

UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE FÍSICA

Jairo Sousa Santos

Clenilton Costa dos Santos

**APRIMORAMENTO DE APLICATIVO MOBILE PARA DETERMINAÇÃO VIA
INTELIGÊNCIA ARTIFICIAL DA ÁREA DE AMOSTRAS PARA MEDIDAS ELÉTRICAS**

São Luís - MA
2023

Informações do Bolsista

Nome: Jairo Sousa Santos

Telefone: (98) 989070-6054

E-mail: jairo.ss@discente.ufma.br

Informações da Instituição/Departamento

Nome: Universidade Federal do Maranhão

Endereço: Av. dos Portugueses, 1996, Bacanga - CEP 65080-805, São Luís - MA

Telefone: (98) 3272-8000

E-mail: atendimento@ufma.br

Informações do Orientador

Nome: Clenilton Costa dos Santos

Telefone: (98) 3272-9202

E-mail: cleniltoncs@gmail.com

RESUMO

Em planos de trabalho anteriores, foi idealizado um software que, por meio de algoritmos de machine learning, fosse capaz de medir a área da face de amostras, em geral pastilhas cerâmicas ou filmes poliméricos, utilizadas em medidas elétricas. A caracterização das propriedades elétricas das amostras consiste numa etapa crucial na descoberta de novos materiais avançados, sendo necessário, então, obter a área das faces planas opostas destas amostras. Todavia, a determinação da área das pastilhas é um processo trabalhoso e repetitivo, que despende muito tempo do pesquisador. Com isso em vista, um aplicativo mobile, de nome PAC (Pellet Area Calculator), automatizaria este processo, aumentando a produtividade do laboratório. Nesse sentido, o presente trabalho é focado no aperfeiçoamento do modelo que realiza o cálculo da área, e também na melhoria da interface gráfica do PAC. Isto foi executado separando o modelo em duas partes: detecção da escala, utilizando ideias de análise espectral, e segmentação das amostras, via inteligência artificial. O algoritmo de inteligência artificial em questão consiste numa U-Net, um tipo de rede neural convolucional especializada na segmentação de imagens. Assim, o modelo desenvolvido foi capaz de prever as áreas das amostras com um erro percentual médio de $\approx (2 \pm 1)\%$. Além disso, o aplicativo foi reescrito utilizando o framework Flutter, voltado para o desenvolvimento mobile, resultando numa interface gráfica agradável e de fácil uso. Espera-se, desta forma, melhorar o dia-dia dos pesquisadores que necessitam obter o valor das áreas superficiais das amostras de maneira rápida e fácil.

Palavras chaves: Inteligência Artificial. Redes Neurais Convolucionais. Visão computacional.

SUMÁRIO

1 INTRODUÇÃO	4
2 JUSTIFICATIVA	4
3 OBJETIVOS	4
3.1 Objetivo geral	5
3.2 Objetivos específicos	5
4 METODOLOGIA E FUNDAMENTOS TEÓRICOS	5
4.1 Determinação convencional da área das amostras	5
4.2 Introdução à análise de sinais	7
4.2.1 Procedimentos comuns no processamento de sinais	7
4.2.2 Fundamentos da análise espectral	8
4.2.3 Noções básicas sobre processamento de imagens	9
4.3 Introdução às redes neurais artificiais	10
4.3.1 Fundamentos biológicos	10
4.3.2 O neurônio artificial	11
4.3.3 Perceptron multicamadas	13
4.3.4 Treinamento de redes neurais	14
4.3.5 Generalização e memorização	15
4.3.6 Redes neurais convolucionais	16
5 RESULTADOS	16
5.1 O conjunto de dados	16
5.2 Detecção da escala pixel-milímetro	17
5.3 Segmentação via redes neurais convolucionais	19
5.3.1 A arquitetura	20
5.3.2 Função de custo e métricas	21
5.3.3 Treinamento do modelo	21
5.4 Melhorias na interface gráfica	22
6 CONCLUSÕES	24

1 INTRODUÇÃO

A caracterização de propriedades elétricas são fundamentais no desenvolvimento de novos materiais avançados, como células de baterias recarregáveis, capacitores de alta densidade de energia e varistores. Estas propriedades, como a condutividade, a densidade de corrente e a constante dielétrica do material são obtidos por intermédio de medidas elétricas, que podem ser realizadas sob corrente contínua (DC) ou alternada (AC), como no caso da espectroscopia de impedância eletroquímica (EIS) (Buonocore *et al.*, 2019; Sinfrônio *et al.*, 2018).

