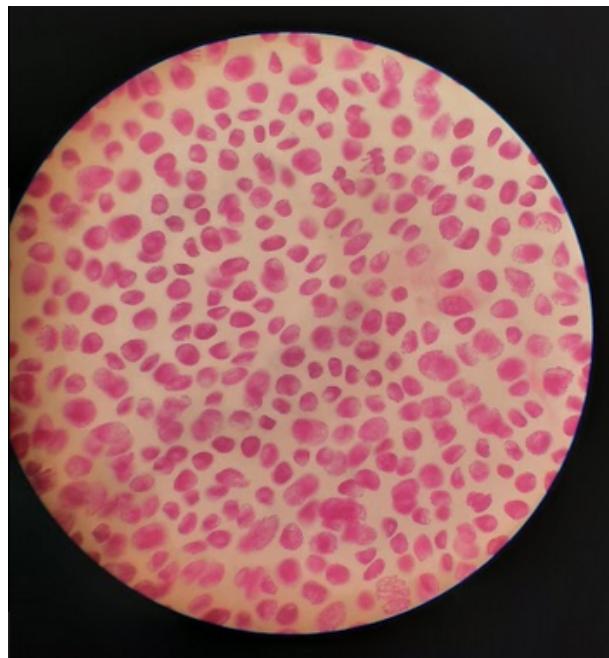
Essas imagens de treinamento porém tem algumas diferenças - além do fato das células virem de algo diferente do tecido mamário, elas também comumente são de menor qualidade, devido ao fato de que os aprendizes de patologia podem não ter acesso a um microscópio de qualidade comparável a um usado em clínicas de diagnóstico, com um exemplo dessas imagens sendo visualizada na Figura 1.

Durante o treinamento os aprendizes de histopatologistas executam o processo manual de identificação de células mitóticas nessas imagens, com a validação dos resultados obtidos sendo feita por um histopatologista com mais experiência. Com isso em mente, uma arquitetura de CAD suficientemente precisa pode auxiliar na validação dos resultados desse aprendiz ou, no melhor dos casos, até substituir a necessidade desse analista experiente.

Tendo em vista as ideias apresentadas, esse projeto tem como objetivo implementar e treinar uma arquitetura de segmentação de imagens para identificar células mitóticas em imagens microscópicas de células vindas de raízes de *allium cepa*, com a proposta de que essa rede pode

ser usada para auxiliar estudantes de histopatologia.

Figura 1 – Imagem microscópica de tecido celular de uma raiz de *allium cepa*, tirada pela câmera do celular de uma das estudantes relacionadas ao projeto



Fonte: Exemplo da base de dados fornecida ao autor.

## 1.3 Objetivo

### 1.3.1 Geral

O objetivo deste projeto é implementar e treinar uma arquitetura de visão computacional para detectar células mitóticas em imagens microscópicas de raízes de *allium cepa* com resultados precisos o suficiente para auxiliar histopatologistas em treinamento.

### 1.3.2 Específico

- Preparar uma base de dados composta por imagens microscópicas de raízes de *allium cepa* de maneira adequada para treinamento.
- Replicar a implementação de uma arquitetura reconhecida por obter resultados em segmentação.
- Treinar a arquitetura implementada, tentando extrair os melhores resultados possíveis da base de dados obtida.
- Avaliar a capacidade do modelo de auxiliar estudantes de histopatologia.

## 2 REVISÃO BIBLIOGRÁFICA

Para a construção deste trabalho foi adotada uma pesquisa exploratória, que consiste em pesquisar e coletar dados e informações correspondentes aos assuntos deste projeto para analisar os estudos mais recentes, ou seja, o Estado da Arte.

Com objetivo não só de contextualizar o estado da arte, mas também selecionar a arquitetura de segmentação que será utilizada no trabalho, foi realizada uma revisão sistemática com busca de artigos e projetos relacionados ao tema deste projeto em repositórios científicos.

### 2.1 Revisão Sistemática

A revisão sistemática é um método de investigação científica que se limita em reunir estudos ou materiais relevantes de vários autores com o intuito de executar uma análise crítica e abrangente da literatura. A revisão sistemática foi executada nas seguintes etapas:

#### 2.1.1 Seleção de repositórios científicos

Os seguintes repositórios foram selecionados para a busca de projetos:

- IEEE Xplore (<https://ieeexplore.ieee.org/>), devido a ser um dos maiores agregadores de artigos.
- Springer Link (<https://link.springer.com/>), devido a mesma motivação do IEEE Xplore.
- arXiv (<https://arxiv.org/>), devido a ser um dos maiores repositórios para artigos relacionados a aprendizado de máquina.

#### 2.1.2 Refinamento da busca

Os seguintes critérios foram aplicadas para a seleção de artigos:

- Critério 1 - Artigos completos em português ou inglês;
- Critério 2 - Artigos que tratem de visão computacional;
- Critério 3 - Artigos com aplicação ou utilidade no diagnóstico de câncer de mama;
- Critério 4 - Artigos com texto completo disponível na web;
- Critério 5 - Artigos recentes, de 2017 até o ano atual;

### 2.1.3 Palavras-chave(*keywords*) e *String* de busca

As seguintes palavras chave ou alternativas serão usadas na busca:

- *deep learning* OU *neural network* OU *machine learning*
- *computer vision* OU *semantic segmentation* OU *object segmentation* OU *feature extraction*
- *mitosis detection* OU *mitosis segmentation* OU *breast cancer* OU *breast histopathology*

A *string* de busca resultante é representada na Tabela 1.

<i>deep learning OR neural network OR machine learning</i>
AND
<i>computer vision OR semantic segmentation OR object segmentation OR feature extraction</i>
AND
<i>mitosis detection OR mitosis segmentation OR breast cancer OR breast histopathology</i>

Tabela 1 – String de busca resultante

### 2.1.4 Critérios de exclusão

Artigos retornados pela busca poderão ser selecionados ou excluídos de acordo com os seguintes critérios:

- Critério 1 - Exclusão de estudos duplicados em diferentes repositórios;
- Critério 2 - Exclusão de estudos com base na leitura do resumo;
- Critério 3 - Exclusão de estudos a partir da leitura dinâmica do texto completo;

### 2.1.5 Seleção dos artigos retornados

Repositório	Retornados	Selecionados
<i>IEEE Xplore</i>	67	8
<i>arXiv</i>	43	4
<i>Springer Link</i>	180	3
<b>Total</b>	<b>290</b>	<b>15</b>

Tabela 2 – Total de resultados retornados e selecionados

## 2.2 Trabalhos Relacionados

Nesta seção são relatadas as principais ideias e desenvolvimentos encontrados na revisão, que servem como fundação teórica do estado da arte na literatura, demonstrando os principais métodos que foram propostos e seus respectivos resultados, para consideração do que pode ou não ser utilizado no desenvolvimento do projeto.

No artigo de ZHU et al.(2017) tinha como objetivo a classificação de tumores de câncer de mama em imagens de ressonância magnética com o uso de redes neurais convolucionais. O problema foi abordado com 3 métodos diferentes:

- Treinamento com um modelo vazio, sem treinamento prévio, em que apenas as imagens dos tumores foram usadas pro treinamento.
- Treinamento com um modelo pré-treinado em imagens naturais, que foi ajustado com o treinamento nas imagens de tumores.
- Treinamento baseado em classificação por uma *Support Vector Machine* (SVM), que classificou as imagens após a extração de características utilizando uma rede pré-treinada em imagens reais.

Após os testes, os melhores resultados foram obtidos no modelo de extração de características que foram treinadas em uma SVM.

No trabalho de HAGOS; MERIDA; TEUWEN(2018) foi testado se a adição de outras informações que um radiólogo tem, mais especificamente a diferença simétrica, pode trazer melhorias na detecção. Para isso foi proposta uma CNN de múltiplas entradas que também aceitava essa informação sobre a simetria além da imagem da ressonância magnética, que foi comparada a uma *Convolutional Neural Network* (CNN) que apenas aceitava as informações da imagem. A rede proposta pelo artigo obteve resultados pouco melhores comparados a rede padrão, porém nada considerável ,com as melhorias ainda estando dentro da margem de erro.

A abordagem proposta por LI; WU; WU(2019) se trata do uso de CNNs não só para classificação, mas também para selecionar regiões das imagens que tem mais características distintivas. No primeiro estágio CNNs são aplicadas para selecionar as seções de imagens com mais características distintivas, a nível tanto celular com imagens menores quanto a nível de tecido com imagens maiores. Após isso, as seções selecionadas são normalizadas e passam pelo processo de extração de características por uma rede baseada no modelo *ResNet50*. Finalmente, as características extraídas são classificadas usando SVM. A arquitetura proposta no artigo obtém resultados competitivos com outros modelos do estado da arte.

Na pesquisa de LI et al.(2019) foi utilizado um modelo que já obteve sucesso na segmentação de massa cancerígena em mamografias por Raio X, a arquitetura *U-Net*. A proposta desse projeto foi a aplicação de Attention Gates (AGs), um método que direciona o modelo a

buscar apenas em regiões de maior interesse enquanto desativa as ativações de reconhecimento em áreas irrelevantes. Um extrator de características foi utilizado para decodificar a imagem enquanto um modelo de *U-Net* implementado com adição de AGs re-codifica e retorna a camada de segmentação. Esse modelo obteve resultados melhores comparados a apenas *U-Net* e outros modelos do estado da arte.

No trabalho de AKHTAR et al.(2019) foi utilizado *transfer learning* em imagens de histopatologia de tecido mamário para auxiliar a detecção de câncer de próstata. Devido a pouca quantidade de dados disponíveis de histopatologia de próstata, um modelo é treinado em bases de dados de mamografia antes de ser treinado com os dados de câncer de próstata em si. Esse método de treinamento demonstra ganhos em relação ao pré-treinamento em imagens naturais do dataset *ImageNet*.

MAN; YANG; XU(2020) lida com a dificuldade de trabalhar em certos datasets de histopatologia de mama, em que as imagens tem resoluções altas de mais para serem trabalhadas por CNNs tradicionais. Nesses casos são aplicados métodos que classificam apenas segmentos da imagem, com o problema que esse trabalho aborda sendo os erros nessa classificação, que acontece por exemplo quando seções de um tumor benigno aparecem no mesmo segmento de imagem que contém um tumor maligno. Para resolver isso é treinada uma *Generative Adversarial Network* de maneira não supervisionada para filtrar apenas os segmentos de imagens com características mais claras, com esses segmentos filtrados sendo então classificados pela CNN *DenseNet*. Essa arquitetura atingiu os melhores resultados comparado com o estado da arte em imagens com zoom de 40X e 100X, mas perdeu para outros modelos em imagens com zoom de 100X e 400X.

A pesquisa de SEBAI; WANG; AL-FADHLI(2020) trabalha com a segmentação de células mitóticas, criando dois modelos que trabalharão com duas formas de anotação: *weak labels* onde apenas o centro da região de interesse é identificada para o modelo e *strong labels*, anotações da região de interesse completa a nível de pixel. Na fase de detecção os mapas de segmentação desses modelos são então combinados para obter as detecções finais. O modelo combinado obteve resultados superiores a outros modelos de detecção comparáveis.

ALOM et al.(2020) propõe uma rede de modelos de *deep learning* chamada *MitosisNet* que utiliza três modelos para executar seu processo: um modelo *Recurrent Residual U-Net* (R2UNet) para segmentação, um modelo baseado na R2UNet com regressão para detecção e por último *Inception Recurrent Residual Convolutional Neural Network*(IRRCNN) para classificação. Os modelos de segmentação e detecção são executados em paralelo e têm seus resultados combinados, que passam por um estágio final de validação pela IRRCNN. Os resultados do experimento demonstram competitividade com outros modelos do estado da arte e velocidade de análise o suficiente para ser aplicada em prática clínica.

O trabalho de ZHENG et al.(2020) combina vários métodos de machine learning em série para detecção e classificação de tumores de várias métodos de diagnóstico diferentes: Ressonâ-

cia magnética, ultrassom, tomossíntese mamária e mamografia. Esse modelo denominado *Deep Learning assisted Efficient AdaBoost Algorithm* contém 4 camadas principais: Primeiramente um auto codificador e decodificador para aprendizado de características seguido por um algoritmo baseado em *AdaBoost* para treinar classificadores. Continuando a análise com o uso de uma CNN com *Long-Short Term Memory* que termina com uma regressão logística *softmax* para retornar a probabilidade dos valores de classe encontrados. Esse modelo conseguiu alta precisão nos bancos de dados testados.

HIRRA et al.(2021) propõem um método de extração de características e classificação baseado em *Deep Belief Networks* (DBNs) que se utilizam tanto de treinamento não supervisionado quanto supervisionado chamado *Pa-DBN-BC*. Além do processo complexo de pré-processamento e criação da base de dados de treinamento elimina as seções não essenciais e separa os segmentos em cancerígenos e não cancerígenos com uma razão de 1:1 sobre essas classes. O modelo em si, baseado em DBNs, composto por camadas de *Restricted Boltzmann Machine* que são treinadas inicialmente de maneira não-supervisionada para inicializar os pesos da rede, e apenas após isso que ocorre o treinamento supervisionado com a adição de uma camada de saída na rede e classificação dos dados de treinamento. Para a fase de teste, os dados separados passam pelo mesmo pre-processamento que os dados de treinamento passaram. Esse modelo atingiu resultados superiores aos seus competidores no estado da arte.

Um modelo heteromorfo composto de várias redes neurais é projetado por IQBAL; QURESHI(2022) chamado *BreastUNet*. A primeira rede se utiliza do processo de *Stain Normalization* para minimizar a variação que pode existir nas imagens histopatológicas que podem vir de variados laboratórios com diferentes padrões, seguidas por um modelo baseado em *Mask R-CNN* para otimizar bancos de dados com *weak annotations*, em onde apenas o centro da célula mitótica é marcada. Após isso os dados passam pelo *GLocal Pyramid Pattern* para extração de características de textura que é verificado por um *Class Activation Map*, que por último passa as características por um modelo customizado baseado na *U-Net* para formar o modelo *BreastUNet*. A avaliação demonstra que esse modelo apresenta consideráveis ganhos em precisão comparados com outros modelos do estado da arte.

Um algoritmo de otimização chamado *Improved Marine Predators Algorithm* é utilizado e avaliado por HOUSSEIN; EMAM; ALI(2022). Esse algoritmo baseado em como um predador marítimo ataca sua presa é utilizado para otimização durante o treinamento de uma CNN *ResNet50*, o que resultou na melhor performance quando comparado a vários outros algoritmos de otimização usados no treino da mesma CNN.

O modelo *Back Propagation Boosting Recurrent Wienmed* (BPBRW) é combinado com o algoritmo de otimização híbrido *Hybrid Krill Herd African Buffalo Optimization* (HKH-ABO) no trabalho de DEWANGAN et al.(2022), com o HKH-ABO sendo uma combinação híbrida de dois algoritmos de otimização, *Hybrid Krill Herd* e *African Buffalo Optimization* para classificação de tumores entre benigno e maligno em imagens de ressonância magnética. A abordagem proposta

atingiu melhores valores de precisão e menor taxa de erro que as propostas em que foi comparado.

O trabalho de CAYIR et al.(2022) faz uma proposta de modelo para classificação de células mitóticas chamado *MITNET*. Esse modelo é dividido em dois componentes principais: *MITNET-det* que tem como responsabilidade fazer detecção de núcleos celulares nas imagens em 3 passos, iniciando com a extração de características com um modelo baseado em *YOLOv4*. Com os mesmos sendo então transformados em mapas de características de diferentes escalas com o uso de camadas de *pooling*, com esses mapas de diferentes escalas sendo repassados para a última seção que retorna as *bounding boxes*, classes e valores de confiança. Com os núcleos detectados, eles são então classificados entre mitótico e não mitótico pela segunda parte do modelo, denominada *MITNET-rec*, que é uma CNN baseada no modelo VGG-11. Na avaliação o *MITNET-det* obtém bons resultados de detecção, com o *MITNET-rec* obtendo resultados significativamente melhores que outros classificadores nas bases de dados usadas.

BENAGGOUNE et al. (2022) propõem uma *pipeline* de soluções para automatização de classificação no índice de proliferação KI-67, que é um índice essencial para patologistas diagnosticarem e escolher o tratamento adequado para câncer de mama. O principal diferencial desse projeto é como ele lida com núcleos celulares sobrepostos: Após a segmentação os núcleos são separados entre núcleos únicos e sobrepostos, com um algoritmo de separação *Watershed* sendo usado nas células sobrepostas para separação. Com os núcleos separados as características são extraídas e classificadas com um algoritmo de *Random Forest*. O projeto obteve bons resultados comparando ao estado da arte porém tem a desvantagem de que as áreas de interesse precisam ser manualmente selecionadas.

## 2.3 Conclusões

Além de aprender sobre o estado da arte da segmentação de mitose e CADs para diagnóstico de câncer de mama, também foi notada uma arquitetura em específico que apesar de não ser usada de maneira isolada em nenhum dos projetos faz parte das arquiteturas dos trabalhos de LI et al. (2019), ALOM et al. (2020) e IQBAL; QURESHI (2022), a U-Net.

Devido a essa popularidade na área da medicina, a U-Net foi selecionada como a arquitetura de aprendizado profundo para segmentação de imagens que será usada nesse projeto.

# 3 FUNDAMENTAÇÃO TEÓRICA

## 3.1 *Perceptrons* e Redes Neurais Artificiais

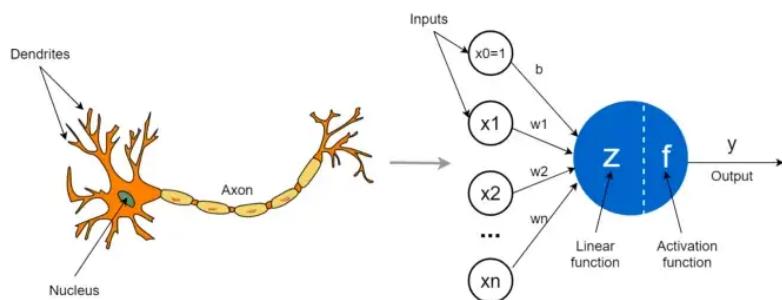
Os métodos tradicionais de inteligência artificial, que utilizam ifs, *loops* e portas lógicas como *AND* e *OR* dependem da manipulação lógica feita pelo programador para a tomada de decisões, que já poderia obter bons resultados dependendo do problema.

Porém havia também um interesse em replicar o método de aprendizado do cérebro humano - onde a resposta é decidida baseada na prática, com tentativas e erros até que eventualmente o resultado é atingido e as decisões para chegar ao mesmo são fixadas na memória (FERREIRA, 2017).

Os neurônios são células que compõem a maior parte do cérebro e precisariam ser simulados para que essa forma de aprendizado possa acontecer, com o primeiro modelo matemático de um neurônio sendo criado por MCCULLOCH; PITTS (1943), chamado de modelo *McCulloch-Pitts*.

No final da década de 50, o psicólogo Frank Rosenblatt colocou esse modelo em prática utilizando um computador *IBM 704* (LEFKOWITZ, 2019), melhorando também a capacidade de retenção de informação e implementando o primeiro algoritmo de treinamento (ROSENBLATT, 1957), denominando esse modelo como *Perceptron*, representado à direita na Figura 2.

Figura 2 – Representações de um neurônio biológico(à esquerda) e um *perceptron*(à direita)



Fonte: PRAMODITHA(2021)

O perceptron é um classificador linear que recebe as entradas e retorna um valor entre 0 e 1. Sua composição consiste dos seguintes elementos:

- Entrada: A quantidade de valores que será enviada para o perceptron, denominados por  $x_1, x_2, x_3, \dots, x_n$  .

- Pesos: Cada valor de entrada tem um peso associado, que indica a importância daquela entrada em específico, denominados por  $w_1, w_2, w_3, \dots, w_n$ .
- Soma Ponderada: Soma de cada uma das entradas multiplicadas pelos respectivos pesos, geralmente representados por um  $\Sigma$ .
- Função de ativação: Função que define que resultado o valor da soma ponderada retornará na saída, geralmente representado por  $f(x)$  ou  $g(x)$ .
- Saída: Resultado do *perceptron*, geralmente um valor entre 0 e 1.

Com essa configuração já é possível obter respostas simples, porém *perceptrons* geralmente são criados com pesos aleatórios, que faz com que os resultados sejam imprevisíveis. Para viabilizar o uso de um *perceptron* é necessário um processo de treinamento, onde dados reais, com a entrada e resposta, são processados pelo mesmo iterativamente, ajustando os pesos para se adaptar aos dados passados durante esse treinamento(O'SHEA; NASH, 2015).

Porém, um *perceptron* em si apenas consegue resolver problemas simples, além de limitações com tipos de problemas que não podem ser resolvidos, como portas lógicas *XOR* (LI et al., 2021). Essa limitação é resolvida com o aumento da complexidade, pois assim como o cérebro humano é formado de neurônios interconectados, os *perceptrons* também podem ser agregados em uma estrutura maior, formando as Redes Neurais Artificiais (RNAs).

RNAs na sua forma mais básica contém 3 segmentos: A camada de entrada, que tem o tamanho definido pelos dados quais a rede neural se adaptará, a camada escondida, que é composta de uma ou mais camadas de *perceptrons* e a camada de saída, que tem sua escala definida pelo resultado que deve ser obtido(O'SHEA; NASH, 2015). A Figura 3 contém um exemplo dessas redes básicas.

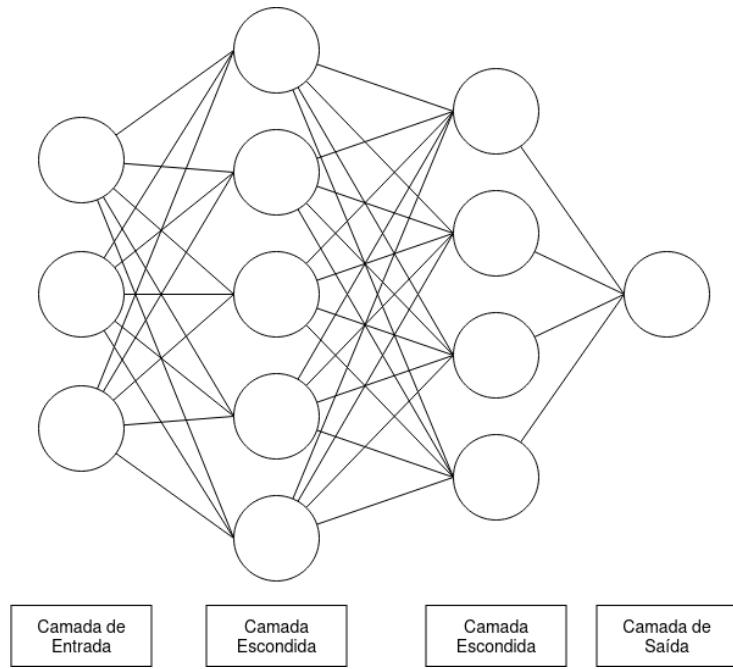
*Backpropagation* é o principal algoritmo usado para treinamento de redes neurais (FERREIRA, 2017), onde as entradas são processadas por todas as camadas da RNA (esse procedimento é denominado *feed-forward*), e a resposta que é obtida na saída é comparada com o resultado real dos dados de treinamento, que resulta em um valor denominado Erro(ou perda) baseado no quanto longe da resposta real a saída estava (O'SHEA; NASH, 2015).

Esse valor então é usado para calcular novos valores para os pesos da rede neural, sendo propagados em ordem reversa com o objetivo de chegar cada vez mais perto do resultado real, isso é, com o objetivo de reduzir o valor de erro em si, adaptando a rede para os dados que foram usados no treinamento (O'SHEA; NASH, 2015).

## 3.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais, do inglês *Convolutional Neural Networks* (CNNs), são uma evolução das RNAs que continuam utilizando o conceito de treinamento e aprendizado,

Figura 3 – Representação de uma rede neural simples com uma camada de entrada de tamanho 3, duas camadas escondidas com 5 e 4 perceptrons respectivamente, e uma camada de saída de tamanho 1 para resposta binária



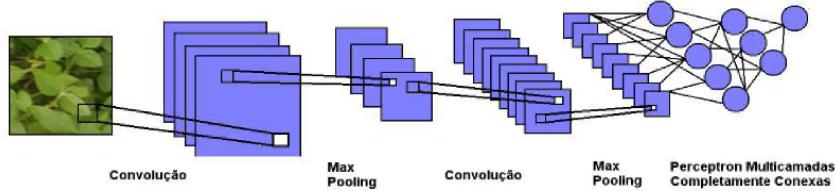
Fonte: Do autor.

com uma entrada que passa pelas camadas de neurônios com seus respectivos valores de peso e expressa um resultado final de saída, com a principal diferença que CNNs são feitas particularmente para trabalhar com tarefas envolvendo a análise de imagens (O'SHEA; NASH, 2015).

Os primeiros trabalhos que tentavam identificar características por um meio visual trabalhavam com problemas simples, com imagens pequenas e com apenas um canal de cor, como por exemplo o reconhecimento de dígitos numéricos escritos a mão (LECUN; BOSEN; DENKER, 1989). Isso era algo que RNAs já conseguiam obter resultados satisfatórios. Contudo, o aumento da complexidade dos dados de entrada não poderia ser resolvido por apenas aumentar a quantidade de neurônios ou camadas nas RNAs.

Para isso o paradigma das CNNs foi criado, com o modelo pioneiro sendo a *LeNet* (LECUN et al., 1998), que além de lidar melhor com maior dimensionalidade nos dados de entrada, também demonstrou melhor desempenho nas tarefas mais simples que as RNAs já conseguiam resolver.

Assim como redes neurais, CNNs necessitam de uma camada de entrada e saída, mas a diferença ocorre nas suas camadas escondidas, que são compostas por 3 tipos diferentes: Camadas de convolução, camadas de *pooling* e camadas completamente conectadas (FERREIRA, 2017). Essa arquitetura é considerada o modelo padrão das CNNs e é representada na Figura 4.

Figura 4 – Estrutura da CNN *LeNet*

Fonte: FERREIRA (2017)

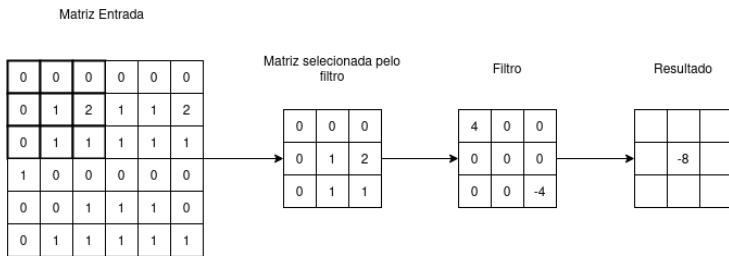
### 3.2.1 Camada de Convolução

A camada de convolução executa o trabalho principal dentro de uma CNN, que é a identificação de características da imagem de entrada, baseados na operação matemática de convolução (SOUZA, 2021). Que faz a utilização de filtros que percorrem a imagem e respondem a várias características encontradas na imagem de uma maneira que simula os receptores visuais que existem no olho humano. (LI et al., 2021)

Essa operação funciona transformando a imagem em uma matriz de tamanho  $X * Y * Z$ , em que  $X$  e  $Y$  correspondem ao tamanho da imagem e  $Z$  é a quantidade de dimensões, com esse valor sendo comumente a quantidade de canais de cor na imagem.

Cada uma dessas matrizes são então percorridas pelos filtros, que são ativados ao encontrar uma característica específica em uma certa posição da matriz de entrada, com cada filtro resultando um mapa de atributos que será a saída desse elemento da camada, como é representada na Figura 5.

Figura 5 – Representação do funcionamento de um filtro. O segmento da matriz de entrada é processado com o resultado sendo a soma ponderada da aplicação do filtro, indicando a ativação do segmento filtrado.