Em geral, as amostras são preparadas nas formas de filmes poliméricos ou pastilhas cerâmicas, com espessura $\sim 1\text{mm}$. Uma pequena diferença de potencial é aplicada às superfícies planas opostas das partilhas, previamente pintadas com tinta ou pasta de prata, de modo a caracterizar a densidade de corrente do material (Buonocore *et al.*, 2019; Sinfrônio *et al.*, 2018). Porém, a análise destas medidas requer informações a respeito da geometria das amostras, como a espessura, que pode ser aferida com um paquímetro, e a área superficial. Porém, obter a área superficial das amostras não é algo trivial, pois as pastilhas apresentam formatos variados.

A determinação da área superficial das pastilhas não é um trabalho complexo, mas despende boa parte do tempo do pesquisador, por se tratar de um procedimento repetitivo, além da quebra do fluxo do seu trabalho. Assim, em projetos anteriores, foi idealizado um software que realiza a detecção da área das amostras de maneira automática, via redes neurais convolucionais (CNNs). Entretanto, devido ao erro obtido em modelos antecedentes, foi necessário a construção de novos algoritmos que aprimorassem o processo de determinação das áreas, além da melhoria da interface gráfica (aplicativo PAC - Pellet Area Calculator).

2 JUSTIFICATIVA

Tendo em vista a falta de um software que calcule a área da superfície das pastilhas de maneira automática, o aprimoramento do aplicativo PAC apresentaria uma melhora significativa no processo de caracterização de novos materiais, pois pouparia os pesquisadores, tanto da academia como da industria, do processo trabalhoso que é obtê-la.

3 OBJETIVOS

3.1 Objetivo geral

- Aprimorar o aplicativo mobile Pellet Area Calculator (PAC), que determina a área da face de pastilhas cerâmicas ou filmes poliméricos sobre papel milimetrado.

3.2 Objetivos específicos

- Tirar fotografias de amostras (pastilhas cerâmicas e filmes poliméricos) de diferentes formatos e cores sobre papel milimetrado.
- Obter as áreas das pastilhas da forma tradicional (mais trabalhosa) a partir das fotografias, usando o ImageJ.
- Realizar novos treinamentos e testes da CNN adicionando ao conjunto de treinamento as novas imagens e seus respectivos valores alvo (áreas determinadas pelo ImageJ)
- Aperfeiçoar a interface gráfica do PAC de forma que ele seja de uso fácil, rápido e intuitivo.

4 METODOLOGIA E FUNDAMENTOS TEÓRICOS

4.1 Determinação convencional da área das amostras

O método convencional para a determinação da área é feita utilizando algum programa de análise de imagens. Dispondo a amostra sobre um papel milimetrado e posicionando câmera paralelamente à superfície, tira-se dela uma foto. Após isto, o pesquisador utiliza o programa de análise de imagens para encontrar a escala de conversão entre a distância em pixel e em milímetros (referenciada neste trabalho por escala pixel-milímetro).

Definida a escala, a face da pastilha é contornada manualmente, e, por fim, o programa calcula a área. Além disso, foi necessário a obtenção da máscara (ou ground truth), que consiste numa imagem preto e branco, com as mesmas dimensões da original, delimitando a região que pertence à pastilha. Com a finalidade de coletar a área e as máscaras, o programa ImageJ foi utilizado realizando o procedimento que será descrito a seguir.