Fonte: Adaptado de O'SHEA; NASH (2015)

Como esses filtros funcionarão durante a convolução é definido pelo arquiteto da CNN, com essa parametrização sendo o que definirá o sucesso ou fracasso da operação de convolução ocorrida nessas camadas (O'SHEA; NASH, 2015), com os principais parâmetros sendo:

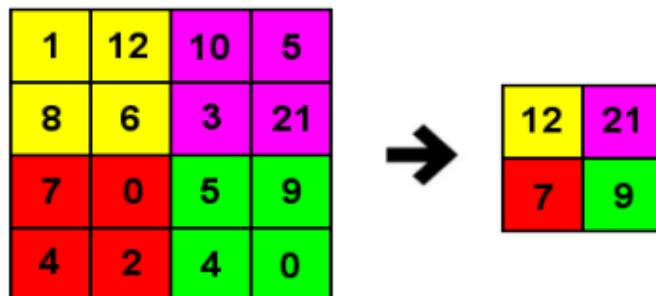
- Dimensão: O tamanho do filtro, definido por 2 valores  $X$  e  $Y$ .
- Profundidade: O tamanho da camada de convolução, equivalente a quantidade de neurônios em uma camada de RNA, que é o que define a quantidade de filtros diferentes que serão aplicados a entrada para a busca de características.
- Deslocamento: Define o quanto o filtro vai se mover em cada passo da análise da matriz de entrada. Se esse valor for muito baixo, o mesmo valor será analisado em múltiplos segmentos que causa uma sobreposição de múltiplas ativações na mesma parte da imagem. O oposto também é verdade, deslocamento muito grande causa partes da imagem a serem ignoradas.
- Zero-padding: Preenchimento com elementos de valor 0 aplicado nas bordas da matriz para permitir que o filtro se encaixe ao percorrer a imagem, para impedir que certas partes da matriz não sejam processadas devido aos valores de dimensão e deslocamento.

### 3.2.2 Camada de *pooling*

*Pooling* é uma forma de *down-sampling* que reduz a dimensão dos mapas de ativação resultantes da camada de convolução, com intenção de reduzir o processamento necessário pelas camadas posteriores da CNN enquanto ainda mantém as informações mais relevantes(O'SHEA; NASH, 2015).

A típica camada de *pooling* funciona usando o método de *max pooling*, que retorna o valor máximo local de um determinado segmento do mapa de atributos, como é representado na Figura 6.

Figura 6 – Aplicação de *max pooling* em uma imagem 4x4 utilizando um filtro 2x2. A redução do tamanho da imagem consequentemente também reduz o processamento das próximas camadas.



Fonte: FERREIRA(2017)

O resultado do *pooling* é uma versão menor das características que foram detectados pelos filtros das camadas de convolução, em que apenas os valores máximos e mais distintivos

foram mantidos. Um *pooling* de tamanho 2x2 como na Figura 6 já reduz o valor para apenas 25% do tamanho original.

### 3.2.3 Camadas completamente conectadas

As camadas completamente conectadas(ou conexas) se responsabilizam pelo aprendizado da rede, identificando quais filtros estão ou não retornando as características buscadas, após todo o processamento e *down-sampling* feito pelas camadas anteriores. Essa camada é composta por neurônios ou perceptrons interconectados, da mesma maneira que são formadas as camadas escondidas de uma RNA

## 3.3 Paradigmas de resposta

Nos primeiros usos de aprendizado de máquina para visão computacional a resposta era dada como um número entre 0 e 1, conhecido como valor de confiança, que indicava o quanto que a rede acredita que aquela classe de objeto está presente naquela imagem processada. Essa forma de resposta é chamada de classificação.

Essa forma de resposta é efetivamente utilizada para resolver vários tipos de problemas, como por exemplo o primeiro artigo a propor o termo convolução por LECUN; BOSER; DENKER (1989) que usava classificação para detecção de dígitos entre 0 a 9, com um valor de confiança para cada dígito numérico.

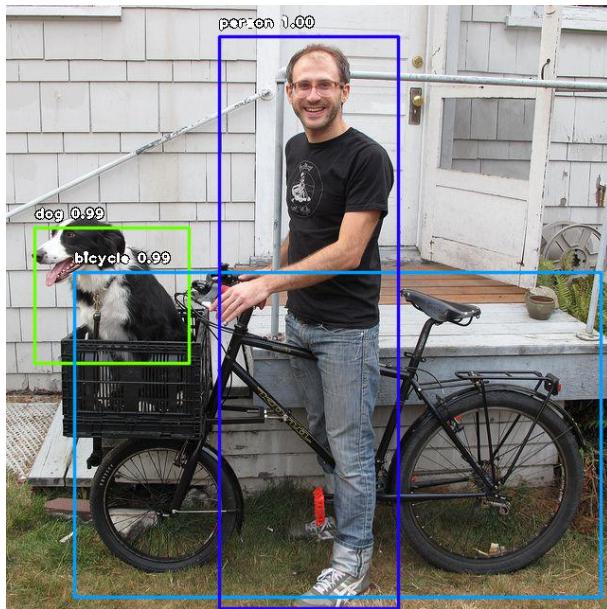
Porém, apenas a habilidade de prever se algo ou não faz parte da imagem analisada nem sempre é o suficiente, como por exemplo muitos problemas na área da medicina não só precisam da identificação de que algo está na imagem, mas sim a região exata de onde está o objeto (RONNEBERGER; FISCHER; BROX, 2015) e até a quantidade de vezes que esse tipo de objeto aparece na imagem.

O primeiro paradigma que atendeu a esses requisitos surgiu na década de 2000, a detecção de objetos, que analisa uma imagem e propõe uma ou mais regiões da imagem junto com o valor de confiança de onde o objeto ou objetos foram detectados como é representado na Figura 7. No entanto, essas primeiras formas de detecção tinham várias limitações, especialmente na precisão dos resultados, que resultou em uma estagnação desse método.

Detecção como forma de resposta voltou a evoluir quando *deep learning* começou a ser utilizado, com o artigo publicado por GIRSHICK et al. (2013) propondo a R-CNN: *Regions with CNN Features*, que se utilizava de CNNs em várias regiões da imagem para extração de características e uma SVM para classificar em quais dessas regiões os objetos existem.

Paralelo ao surgimento da detecção com uso de *deep learning* no início da década de 2010 também surgiu outro paradigma: a segmentação, que consiste em identificar e agrupar as

Figura 7 – Exemplo de detecção, com cachorro, pessoa e bicicleta sendo as classes detectadas

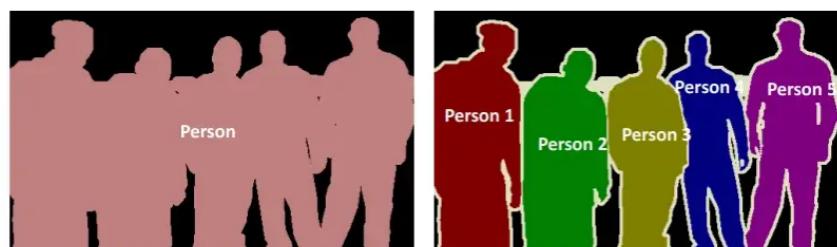


Fonte: KUMAR(2020)

classes a nível de pixel, que pode resultar em respostas significativamente mais precisas quando comparadas a detecção, que apenas retorna um retângulo com a região da imagem.

Além dessa caracterização pela seleção dos *pixels* em si, ainda existem também duas variações principais: segmentação semântica, em que cada pixel da imagem é associado com uma das classes e segmentação de instância, que marca cada instância da classe encontrada independentemente (SULTANA; SUFIAN; DUTTA, 2020), como é representado na Figura 8.

Figura 8 – Exemplos de segmentação de pessoas, com segmentação semântica na esquerda e segmentação de instância na direita



Fonte: LIU (2020)

### 3.4 U-Net

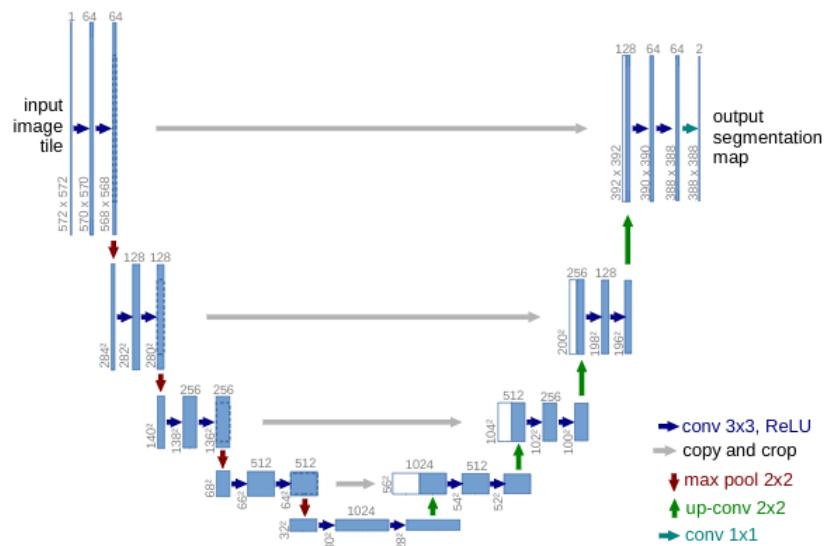
U-Net é uma arquitetura de redes convolucionais proposta por RONNEBERGER; FISCHER; BROX em que o principal objetivo é o processamento de imagens médicas com segmentação e eficiência de aprendizado o suficiente para trabalhar com bases de dados de tamanho limitado.

A principal ideia da arquitetura é combinar características extraídas em resoluções baixas e altas para montar uma saída baseada nessas informações. Para isso, operações de *downsampling* e *upsampling* são utilizadas junto com convolução para extração de características. A arquitetura em si é composta de duas partes:

- Contração, que segue a arquitetura típica de uma rede neural onde a imagem passa pelas camadas de convolução seguidas de *max pooling*, com o detalhe que a quantidade de extractores de características aumenta cada vez que a imagem é contraída.
- Expansão, que executa o oposto da contração, expandindo a imagem com *up-convolution* que expande a imagem e reduz o número mapas de características várias vezes até o retorno da camada de segmentação na saída da rede. Com detalhe que os mapas de características extraídos na contração são concatenados com as camadas equivalentes espelhadas no lado de expansão.

Esses dois segmentos da arquitetura resultam na formação de uma rede simétrica, que como pode ser visualizado na Figura 9, é equivalente a o formato da letra u, que é o que origina o título U-Net. Essa arquitetura é extensivamente usada como base ou segmento de várias arquiteturas ou *frameworks* para segmentação de imagens, especialmente na histopatologia.

Figura 9 – Visualização da estrutura da arquitetura U-Net



Fonte: (RONNEBERGER; FISCHER; BROX, 2015)

## 4 PROCEDIMENTO METODOLÓGICO

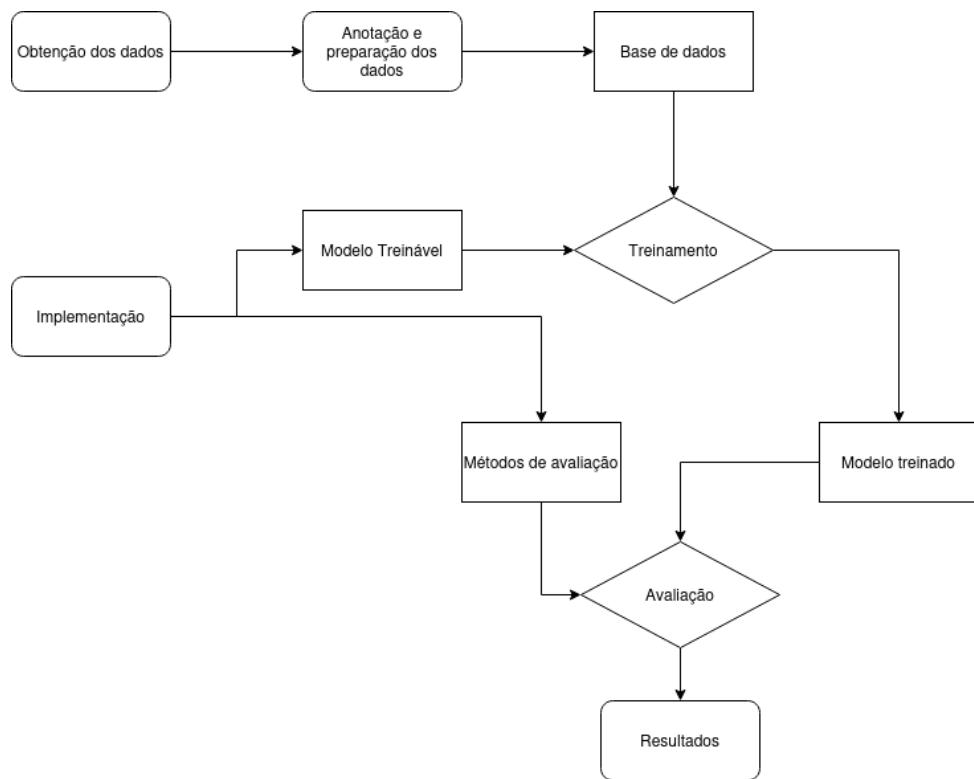
O procedimento metodológico deste projeto foi dividido e executado em 5 etapas principais, baseadas no método mais comum de treinamento de modelos de aprendizado de máquina. As 5 etapas são:

- Etapa 1 - Obtenção da base de dados: Uma base de dados de imagens microscópicas de tecidos vindos de raízes de *allium cepa* será fornecido por um time de patologistas do curso de medicina da UESPI, com indicações de quais células estão em fase de mitose.
- Etapa 2 - Preparação dos dados: Seguindo as indicações do especialista, as células mitóticas da base de dados passarão por um processo de anotação para identificar quais células das imagens estão em fase de mitose de uma maneira que possa ser alimentada a arquitetura de segmentação selecionada para o projeto.
- Etapa 3 - Implementação: A arquitetura U-Net e os métodos de avaliação pós-treinamento serão implementados com o auxílio das bibliotecas *tensorflow* e *keras*, além de outras ferramentas que podem terminar sendo usadas durante o recorrer do projeto.
- Etapa 4 - Treinamento: Ocorrerá o treinamento do modelo implementado na base de dados anotado, com testes e registro de quais parâmetros de treino obtiveram os melhores resultados, qual hardware utilizado e outros detalhes caso relevantes.
- Etapa 5 - Avaliação: Assim que for definido os parâmetros e métodos que obtém os melhores resultados, ocorrerão vários treinamentos usando esses parâmetros devido a natureza aleatória das redes neurais, com o resultado final sendo uma média dos treinamentos. A métrica de avaliação usada será *Intersection Over Union* (IoU) para comparar a segmentação da rede com as anotações da base de dados.

# 5 DESENVOLVIMENTO

Neste capítulo serão apresentados os métodos e práticas utilizados durante a execução do processo metodológico, que consistem principalmente na criação de uma base de dados treinável, a implementação do modelo treinável e seus métodos de avaliação, como é representado na Figura 10.

Figura 10 – Diagrama do processo de desenvolvimento desse projeto



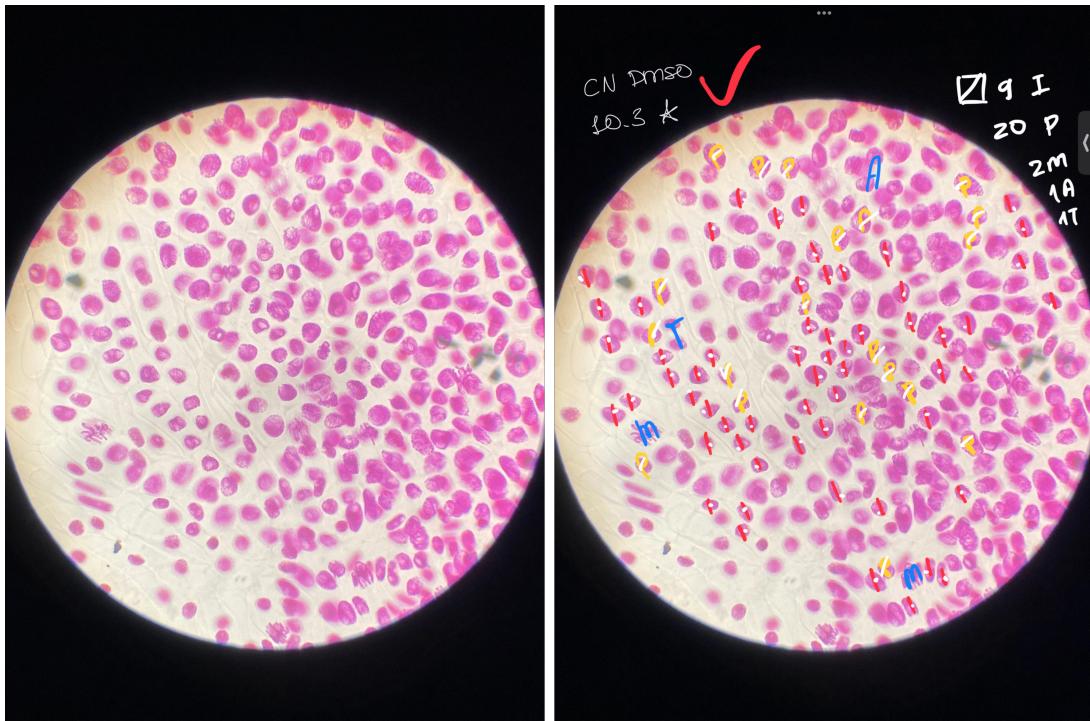
Fonte: Do autor.

## 5.1 Obtenção da base de dados

A base de dados fornecida para o projeto foi disponibilizada por pesquisadores do curso de medicina da UESPI, contendo um total de 118 imagens, com 59 imagens sendo as imagens originais que serão usadas para o treinamento e 59 imagens sendo versões anotadas por patologistas, identificando as células em qual fase exata do ciclo de vida de uma célula elas estão, com um exemplo representado na Figura 11.

As imagens anotadas utilizam uma letra para indicar em que fase a célula está, com a letra I indicando Interfase, P indicando Prófase, M indicando Metáfase, A indicando Anáfase

Figura 11 – Exemplo de anotação fornecida para cada uma das imagens da base de dados, com a imagem original na esquerda e a anotada na direita



Fonte: Do autor.

e T indicando Telófase. Para esse projeto apenas interessa identificar se as células estão em mitose, isso é, em divisão ativa, e dentre essas fases apenas uma delas não é considerada mitose, a Interfase (O'CONNOR, 2008), o que quer dizer que apenas as outras 4 são relevantes para o projeto.

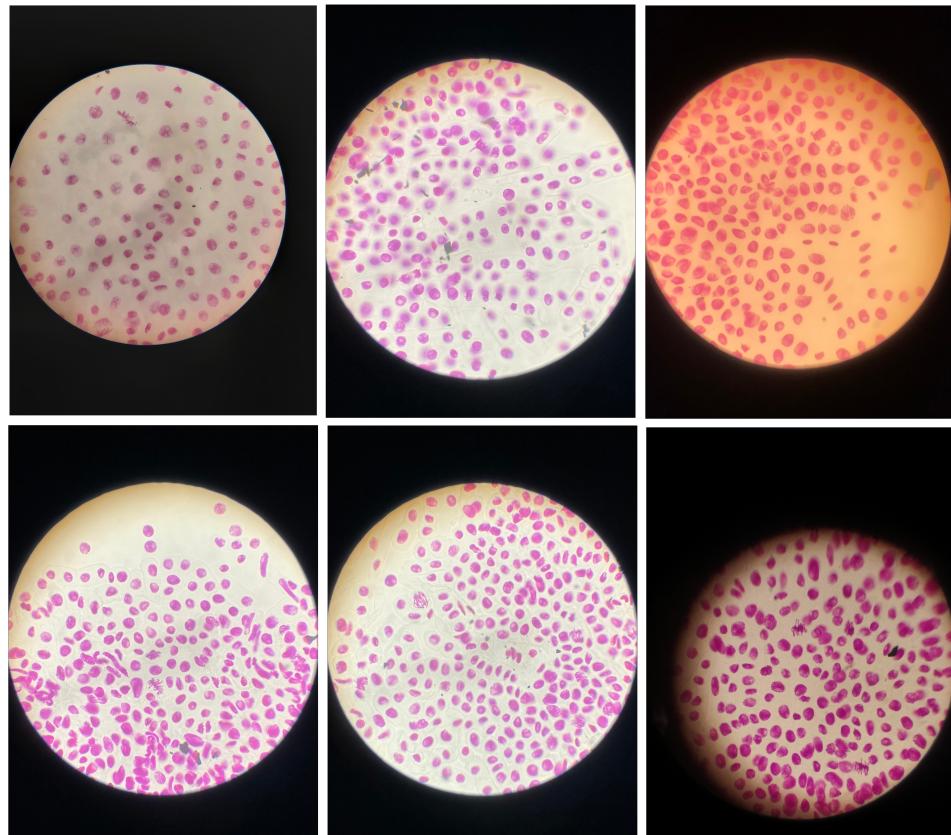
Na Figura 12 estão representados exemplos das imagens da base de dados fornecida, que são imagens de microscópio de tecidos celulares tiradas pela câmera de celular dos patologistas, com um corante aplicado para realçar o núcleo das células, além de outro líquido adicional a mais, que é o motivo pelo qual algumas imagens tem coloração diferente.

Importante também é chamar atenção ao fato de que nas imagens anotadas pelos patologistas, apenas em torno de 25% a 40% das células contêm uma anotação identificando a fase em que ela está, isso é devido ao fato de que, como mencionado na seção 1.2 desse documento, as imagens não terem clareza o suficiente para que seja possível identificar todas as células, então apenas as células com características mais perceptíveis foram anotadas pelos patologistas.

## 5.2 Preparação dos dados

Essa fase consiste em preparar os dados que foram fornecidos para utilização no treinamento e avaliação do modelo que será implementado e será dividida em 3 etapas principais:

Figura 12 – Exemplos de imagens da base de dados



Fonte: Do autor.

Anotação, Conversão e Divisão, com cada uma delas sendo descrita em detalhe a seguir.

### 5.2.1 Anotação

A fase de anotação consiste em registrar em quais regiões das imagens estão as células em estado de divisão, de acordo com as indicações das imagens anotadas pelos patologistas disponibilizadas para esse projeto.

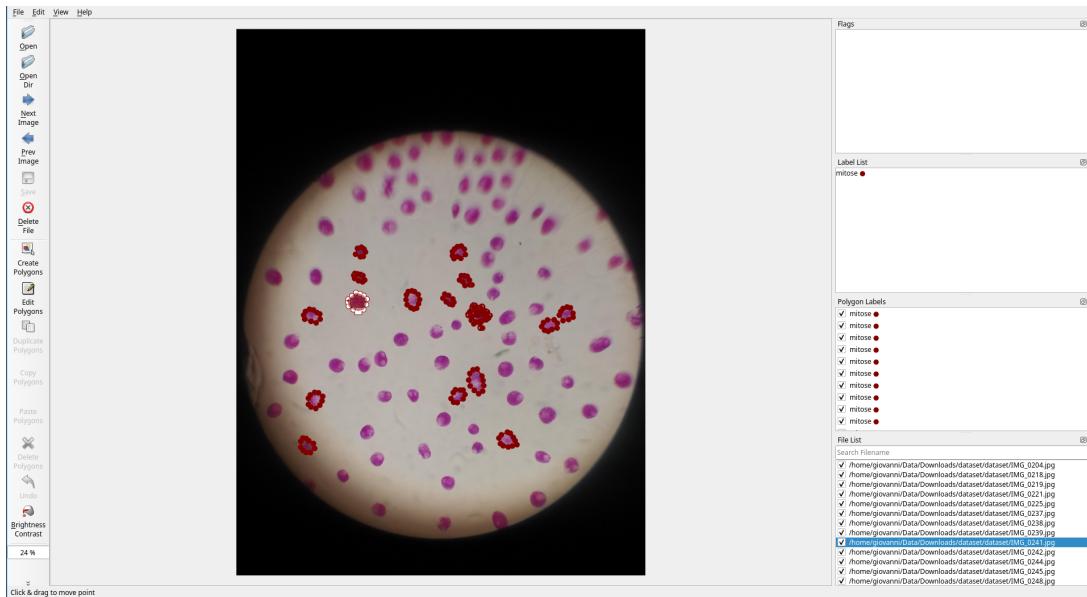
As anotações em si serão feitas apenas nas células que estão em divisão para seguir o mesmo padrão dos *datasets* públicos de células mitóticas ICPR 2012 e ICPR 2014, que obtiveram bons resultados em vários artigos publicados, como os trabalhos de MAHMOOD et al.(2020) e SEBAI; WANG; AL-FADHLI(2020).

Esse formato de anotação funciona pois o modelo aprende as características das células que estão em mitose e das células que não estão em mitose, que são consideradas da classe fundo ou *background*, isso é, não relevante para a segmentação.

Para realizar as anotações em si será utilizada a ferramenta *labelme*, versão 5.2.0, que permite abrir essas imagens e executar essas anotações em formato de polígonos, como representado

na Figura 13.

Figura 13 – Visualização do uso da ferramenta *labelme* para anotação da base de dados fornecida



Fonte: Do autor.

Ao salvar as anotações a ferramenta *labelme* cria um arquivo *json* indicando em quais coordenadas estão as vértices de cada polígono criado, também indicando a qual classe pertence esse polígono.

### 5.2.2 Conversão

As anotações feitas na etapa anterior são salvas no formato interno da ferramenta *labelme*, para facilitar modificações futuras pela mesma. Porém, esse formato não é ideal para alimentar modelos de aprendizado de máquina: apesar de ser possível, é necessário uma grande quantidade de código para converter as imagens e anotações para um formato ideal para o treinamento.

Nesse caso, a recomendação é converter as anotações para um formato mais padronizado e reconhecido, com o formato escolhido para esse projeto foi o formato do desafio *Visual Object Classes* (VOC), uma competição anual de reconhecimento e detecção visual de objetos (EVERINGHAM et al., 2010) que ocorreu entre 2005 e 2012 sendo organizada pela PASCAL<sup>1</sup>.

A ferramenta *labelme* já oferece no seu repositório de código fonte no *github* alguns *scripts* para fácil conversão das anotações do seu formato próprio para alguns formatos mais populares de bases de dados - incluindo o formato VOC que será usado nos dados anotados desse projeto com o *script* `labelme2voc.py`.

<sup>1</sup> Comunidade de pesquisadores focada na análise e modelagem de padrões estatísticos e aprendizado de máquina (SHAWE-TAYLOR; TRIGGS, 2002).

### 5.2.3 Separação

Após a conversão os dados ainda passam por mais uma fase, onde os dados são separados, um processo necessário para o treinamento e avaliação de modelos de aprendizado de máquina. As três subdivisões que serão usadas nesse projeto são os dados de treinamento, de validação e de teste.

- Os dados de treinamento são os que serão efetivamente utilizados para o aprendizado da rede neural em si, geralmente é nessa subdivisão que fica a maior parte dos dados.
- Os dados de validação são usados durante o treinamento para rapidamente verificar se realmente está acontecendo um aprendizado útil dos dados passados para o modelo, pois esses dados não são usados para aprendizado da arquitetura sendo treinada. Se a precisão do modelo subir nos dados de treinamento e nos de validação o treinamento está sendo efetivo, porém se a precisão apenas subir nos dados de treinamento e estagnar ou ser inconsistente nos dados de validação, isso indica algum problema no treinamento - mais comumente *overfit*<sup>2</sup>.
- Os dados de teste são também não são usados para o aprendizado - apenas após o treinamento e validação do modelo que eles são usados para avaliar a verdadeira efetividade do modelo.

Não existem regras fixas que indiquem qual deve ser o tamanho de cada subdivisão, geralmente isso é muito dependente do tamanho da base de dados obtida - o quanto maior for a base de dados, menores podem ser as subdivisões de validação e teste. Com a base de dados relativamente limitada desse projeto, a divisão será feita com 40 imagens para treinamento, 7 para validação e 12 para teste.