Com a imagem aberta no ImageJ, a ferramenta *Straight* foi utilizada para traçar uma barra de referência com dimensões similares à da pastilha, como pode ser observado Figura 1a. Em seguida, a escala foi definida em *Analyze > Set Scale*, preenchendo o campo *Known distance* pela distância, em milímetros, correspondente ao comprimento

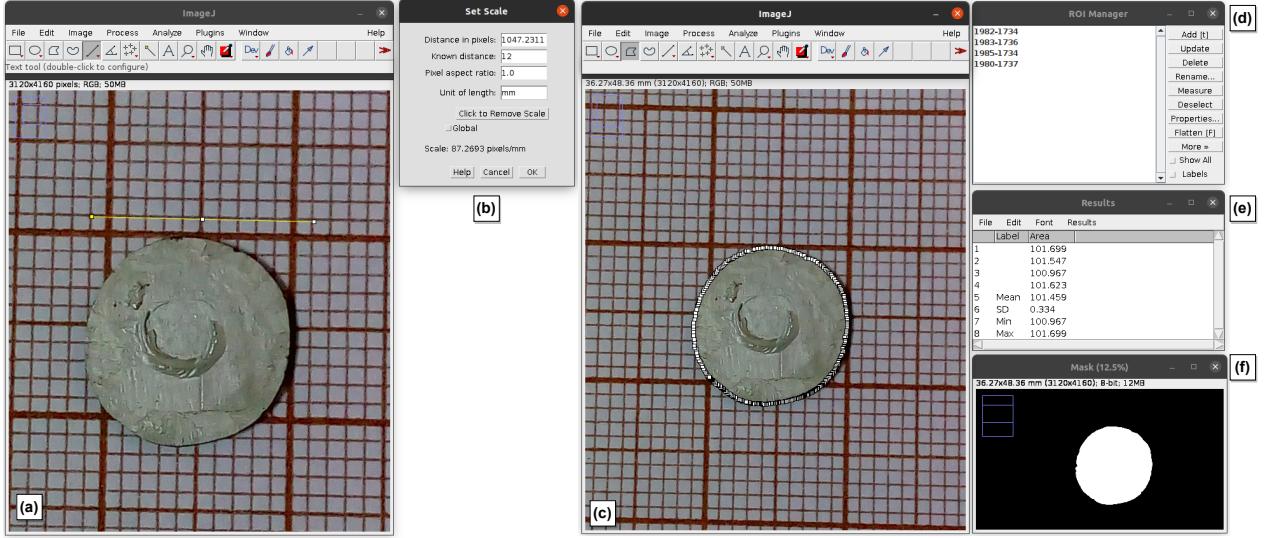


Figura 1: Procedimento de coleta da área e máscara de amostras via ImageJ. (a) Barra de escala (em amarelo) gerada pela ferramenta *Straigth*. (b) Janela *Set Scale*, com os valores referentes à linha traçada. (c) Contorno feito manualmente utilizando a ferramenta *Polygon selections*. (d) Janela *ROI Manager* mostrando os quatro diferentes contornos coletados variando a posição da barra de escala. (e) Janela *Results* com as áreas referentes à cada contorno e a média (“Mean”) destes valores. (f) Janela *Mask* exibindo a máscara criada a partir da seleção que contém a área mais próxima da média.

da barra (Figura 1b). Definida a escala, aplicou-se a ferramenta *Polygon selections* para contornar a amostra de maneira minuciosa, vide a Figura 1c. Fazendo o contorno da pastilha, um objeto do tipo *Selection* (seleção) é criado, e deve ser imediatamente inserido no gerenciador de seleções: indo em *Edit > Selection > Add to Manager*.

Um fato importante que deve ser levado em consideração, é a presença de distorções e aberrações nas fotos, tendo em vista que nem sempre a câmera está posicionada paralelamente à superfície onde a pastilha se encontra. Além, é claro, das distorções naturalmente causadas pela lente do aparelho. Pensando nisso, é necessário realizar os procedimentos de definição da escala quatro vezes, uma vez para cada região no entorno da amostra (cima, baixo, esquerda e direita), contornando a pastilha e salvando a seleção todas as vezes. Isto pode ser observado na janela *ROI Manager* (gerenciador de seleções) da Figura 1d.