## 5.3 Implementação

O projeto inteiro foi implementado na linguagem *python* devido a amplitude de bibliotecas e ferramentas para ciência de dados já existentes na linguagem, que serão utilizadas extensivamente nesse projeto, com essas bibliotecas e suas respectivas versões sendo:

- *Tensorflow* - Versão 2.12: Biblioteca livre para desenvolvimento de aplicações de aprendizado de máquina e inteligência artificial, com foco particular em *deep learning*.
- *Keras* - Versão 2.12: Biblioteca escrita em *python* que disponibiliza uma interface para criação de RNAs, com foco principal em permitir fácil e rápida implementação de modelos

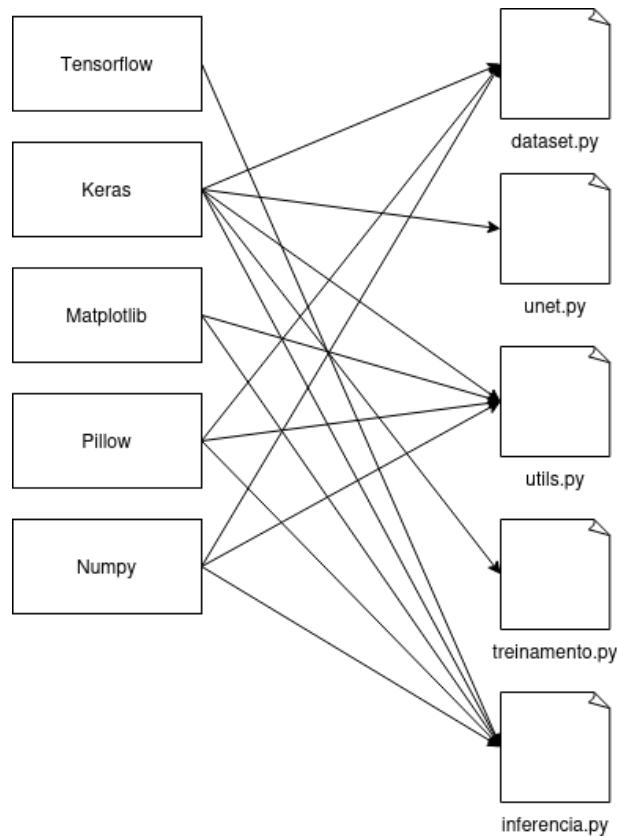
<sup>2</sup> Quando o modelo treinado se adapta apenas aos dados de treinamento e não consegue generalizar para respostas em outros dados.

de aprendizado de máquina. Utiliza as funções da biblioteca *tensorflow* como base para construção dos modelos.

- *Numpy* - Versão 1.23.5: Biblioteca científica para números e *arrays* em *python* comumente usada em ciência de dados.
- *Pillow* - Versão 9.5.0: Também chamada de PIL (de *Python Imaging Library*) é uma biblioteca de processamento e manipulação de imagens.
- *Matplotlib* - Versão 3.0.1: Biblioteca para criar visualizações de dados em *python*, sendo usado nesse projeto para criação e visualização de gráficos e imagens.

Os principais elementos do projeto foram implementados cada um em cinco arquivos principais: *dataset.py*, *unet.py*, *utils.py*, *treinamento.py* e *inferencia.py*, com o uso de cada uma das bibliotecas sendo exemplificado na Figura 14. O código fonte de cada um dos arquivos está disponibilizado no capítulo A e suas respectivas funções serão explicadas a seguir.

Figura 14 – Diagrama de componentes das bibliotecas utilizadas em cada arquivo do projeto



Fonte: Do autor.

### 5.3.1 Dataset

O arquivo `dataset.py` é a classe que se responsabiliza por carregar dos dados nas variáveis que vão ser usadas pelo modelo - o carregamento e pré-processamento dos dados do formato VOC para arrays da biblioteca `numpy`. A classe em si é herança da classe `Sequence` da biblioteca `keras`, que é um dos formatos padronizados de `dataset` da mesma biblioteca, que facilita a implementação e carregamento dos dados para o treinamento.

Ao inicializar um objeto dessa classe podem ser passados 3 parâmetros:

- `caminho`: Parâmetro obrigatório que é uma *string* que indica onde está a base de dados, que já deve estar convertida no formato VOC.
- `tamanho_imagem`: Parâmetro obrigatório que recebe um *tuple* da linguagem python no formato (`altura, largura`) em que se é informado em que resolução devem ser convertidos os dados que serão usados para treinamento do modelo.
- `batch_size`: Parâmetro opcional vindo das noções de aprendizado de máquina em si, que define quantas amostras da base de dados que serão processadas antes do modelo ser atualizado com novos valores.

Devido ao fato dessa classe ser herança da classe `Sequence` da biblioteca `keras` ela também tem necessariamente que implementar dois métodos: `__getitem__` que é utilizado quando o modelo em treinamento quer receber um *batch* de dados, e o método `__len__`, que deve retornar o tamanho total da base de dados, também em *batches*. As imagens processadas e o parâmetro `batch_size` são usados nesses métodos.

### 5.3.2 UNet

O arquivo `unet.py` contém uma função principal, `UNet`, que retorna o modelo em si implementado seguindo os padrões da *Functional API* da biblioteca `keras` para implementação de modelos.

A única diferença entre essa implementação e a original descrita no artigo da arquitetura é que a entrada, saída e o tamanho das camadas tem maior flexibilidade - na definição original todas as camadas tem um tamanho específico, com largura e altura igual, para todas as camadas.

O fato da implementação ter essa maior flexibilidade no tamanho ocorreu por dois motivos: principalmente sendo devido ao formato das fotos da base de dados em si, que foram tiradas por celulares na vertical que resultam nas imagens sempre tendo a largura maior que a altura - manter essa razão na entrada do modelo seria ideal para reduzir a quantidade de informação perdida pela imagem no redimensionamento, que teria uma perda grande de informação vertical ao converter as imagens do formato base para um formato quadrado.

O outro motivo do modelo ter essa flexibilidade é também a possibilidade de testar com vários tamanhos de entrada diferentes, por exemplo pode ser descoberto que o modelo responde muito bem a valores de entrada maior ou o oposto pode ser a verdade - que o modelo obtenha bons resultados mesmo com modelos menores(e consequentemente mais rápidos).

A única limitação dessa abordagem são os valores que podem ser usados como entrada em si, cada camada de contração tem metade do tamanho da anterior enquanto as camadas de expansão tem o dobro - o problema ocorre na contração, onde não pode existir uma camada com tamanho em fração, como por exemplo 62.5, pois isso causa erro ao instanciar os *arrays* que representam a arquitetura.

Mas essa limitação é algo que se pode adaptar facilmente, apenas é necessário utilizar valores que podem ser divididos por 2 até quatro vezes sem retornar numero fracionário, como por exemplo 192, 256, 320, 384, 512 e todas as outras opções.

### 5.3.3 Utils

O arquivo `utils.py` é composto apenas por funções que são utilizadas pelos outros arquivos da implementação. Todas essas funções se responsabilizam por 3 funcionalidades principais:

- Pré-processamento de imagens: Converter a imagem do formato de arquivo que está no disco rígido em *arrays numpy*. Incluindo redimensionamento, para converter a imagem do tamanho original para o tamanho que pode ser utilizado pelo modelo e vice versa e normalização, que reduz a variação de dados modificando os valores RGB das imagens para que eles fiquem entre -1 e 1 ao invés de 0 e 255.
- Visualização: Serve para mostrar as imagens, anotações e predições da base de dados e modelo visualmente com a biblioteca *matplotlib* mesmo depois do carregamento em variáveis e as várias modificações que ocorrem durante a execução dos arquivos.
- Avaliação: A função que calcula o *Intersection over Union* com a anotação original e a predição do modelo também está contida nesse arquivo.

### 5.3.4 Treinamento

O arquivo `treinamento.py` contém todos os elementos necessários para treinar o modelo e salvar ele localmente no formato *SavedModel* da biblioteca *tensorflow* para uso posterior, seja para continuação do treinamento ou para inferência.

No início do código é definida quais variáveis serão usadas para o treinamento, que no caso desse projeto são os parâmetros do treinamento de aprendizado de máquina que vai ocorrer. Esses parâmetros são:

- EPOCHS: A quantidade de épocas de treinamento, isso é, a quantidade de vezes que o modelo vai iterar sobre todas as *batches* da base de dados. Para esse projeto as quantidades de épocas que foram selecionadas foram 25, 50 e 100.
- INPUT\_SIZE: Identifica para qual resolução será redimensionada a imagem sendo passada para o modelo, além de também definir quais são os tamanhos das camadas da arquitetura e a saída da mesma devido ao que foi explicado na seção 5.3.2.

Para esse projeto foram utilizadas 3 variações de resolução de entrada: (208, 160), (416, 320) e (544, 416)<sup>3</sup>. Inicialmente o terceiro e maior valor da resolução que seria tentada era maior que esse, porém devido a limitações de *hardware*, mais especificamente a falta de memória de vídeo, modelos maiores que (544, 416) não puderam ser treinados.

- BATCH\_SIZE: Variável que é passada para o *dataset* para indicar qual o tamanho da *batch*, ou agrupamento de dados, que o modelo deve analisar antes de atualizar os seus pesos. Os valores testados nesse projeto serão 1, 2 e 4.
- LEARNING\_RATE: Taxa de aprendizado do treinamento. Inicialmente esse projeto também treinaria esses modelos com valores variados para a taxa de aprendizado, mas devido ao algoritmo de otimização selecionado, que automaticamente adapta esse valor durante o treinamento, os valores testados não demonstraram nenhuma diferença significativa. O valor utilizado no final foi sempre, inicialmente,  $5 \cdot 10^{-6}$ .

Após a definição desses parâmetros, o arquivo segue com o carregamento do modelo vindo do arquivo *unet.py* e de dois objetos de base de dados, o *dataset\_treinamento* e o *dataset\_validacao* como foram explicados na seção 5.3.1, seguido pela compilação do modelo carregado, que é uma etapa necessária para o treinamento de modelos implementados na biblioteca *keras*, onde é definido o algoritmo de otimização e o algoritmo de perda/*loss*.

O algoritmo de otimização selecionado para esse projeto foi o *Adam*, um algoritmo baseado no *stochastic gradient descent* que deriva seu nome de *adaptive moment estimation*, a habilidade de adaptar o valor a sua taxa de aprendizado baseado durante o treinamento (KINGMA; BA, 2014). Esse algoritmo foi selecionado devido aos seus bons resultados e a capacidade de ser usado em qualquer projeto que não precise de algum algoritmo em específico.

Na contraparte, o algoritmo de perda, que se responsabiliza por calcular o quanto longe a resposta do modelo ficou da resposta real, é escolhido dependendo de como os dados estão preparados para o treinamento. No caso desse projeto e segmentação de imagens em geral, o algoritmo escolhido foi *sparse categorical crossentropy*.

Após o treinamento com todos os parâmetros acima definidos, o modelo é então salvo no formato *SavedModel* para utilização futura, no caso desse projeto - para avaliação.

<sup>3</sup> Essas resoluções estão no formato (*altura, largura*).

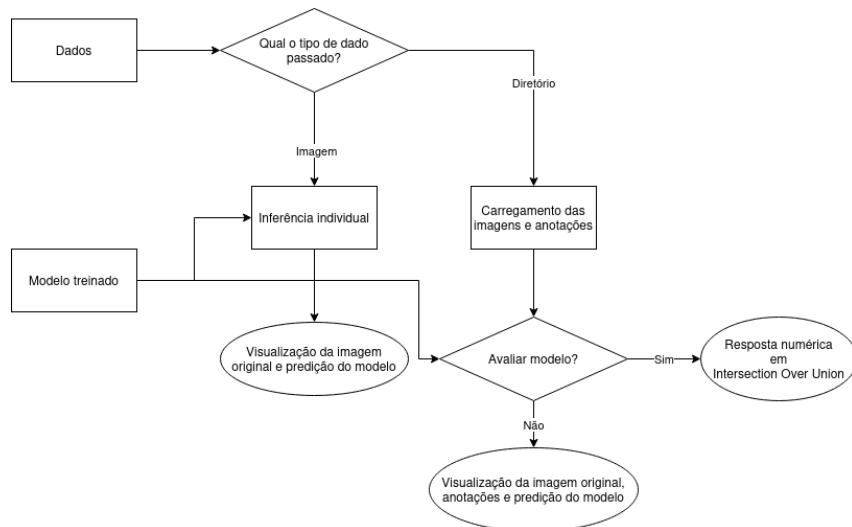
### 5.3.5 Inferência

O último arquivo do projeto é `inferencia.py`, que utiliza um modelo salvo no sistema para gerar respostas, sejam individualmente ou para uma base de dados anotada, além de também avaliar com IoU a capacidade de segmentação do modelo.

No início do código duas variáveis são definidas: `caminho_modelo`, que indica onde está a pasta com um modelo carregável pela biblioteca `keras` e `caminho_dados`, que indica onde está os dados que o modelo deve realizar predição.

Os dados que são referenciados por essa segunda variável podem estar em 2 formatos diferentes que modificam o funcionamento dependendo do escolhido, com a primeira opção sendo diretamente um arquivo de imagem, em que o modelo vai retornar uma visualização da imagem e a predição do modelo, isso é, a máscara de segmentação onde ele acredita que estão as células em mitose, com esses fluxos representados na Figura 15.

Figura 15 – Visualização do fluxo de funcionamento arquivo `inferencia.py`



Fonte: Do autor.

Caso a variável dos dados apontem para um diretório, o arquivo carrega as imagens e anotações esperando que eles estejam organizados no formato VOC e usa elas para gerar as predições do modelo, que são usadas de maneiras diferentes dependendo da última variável do arquivo, `avaliar`.

Se essa variável tiver valor `True`, é retornado o valor calculado de *Intersection over Union* com a máscara verdadeira e a da predição do modelo. Porém, se for falsa, vai retornar visualizações da imagem, máscara real e máscara de predição, como representado na 16.

Durante o treinamento, após a observação dos primeiros resultados, também foi implementada mais uma variável: `mostrar_camada_resposta`, que altera a visualização da camada

de segmentação retornada pelo modelo para especificamente a camada das mitoses. A motivação de implementar essa visualização alternativa é explicada na seção 6.2.

## 5.4 Treinamento

Os treinamentos foram executados em um computador com as seguintes especificações de *hardware e software*:

- Placa de vídeo: NVIDIA GeForce RTX 2060 com 6GB de memória de vídeo
- Processador: Ryzen 3 2200G
- Memória: 16GB DDR4
- Sistema Operacional: Debian Sid

Devido a natureza aleatória da inicialização dos pesos da arquitetura escolhida, é comum a execução de múltiplos treinamentos com o resultado final sendo uma média aritmética de todos os resultados obtidos com os mesmos parâmetros de treinamento, como por exemplo a média resultante de 10 treinamentos.

Porém, executar dez treinamentos com todas as combinações de parâmetros possíveis escolhidas seria um uso ineficiente do tempo devido ao quanto que demoraria para concluir todos esses treinamentos. Para reduzir a quantidade de tempo que seria gasta executando e avaliando, serão executados primeiramente uma série de treinamentos menos organizados, com o intuito de definir valores para parâmetros de maior interesse.

Nesses treinamentos iniciais diferentes combinações de parâmetros serão treinadas um total de 3 vezes com o intuito de encontrar quais variações mais consistentemente trazem melhores resultados, por exemplo se `BATCH_SIZE` maior ou menor retorna os melhores valores de `IoU` em média, ou se há uma grande perda ou ganho de precisão dependendo da resolução selecionada.

Os resultados desses treinamentos foram anotados em uma planilha, com os 3 resultados de *IoU* para cada combinação que será então analisada em busca de padrões que revelam quais parâmetros trazem melhores resultados mais consistentemente.

Esses treinamentos iniciais também já devem revelar a eficiência geral da arquitetura treinada na base de dados, caso já sejam retornados resultados aceitáveis, mesmo apenas com combinações de parâmetros específicas, os treinamentos e avaliações que ocorrerão após isso apenas servirão para encontrar o treinamento mais otimizado, isso é, que traz os melhores resultados para as condições atuais.

Caso nenhuma variação inicial do modelo treinado retornar resultados pelo menos próximos de satisfatórios, ocorrerá uma investigação mais profunda dos resultados obtidos para tentar compreender o que pode ter dado errado no aprendizado do modelo.

## 5.5 Avaliação

No caso de sucesso em segmentação do modelo, ocorrerá o treinamento com essa variedade de parâmetros já mais afunilada, porém com a média de IoU sendo calculada entre os 10 treinamentos, com tabelas comparativas demonstrando numericamente a efetividade dos modelos com melhores resultados.

Um detalhe sobre o valor de IoU do modelo é que o mesmo está sendo treinado apenas com parte das mitoses que realmente tem na imagem, as que foram identificadas visualmente pelos patologistas que disponibilizaram a base de dados. Porém, um modelo treinado que consiga distinguir as características específicas das células em mitose das outras células também conseguiria identificar mitoses que o patologista pode ter deixado passar.

O que isso quer dizer é que o modelo ideal não terá o valor de IoU próximo do valor perfeito, que seria 1, pois as camadas de segmentação se distinguiriam: a camada de predição retornada pelo modelo teria mais células que a camada de anotações da imagem original.

Nesse caso, com o modelo retornando células a mais do que foi anotado na mesma imagem, essas respostas serão repassadas para um patologista para validação, que retornará para a gente se essas células extras realmente estão em mitose ou se é um falso positivo.

# 6 RESULTADOS E DISCUSSÕES

## 6.1 Resultados iniciais

Durante a observação dos resultados dos primeiros treinamentos já era perceptível que a arquitetura não estava conseguindo segmentar corretamente as células em mitose, por mais que tenha sido identificado um padrão nos parâmetros de treinamento - que modelos com a quantidade de EPOCHS em 100 geralmente retornava melhores resultados<sup>1</sup>, o valor de IoU retornado sempre era extremamente baixo, com uma parte dos casos sendo exatamente 0, como é demonstrado na Tabela 3.

EPOCHS	INPUT_SIZE	BATCH_SIZE	Média de IoU - 3 treinamentos
25	(416, 320)	1	0.004878622912
25	(416, 320)	4	0
50	(208, 160)	1	0
50	(208, 160)	4	0.0000452084
50	(416, 320)	2	0.00288041802
50	(544, 416)	1	0.00019799962
50	(544, 416)	2	0.019099800599
100	(208, 160)	1	0.00040438301
100	(416, 320)	4	0.016535506375

Tabela 3 – Amostras aleatórias de resultados de IoU por combinação de parâmetros tentados

Para contraste, o valor de IoU retornado sendo exatamente 1 indica que a segmentação real e da predição foram exatamente iguais, com valores próximos a isso indicando modelos extremamente efetivos. Como os valores obtidos são apenas frações dos valores que seriam retornados por um modelo eficaz, é necessário executar uma investigação para entender o que está sendo segmentado pelo modelo.

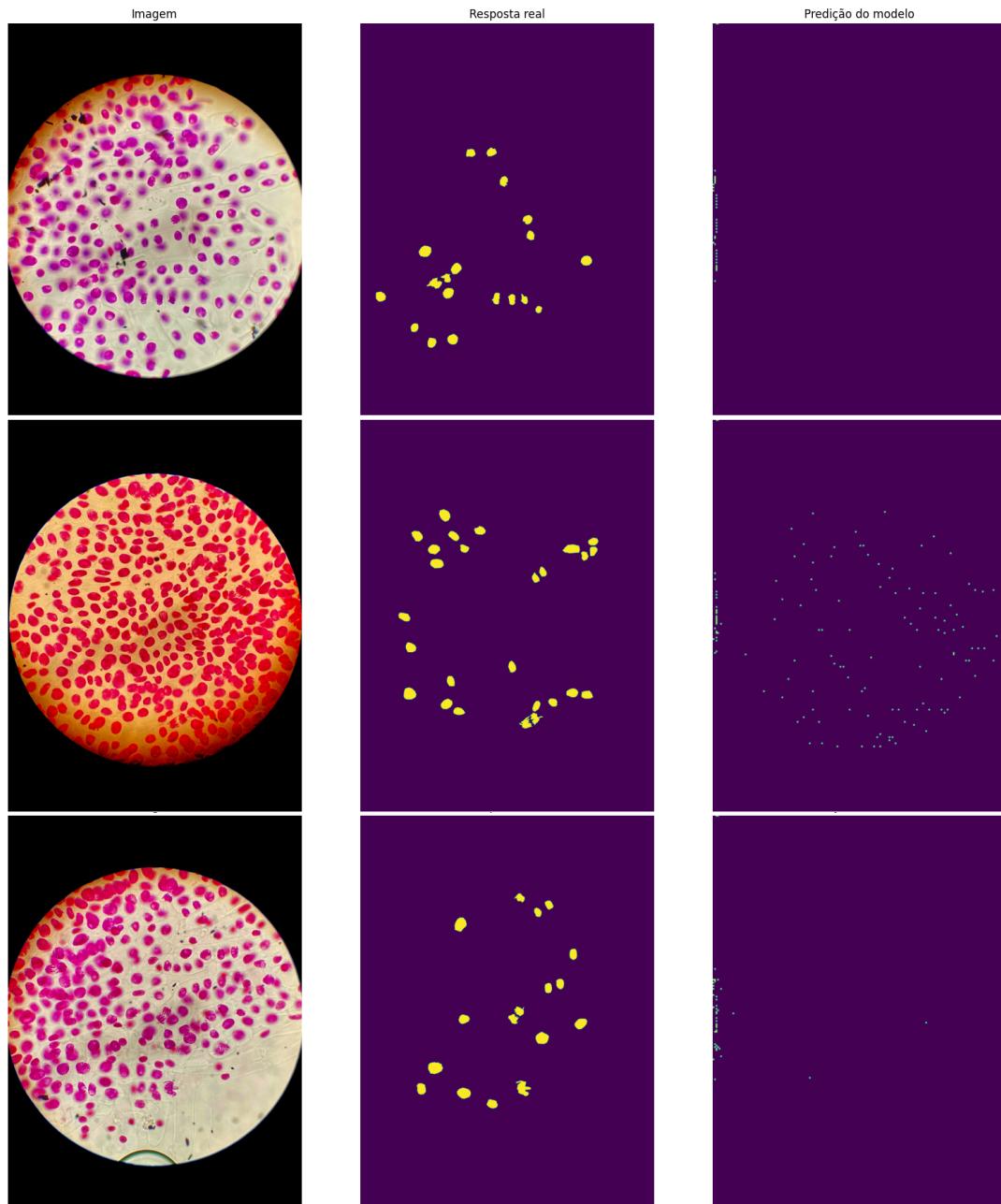
## 6.2 Investigação

*Intersection over Union* é uma maneira matemática de avaliar resultados de segmentação, valores próximos ou iguais a 1 indicam que o resultado sendo obtido é ideal, porém valores menores que isso não contém nenhum esclarecimento sobre o que está sendo retornado errado. Para poder entender o que está acontecendo é preciso de uma análise visual, comparando a segmentação real com a retornada pelo modelo treinado.

<sup>1</sup> Também foram realizados alguns testes com valores maiores que 100, mas a partir desse valor não houve nenhuma diferença perceptível nos resultados.

Após analisar visualmente as respostas dos modelos que obtiveram melhores resultados nas imagens de teste, são encontrados dois padrões de resposta retornada: imagens em que apenas algumas bordas e cantos são identificadas como mitose, e imagens em que apenas alguns *pixels* na região das células são identificadas como mitose, com alguns exemplos sendo representados na Figura 16.

Figura 16 – Visualização das camadas de segmentação retornadas pelo modelo



Fonte: Do autor.

Essa visualização esclarece apenas alguns detalhes, nas imagens em que se tem vários *pixels* dispersos eles geralmente se encontram na região da imagem onde realmente estão os

núcleos celulares, o que pelo menos indica que o modelo aprendeu algo, mas não explica por quê que a imagem quase inteira é classificada como classe fundo.

Para entender melhor o que está acontecendo é necessário destrinchar mais as respostas do modelo, mas antes disso é preciso explicar como que essa resposta é obtida e processada para se tornar a imagem da predição.

O modelo de segmentação que foi implementado e os que seguem os mesmos padrões retornam na saída uma camada para cada classe de objeto que está sendo segmentada, sempre existindo no mínimo duas classes: uma sendo a classe de fundo ou *background*, que indica o que o modelo deve aprender como irrelevante, ou explicando de maneira alternativa, quais são as características do que não é as classes que estão sendo segmentadas.

Para exemplificar, cada tipo de objeto que está sendo segmentada também vira sua própria classe, um modelo de segmentação que identifica gatos e cachorros teria 3 classes no total: *background*, gato e cachorro, e no caso desse projeto existe apenas uma classe além do fundo, a classe das mitoses, o que quer dizer que a resposta do modelo retorna duas camadas de segmentação.

A imagem das predições resultantes é uma junção dessas duas camadas de segmentação, que é a camada resultante após passar a resposta original pelo método da biblioteca *tensorflow* `tf.math.argmax`, onde apenas o índice do maior valor retornado na camada resposta. O que isso quer dizer é que, as partes da predição que estão roxas é onde o modelo tem mais confiança que é fundo, enquanto as amarelas são onde há mais confiança de ser mitose.

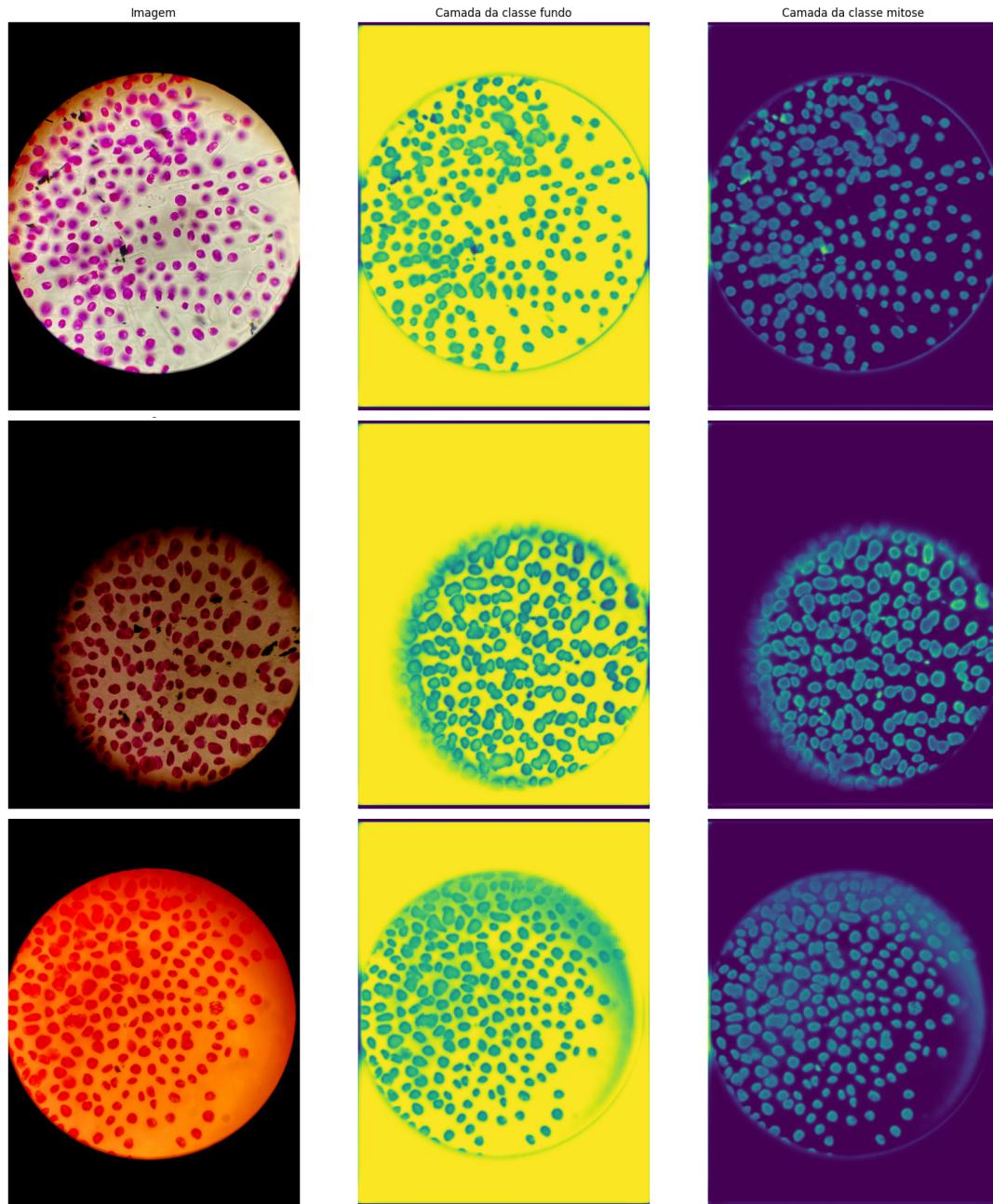
O objetivo de explicar isso é que ainda existe mais uma versão da resposta que pode ser analisada, as camadas individuais, que podem esclarecer significativamente o que o modelo realmente aprendeu. Essas camadas são compostas por um valor de confiança para cada *pixel* da imagem original que indica o quanto que o modelo acredita que esse *pixel* pertence a aquela classe.

A Figura 17 demonstra essas camadas individuais, exatamente o que o modelo está prevendo. Na camada de fundo é possível observar que ele corretamente identifica o que é fundo, acertando que a parte preta da imagem e o fundo da parte que contém as células, que não contém núcleos celulares, com valores amarelos mais fortes que indicam altos valores de confiança.

Porém ainda na camada de fundo dá para se identificar que os segmentos que contém os núcleos celulares não estão exatamente na cor roxa, e sim um meio-termo entre os dois, indicando que o modelo não aprendeu as características necessárias para distinguir essas células de fundo.

Na camada individual das mitoses é observado que são retornados valores baixos de confiança em todas as células da imagem, sem valores de confiança maiores em células específicas, algo que poderia indicar que o modelo acha que é mitose. Frequentemente partes das imagens com coloração mais saturada e próximas da borda da imagem também retornam valores altos de

Figura 17 – Visualização das camadas individuais para cada classe do modelo, com a imagem original na esquerda



Fonte: Do autor.

confiança.

### 6.3 Discussões

Devido a natureza de modelos de aprendizado profundo não é possível apontar para uma variável ou peso específico treinado que resulta na falha, porém com o conhecimento do seu

funcionamento e método de aprendizado é possível extrair algumas conclusões desses resultados apresentados, mais especificamente das camadas individuais.

Existem 2 problemas principais na segmentação: O modelo treinado não tem confiança o suficiente para distinguir uma mitose de fundo, e o que o modelo considera como classe mitose na verdade são todos os núcleos celulares da imagem original.

O fato do modelo não conseguir distinguir mitose de fundo pode ser explicado pelas anotações da base de dados original, como é mencionado na seção 5.1, apenas em torno de 40% das células originais são classificadas em alguma fase do ciclo de vida, e dessa porção, apenas em torno de 25% a 35% estão em uma fase de divisão, um padrão seguido por quase todas as imagens fornecidas.

Com apenas uma fração de uma porção das frações sendo anotadas para o aprendizado, o modelo tem bastante informação para aprender o que é a classe fundo, mas equivalentemente pouca informação para a classe de segmentação. O que não seria um problema se os dois fossem bastante distintos, porém devido as imagens não serem de alta qualidade, as distinções que existem são menos identificáveis.

Juntando isso com o fato de que nem todas as células em mitose da imagem estão anotadas, é possível que o modelo esteja aprendendo as características exclusivas da mitose nas células anotadas, mas também esteja encontrando as mesmas características em uma célula que os patologistas não anotaram e por isso está sendo considerada na classe de fundo.

Outra possível causa de problemas é uma pequena falta de padronização na base de dados, como pode ser visto nas figuras com exemplos de imagens da base de dados ao longo desse projeto, as imagens contêm diferenças de coloração - isso é devido a certos líquidos que são aplicados aos tecidos analisados além do colorante para os núcleos celulares pelos patologistas, devido a natureza do projeto de onde vêm esses dados.

Em geral, considerando que a arquitetura de segmentação selecionada já obteve sucesso com problemas similares, que a implementação utilizou ferramentas adequadas para o problema, e que o treinamento foi executado de maneira que deveria ter obtido respostas se fosse possível, a conclusão é que o principal responsável pela falha do projeto vem da base de dados ou das anotações.

Como mencionado na seção 5.2.1 esse padrão de anotação para segmentar células em mitose em que apenas as mitoses em si são anotadas já obteve resultados em outras bases de dados, porém é possível que outro formato de anotação seja mais adequado para essa base de dados, como por exemplo, com uma classe para células anotadas em interfase, uma para células em mitose e outra para células não identificáveis.

## 7 CONCLUSÃO

Este trabalho teve como propósito principal a implementação de um modelo preditivo baseado em aprendizado de máquina para segmentação de células mitóticas para o auxílio de estudantes de patologia.

Apesar desse tipo da segmentação de células mitóticas já ser algo que teve resultados comprovados em muitos estudos como é exemplificado na revisão sistemática desse projeto, estudantes de patologia trabalham com células diferentes, derivadas de cebolas, ou *allium cepa*, onde não foi encontrado nenhum estudo com essa proposta.

Para conseguir as melhores chances de sucesso possível, a revisão sistemática foi executada não só com o objetivo de aprender sobre o estado da arte da segmentação de imagens médicas, mas também para observar qual arquitetura já existente era mais frequentemente usada para obter bons resultados, com a selecionada sendo a arquitetura *U-Net*.

Após o término de todas as etapas da metodologia foi observado logo nos primeiros treinamentos que o modelo não estava obtendo resultados satisfatórios. Foi executada uma investigação em que as respostas do modelo foram analisadas profundamente para entender o que estava ocorrendo de errado, e foi concluído que o modelo não estava conseguindo distinguir as células em mitose das outras células.

Essa falha mais provavelmente foi causada pela base de dados ou a maneira que a anotação dos dados foi feita, o que quer dizer que ainda é possível que a ideia do projeto possa ter êxito, mas que precisaria de uma base de dados melhor ou no mínimo algumas diferenças na abordagem.

## 8 TRABALHOS FUTUROS

Mesmo com a falha desse projeto ainda é possível que a ideia de treinar um modelo de segmentação para auxiliar estudantes de patologia funcione, evoluindo desse projeto em si existem algumas possibilidades que podem resultar em um modelo funcional.

A mais simples é uma base de dados melhor, tanto em tamanho quanto possivelmente com menos variação, isso é, sem nenhum dos líquidos que alteram a cor adicionados, resultem em um melhor aprendizado. Outra possibilidade também é realizar as anotações de uma maneira diferente, pois além da sugestão na seção 6.3 os integrantes do projeto de medicina ligado a esse também expressaram interesse em um modelo que consiga classificar em qual fase da divisão a célula está.

# REFERÊNCIAS

- AKHTAR, U. et al. Improving prostate cancer detection with breast histopathology images. 2019. Citado 3 vezes nas páginas 13, 14 e 19.
- ALOM, M. Z. et al. Mitosisnet: End-to-end mitotic cell detection by multi-task learning. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 68695–68710, 2020. ISSN 21693536. Citado 2 vezes nas páginas 19 e 21.
- BENAGGOUNE, K. et al. A deep learning pipeline for breast cancer ki-67 proliferation index scoring. 2022. Citado na página 21.
- CAYIR, S. et al. Mitnet: a novel dataset and a two-stage deep learning approach for mitosis recognition in whole slide images of breast cancer tissue. *Neural Computing and Applications*, Springer Science and Business Media Deutschland GmbH, v. 34, p. 17837–17851, 10 2022. ISSN 14333058. Citado na página 21.
- DEWANGAN, K. K. et al. Breast cancer diagnosis in an early stage using novel deep learning with hybrid optimization technique. *Multimedia Tools and Applications*, Springer, v. 81, p. 13935–13960, 4 2022. ISSN 15737721. Citado 2 vezes nas páginas 13 e 20.
- EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, v. 88, p. 303–338, 6 2010. ISSN 09205691. Citado na página 35.
- FERREIRA, A. dos S. Redes neurais convolucionais profundas na detecção de plantas daninhas em lavoura de soja. 2017. Citado 5 vezes nas páginas 22, 23, 24, 25 e 26.
- GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. 11 2013. Disponível em: <<http://arxiv.org/abs/1311.2524>>. Citado na página 27.
- HAGOS, Y. B.; MERIDA, A. G.; TEUWEN, J. Improving breast cancer detection using symmetry information with deep learning. 2018. Citado 2 vezes nas páginas 14 e 18.
- HIRRA, I. et al. Breast cancer classification from histopathological images using patch-based deep learning modeling. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 9, p. 24273–24287, 2021. ISSN 21693536. Citado na página 20.
- HOUSSEIN, E. H.; EMAM, M. M.; ALI, A. A. An optimized deep learning architecture for breast cancer diagnosis based on improved marine predators algorithm. *Neural Computing and Applications*, Springer Science and Business Media Deutschland GmbH, 2022. ISSN 14333058. Citado 2 vezes nas páginas 13 e 20.
- INCA. *Câncer de mama — gov.br*. 2022. <<https://www.gov.br/inca/pt-br/assuntos/cancer/tipos/mama>>. Acessado em 27 dez. 2022. Citado na página 13.
- IQBAL, S.; QURESHI, A. N. A heteromorphous deep cnn framework for medical image segmentation using local binary pattern. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 10, p. 63466–63480, 2022. ISSN 21693536. Citado 2 vezes nas páginas 20 e 21.

- KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. 12 2014. Disponível em: <<http://arxiv.org/abs/1412.6980>>. Citado na página 40.
- KUMAR, V. *Hands-on guide to object detection using YOLO*. 2020. Acessado em 22 jan. 2022. Disponível em: <<https://analyticsindiamag.com/hands-on-guide-to-object-detection-using-yolo/>>. Citado na página 28.
- LECUN, Y.; BOSEN, B.; DENKER, J. S. Backpropagation applied to handwritten zip code recognition. 1989. Citado 2 vezes nas páginas 24 e 27.
- LECUN, Y. et al. Gradient-based learning applied to document recognition. 1998. Citado na página 24.
- LEFKOWITZ, M. *Professor's Perceptron paved the way for AI – 60 years too soon*. 2019. Disponível em: <<https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon>>. Citado na página 22.
- LI, S. et al. Attention dense-u-net for automatic breast mass segmentation in digital mammogram. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 7, p. 59037–59047, 2019. ISSN 21693536. Citado 2 vezes nas páginas 18 e 21.
- LI, Y.; WU, J.; WU, Q. Classification of breast cancer histology images using multi-size and discriminative patches based on deep learning. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 7, p. 21400–21408, 2019. ISSN 21693536. Citado na página 18.
- LI, Z. et al. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, Institute of Electrical and Electronics Engineers (IEEE), p. 1–21, 6 2021. ISSN 2162-237X. Citado 2 vezes nas páginas 23 e 25.
- LIU, P. *Single Stage Instance Segmentation - A Review*. 2020. Acessado em 22 jan. 2022. Disponível em: <<https://towardsdatascience.com/single-stage-instance-segmentation-a-review-1eeb66e0cc49>>. Citado na página 28.
- MAHMOOD, T. et al. Artificial intelligence-based mitosis detection in breast cancer histopathology images using faster r-cnn and deep cnns. *Journal of Clinical Medicine*, MDPI, v. 9, 3 2020. ISSN 20770383. Citado na página 34.
- MAN, R.; YANG, P.; XU, B. Classification of breast cancer histopathological images using discriminative patches screened by generative adversarial networks. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 155362–155377, 2020. ISSN 21693536. Citado 2 vezes nas páginas 13 e 19.
- MASUD, M.; RASHED, A. E. E.; HOSSAIN, M. S. Convolutional neural network-based models for diagnosis of breast cancer. *Neural Computing and Applications*, Springer Science and Business Media Deutschland GmbH, v. 34, p. 11383–11394, 7 2022. ISSN 14333058. Citado na página 14.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *BULLETIN OF MATHEMATICAL BIOPHYSICS*, v. 5, 1943. Citado na página 22.

- NEFIC, H. et al. Chromosomal and nuclear alterations in root tip cells of allium cepa l. induced by alprazolam. *Medical archives (Sarajevo, Bosnia and Herzegovina)*, v. 67, p. 388–392, 12 2013. ISSN 0350199X. Citado 2 vezes nas páginas 13 e 14.
- O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. 11 2015. Disponível em: <<http://arxiv.org/abs/1511.08458>>. Citado 4 vezes nas páginas 23, 24, 25 e 26.
- O'CONNOR, C. *Mitosis and Cell Division*. Nature Publishing Group, 2008. Disponível em: <<https://www.nature.com/scitable/topicpage/mitosis-and-cell-division-205/>>. Citado na página 33.
- PRAMODITHA, R. *The concept of artificial neurons (perceptrons) in neural networks*. Towards Data Science, 2021. Disponível em: <<https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc>>. Citado na página 22.
- RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. 5 2015. Disponível em: <<http://arxiv.org/abs/1505.04597>>. Citado 3 vezes nas páginas 27, 29 e 30.
- ROSENBLATT, F. The perceptron—a perceiving and recognizing automaton. 1 1957. Citado na página 22.
- SEBAI, M.; WANG, T.; AL-FADHLI, S. A. Partmitosis: A partially supervised deep learning framework for mitosis detection in breast cancer histopathology images. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 45133–45147, 2020. ISSN 21693536. Citado 2 vezes nas páginas 19 e 34.
- SHawe-Taylor, J.; Triggs, B. Pattern analysis, statistical modelling and computational learning (pascal) \*. 2002. Citado na página 35.
- SOHAIL, A. et al. A multi-phase deep cnn based mitosis detection framework for breast cancer histopathological images. *Scientific Reports*, Nature Research, v. 11, 12 2021. ISSN 20452322. Citado na página 13.
- SOUZA, H. E. Classificação de lesões em imagens histológicas de mama usando wavelet e resnet-50. 2021. Citado na página 25.
- SULTANA, F.; SUFIAN, A.; DUTTA, P. Evolution of image segmentation using deep convolutional neural network: A survey. 1 2020. Disponível em: <<http://arxiv.org/abs/2001.04074>> <<http://dx.doi.org/10.1016/j.knosys.2020.106062>>. Citado na página 28.
- ZHENG, J. et al. Deep learning assisted efficient adaboost algorithm for breast cancer detection and early diagnosis. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 96946–96954, 2020. ISSN 21693536. Citado na página 19.
- ZHU, Z. et al. Deep learning for identifying radiogenomic associations in breast cancer. 2017. Citado na página 18.

# A APÊNDICE

## A.1 dataset.py

```

import PIL
import keras
import os
import numpy as np
import utils
import math

# Dataset carregado de uma pasta com imagens e anotações no formato VOC

# Herda da classe sequence que serve para organizar sequências de dados para o keras
# Obrigatoriamente tem que conter os métodos __getitem__ e __len__
class Dataset(keras.utils.Sequence):
    def __init__(self, caminho, tamanho_imagem, batch_size = 1):
        self.tamanho_imagem = tamanho_imagem
        self.batch_size = batch_size

        self.num_classes = self.num_classes(caminho)
        self.imagens, self.anotacoes = self.carregar_dados(caminho)

    # Carrega os arquivos no dataset, com imagens e anotações no formato VOC
    def carregar_dados(self, caminho):
        imagens = []
        anotacoes = []
        nomes_imagens = os.listdir(caminho + "/JPEGImages")

        for nome_imagem in nomes_imagens:
            nome_imagem = nome_imagem.split(".")[0]

            imagem = utils.processar_imagem(caminho + "/JPEGImages/" + nome_imagem + ".jpg",
                                             self.tamanho_imagem)
            anotacao = utils.processar_anotacao(caminho + "/SegmentationClassPNG/" +
                                              nome_imagem + ".png", self.tamanho_imagem)

            imagens.append(imagem)
            anotacoes.append(anotacao)

        return imagens, anotacoes

    # Define a quantidade de classes baseado no arquivo class_names.txt
    def num_classes(self, caminho):
        with open(caminho + 'class_names.txt') as reader:
            return len(reader.readlines())

```

```
# Método obrigatório da superclasse Sequence, retorna uma batch de dados durante o
# treinamento
def __getitem__(self, index):
    imagens = []
    anotacoes = []

    for i in range(index * self.batch_size, (index + 1) * self.batch_size):
        imagens.append(self.imagens[i])
        anotacoes.append(self.anotacoes[i])

    imagens = np.array(imagens)
    anotacoes = np.array(anotacoes)

    return imagens, anotacoes

# Método obrigatório da superclasse Sequence, retorna o tamanho total da base de dados,
# considerando o batch_size
def __len__(self):
    return math.ceil(len(self.imagens) / self.batch_size)
```

## A.2 unet.py

```

import keras
# Função que retorna o modelo U-Net implementado na biblioteca keras
def UNet(tamanho_entrada, num_classes):
    entrada = keras.layers.Input(shape=tamanho_entrada + (3,)) # Tamanho da imagem de entrada
    ↪ e 3 camadas de cores RGB

    # Contração
    caracteristicas1, contracao1 = bloco_contracao(entrada, 64)
    caracteristicas2, contracao2 = bloco_contracao(contracao1, 128)
    caracteristicas3, contracao3 = bloco_contracao(contracao2, 256)
    caracteristicas4, contracao4 = bloco_contracao(contracao3, 512)

    # Camadas intermediárias/continuação
    continuacao = convolucao_dupla(contracao4, 1024)

    # Expansão
    expansao1 = bloco_expansao(continuacao, caracteristicas4, 512)
    expansao2 = bloco_expansao(expansao1, caracteristicas3, 256)
    expansao3 = bloco_expansao(expansao2, caracteristicas2, 128)
    expansao4 = bloco_expansao(expansao3, caracteristicas1, 64)

    saida = keras.layers.Conv2D(num_classes, 1, padding="same", activation =
    ↪ "softmax")(expansao4)

    # Modelo completo
    unet_model = keras.Model(entrada, saida, name="UNet")

    return unet_model

# Camadas de convolução usados na contração
def convolucao_dupla(camadas, filtros):
    camadas = keras.layers.Conv2D(filtros, 3, padding = "same", activation = "relu",
    ↪ kernel_initializer = "he_normal")(camadas)
    camadas = keras.layers.Conv2D(filtros, 3, padding = "same", activation = "relu",
    ↪ kernel_initializer = "he_normal")(camadas)
    return camadas

# Blocos de convolução completos, com as camadas, pooling e dropout
def bloco_contracao(camadas, filtros):
    camadas_caracteristicas = convolucao_dupla(camadas, filtros)
    camadas = keras.layers.MaxPool2D(2)(camadas_caracteristicas)    # MaxPool reduz o tamanho
    ↪ pela metade
    camadas = keras.layers.Dropout(0.3)(camadas)    # Dropout para evitar overfit
    return camadas_caracteristicas, camadas

# Camadas usadas na expansão da imagem
def bloco_expansao(camadas, caracteristicas_anteriores, filtros):
    camadas = keras.layers.Conv2DTranspose(filtros, 3, 2, padding="same")(camadas)
    camadas = keras.layers.concatenate([camadas, caracteristicas_anteriores])
    camadas = keras.layers.Dropout(0.3)(camadas)
    camadas = convolucao_dupla(camadas, filtros)

    return camadas

```

### A.3 utils.py

```
import PIL
import numpy as np
import matplotlib.pyplot as plt
import keras
# -----
# Funções úteis para o projeto
# -----

# Pre-processamento para treinamento
def processar_imagem(caminho, tamanho):
    imagem = PIL.Image.open(caminho)
    imagem = converter_cores_rgb(imagem)
    if(tamanho != None):
        imagem = redimensionar_imagem(imagem, tamanho)
    imagem = np.array(imagem, np.float32)
    imagem = normalizar(imagem)

    return imagem

# Pre-processamento para treinamento
def processar_anotacao(caminho, tamanho):
    anotacao = PIL.Image.open(caminho)
    anotacao = PIL.Image.fromarray(np.array(anotacao))
    if(tamanho != None):
        anotacao = redimensionar_anotacao(anotacao, tamanho)
    anotacao = np.array(anotacao)

    return anotacao

def redimensionar_imagem(imagem, tamanho):
    imgLargura, imgAltura = imagem.size
    largura, altura = tamanho

    escala = min(largura / imgLargura, altura / imgAltura)
    novaLargura = int(imgLargura * escala)
    novaAltura = int(imgAltura * escala)

    imagem = imagem.resize((novaLargura, novaAltura), PIL.Image.BICUBIC)
    nova_imagem = PIL.Image.new('RGB', tamanho, (128, 128, 128))

    nova_imagem.paste(imagem, ((largura - novaLargura) // 2, (altura - novaAltura) // 2))

    return nova_imagem
```

```
def redimensionar_anotacao(anotacao, tamanho):
    anoLargura, anoAltura = anotacao.size
    largura, altura = tamanho

    escala = min(largura / anoLargura, altura / anoAltura)
    novaLargura = int(anoLargura * escala)
    novaAltura = int(anoAltura * escala)

    anotacao = anotacao.resize((novaLargura, novaAltura), PIL.Image.NEAREST)
    nova_anotacao = PIL.Image.new('L', tamanho, (0))

    nova_anotacao.paste(anotacao, ((largura - novaLargura) // 2, (altura - novaAltura) // 2))

    return nova_anotacao

def normalizar(imagem):
    imagem = imagem / 127.5 - 1
    return imagem

# Garante que a imagem tem 3 canais de cores
def converter_cores_rgb(image):
    if len(np.shape(image)) == 3 and np.shape(image)[-2] == 3:
        return image
    else: # Se não tiver, converter
        image = image.convert('RGB')
    return image

def iou_mascara_binaria(anotacao, predicao):
    # Área total positiva das máscaras
    anotacao_area = np.count_nonzero(anotacao == 1)
    predicao_area = np.count_nonzero(predicao == 1)

    # Área com valor positivo em ambas as máscaras
    intersecao = np.count_nonzero(np.logical_and(anotacao == 1, predicao == 1))

    iou = intersecao / (anotacao_area + predicao_area - intersecao)

    return iou

def visualizar(imagem, anotacao, predicao):
    plt.figure(figsize=(15, 15))

    titulo = ['Imagen', 'Resposta do modelo']

    # if len(anotacao) != 0:
    #     titulo.insert(1, 'Anotação')

    for i in range(len(titulo)):
        plt.subplot(1, len(titulo), i + 1)
        plt.title(titulo[i])
        if(i == 0):
            plt.imshow(imagem)
        if(i == 1):
            plt.imshow(anotacao)
        if(i == 2):
            plt.imshow(predicao)
        plt.axis('off')

    plt.show()
```

## A.4 treinamento.py

```
import keras
from dataset import Dataset
from unet import UNet
import utils
# -----
# Configurações de treinamento
# -----

# Epocas de treinamento
EPOCHS = 25

# Dimensões da imagem de entrada - altura e largura
INPUT_SIZE = (208, 160)
# INPUT_SIZE = (416, 320)
# INPUT_SIZE = (544, 416)

# Quantidade de imagens que o modelo deve analizar por epoca de treinamento
# isso é, antes de modificar os pesos do modelo
BATCH_SIZE = 1

# Taxa de aprendizado do otimizador
LEARNING_RATE = 0.000005

# Carregamento das bases de dados
dataset_treinamento = Dataset('datasets/dataset_treinamento_voc/' , INPUT_SIZE,
                             batch_size=BATCH_SIZE)
dataset_validacao = Dataset('datasets/dataset_validacao_voc/' , INPUT_SIZE, batch_size=1)

# Definição e compilação do modelo
unet = UNet(INPUT_SIZE, dataset_treinamento.num_classes)
unet.compile(
    optimizer = keras.optimizers.Adam(learning_rate=LEARNING_RATE),
    loss ='sparse_categorical_crossentropy',
    metrics = ['accuracy'])

# Treinamento
unet.fit(dataset_treinamento,
          validation_data=dataset_validacao,
          epochs = EPOCHS)

# Salvando o modelo no disco local
# Formato de nome da pasta do modelo baseado nos parametros usados
unet.save('modelos/modelo_' + str(EPOCHS) + '_' + str(INPUT_SIZE) + '_' +
          str(BATCH_SIZE), overwrite=True)
```

## A.5 inferencia.py

```
import keras
import tensorflow as tf
import os
import utils
import numpy as np
import PIL
import matplotlib.pyplot as plt
# -----
# Visualização de inferência e avaliação de modelo treinado
# -----

# Caminho da pasta onde o modelo está salvo, no formato SavedModel
caminho_modelo = ''

# O que vai ser usado para predição, pode ser uma imagem ou pasta, que deve conter uma base
# → de dados no formato VOC
caminho_dados = ''

# Mostrar a camada de resposta real ou apenas a camada de ativação da classe mitose
mostrar_camada_resposta = False

# Avaliar IoU(Intersection over Union) do modelo, apenas funciona com dados em pasta no
# → formato VOC
avaliar = False

# Carregando modelo e tamanho do formato da camada de entrada
modelo = keras.models.load_model(caminho_modelo)
camada_entrada = modelo.layers[0]
tamanho_entrada = (camada_entrada.output_shape[0][2], camada_entrada.output_shape[0][1])

# Se os dados apontarem diretamente para uma imagem, fazer a predição
if(os.path.isfile(caminho_dados)):
    imagem = PIL.Image.open(caminho_dados)
    imagem_predicao = utils.processar_imagem(caminho_dados, tamanho_entrada)
    imagem_predicao = np.expand_dims(imagem_predicao, 0)
    predicao = modelo.predict(imagem_predicao)

    if mostrar_camada_resposta:
        predicao = tf.argmax(predicao, axis=-1)
        predicao = predicao[..., tf.newaxis]
        predicao = tf.keras.preprocessing.image.array_to_img(predicao[0])
    else:
        predicao =
            → tf.keras.preprocessing.image.array_to_img(np.expand_dims(predicao[0][:,:,1],
            → axis=-1))

    predicao = utils.redimensionar_anotação(predicao, imagem.size)
    utils.visualizar(imagem, [], predicao)
```

```
# Se os dados forem uma pasta, assumir que é o formato VOC e também buscar/mostrar as
→ anotações
if(os.path.isdir(caminho_dados)):
    nomes_imagens = os.listdir(caminho_dados + "/JPEGImages")
    imagens = []
    imagens_predicao = []
    anotacoes = []

    for nome_imagem in nomes_imagens:
        nome_imagem = nome_imagem.split(".")[0]

        imagem = utils.processar_imagem(caminho_dados + "/JPEGImages/" + nome_imagem + ".jpg",
                                         → None)
        imagem_predicao = utils.processar_imagem(caminho_dados + "/JPEGImages/" + nome_imagem
                                         → + ".jpg", tamanho_entrada)
        imagem_predicao = np.expand_dims(imagem_predicao, 0)
        anotacao = utils.processar_anotacao(caminho_dados + "/SegmentationClassPNG/" +
                                         → nome_imagem + ".png", None)

        imagens.append(imagem)
        imagens_predicao.append(imagem_predicao)
        anotacoes.append(anotacao)

    predicoes = modelo.predict(np.vstack(imagens_predicao))
    ious = []

    for i in range(len(imagens)):
        predicao = predicoes[i]

        if avaliar:
            anotacao = anotacoes[i]

            predicao = tf.argmax(predicao, axis=-1)

            predicao = predicao[..., tf.newaxis]
            predicao = tf.keras.preprocessing.image.array_to_img(predicao)
            predicao = utils.redimensionar_anotacao(predicao,
                                         → tf.keras.preprocessing.image.array_to_img(np.expand_dims(anotacao,
                                         → axis=-1)).size)
            predicao = np.array(predicao) / 255

            ious.append(utils.iou_mascara_binaria(anotacao, predicao))
        else:
```

```
if mostrar_camada_resposta:
    predicao = tf.argmax(predicao, axis=-1)
    predicao = predicao[..., tf.newaxis]
    predicao = tf.keras.preprocessing.image.array_to_img(predicao)
else:
    anotacoes[i] = predicao[:, :, 0]
    predicao =
        ↪ tf.keras.preprocessing.image.array_to_img(np.expand_dims(predicao[:, :, 1],
        ↪ axis=-1))
    anotacoes[i] =
        ↪ tf.keras.preprocessing.image.array_to_img(np.expand_dims(anotacoes[i],
        ↪ axis=-1))
    anotacoes[i] = utils.redimensionar_anotacao(anotacoes[i],
        ↪ tf.keras.preprocessing.image.array_to_img(imagens[i]).size)

predicao = utils.redimensionar_anotacao(predicao,
    ↪ tf.keras.preprocessing.image.array_to_img(imagens[i]).size)
utils.visualizar(imagens[i], anotacoes[i], predicao)

if avaliar:
    print("Media de IoU do Modelo: " + str(np.mean(ious)))
```