Após isto, selecionando os contornos da janela *ROI Manager*, clica-se em *Measure*, fazendo surgir a janela *Results* com os valores das áreas referentes à cada contorno. Logo em seguida, na janela dos resultados (Figura 1e), é aplicada a opção *Results > Summarize* que gera a média, e algumas outras estatísticas, das áreas encontradas. E por fim, selecionando o contorno cuja área mais se aproxima da média, é criada uma máscara na opção *Edit > Selection > Create Mask* (vide a janela *Mask* da Figura 1f). O valor da média é utilizado para nomear os arquivos da imagem e da máscara, seguindo

um formato específico: “média_mm2.jpg” e “média_mm2.png”, respectivamente.

4.2 Introdução à análise de sinais

Sinais são funções, de uma ou mais variáveis, que representam determinada grandeza, como tensão e corrente em circuitos de corrente alternada, o nível do solo em uma determinada região ou a luminosidade em imagens. Os sinais se classificam em duas categorias principais, sinais contínuos e sinais discretos. Sinais contínuos são representações matemáticas dos fenômenos, que podem estar relacionados a alguma teoria subjacente ou serem puramente empíricas. Enquanto os sinais discretos correspondem às medições das grandezas propriamente ditas, que, na maioria das vezes, provém de sensores que convertem determinada informação em medidas elétricas: por exemplo microfones para áudios, termômetros para temperatura ou câmeras para imagens.

Em outras palavras, um sinal discreto é um sinal que possui variável independente discreta, esta discretização está ligada aos equipamentos que realizam as medições, dado que é impossível obter e armazenar medidas contínuas. Por isso, dado um sinal qualquer representado matematicamente pela função $y(t)$, sua forma discreta é simbolizada algebricamente por $y[n]$, sendo n um índice inteiro (Oppenheim; Willsky; Nawab, 2010).

4.2.1 Procedimentos comuns no processamento de sinais

Para fins deste trabalho, é necessário introduzir algumas operações comumente utilizadas na análise de sinais, como a convolução, correlação e o filtro Gaussiano. A convolução consiste na combinação de dois sinais distintos: dados dois sinais $x[n]$ e $h[n]$, a operação de convolução “ $*$ ” é definida por Richter *et al.* (2022, p. 52) como

$$(x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]. \quad (1)$$

Apesar de muito utilizada na análise de sinais, a convolução tem aplicações em diversas áreas como espectroscopia, probabilidade e engenharia. A correlação (ou cross-correlation), por sua vez, é uma operação que, se aplicada a dois sinais diferentes, extrai a similaridade entre eles; Richter *et al.* (2022, p. 53) define a correlação entre os sinais $x[n]$ e $h[n]$ como

$$\mathcal{C}(x, h)[n] = \sum_{k=-\infty}^{\infty} x[k]h[k+n]. \quad (2)$$

Observando as equações (1) e (2), é notável a semelhança entre as operações

de convolução e correlação, podendo gerar resultados iguais em alguns casos. Entretanto, vale ressaltar que ambas operações tem origem em conceitos mais básicos, fundamentados em álgebra linear e no produto interno de funções. Na prática, a convolução é a base da maioria dos filtros digitais, enquanto a correlação fornece uma medida de semelhança entre os sinais (Choudhury, 2017).

Filtros são processos realizados em sinais com o objetivo de realçar ou amenizar determinadas características. Nesse sentido, o filtro Gaussiano é utilizado para desfocar o sinal, reduzindo o ruído. Por ruído, entende-se o caráter aleatório presente na maioria dos sinal capturados por equipamentos e sensores. Assim, o filtro Gaussiano remove as chamadas altas-frequências, que são referentes às transições abruptas causadas por eventos aleatórios.

Um procedimento que também pode ser utilizado para reduzir o ruído é a chamada auto-correlação, definida pela correlação do sinal consigo mesmo:

$$\bar{\mathcal{C}}(x) = \mathcal{C}(x, x).$$

4.2.2 Fundamentos da análise espectral

Sinais periódicos são uma classe de sinais de extrema importância para este trabalho, e são assim chamados por apresentarem características que se repetem ao longo do domínio. Baseado nisso, a análise espectral consiste em um conjunto de métodos que tem o objetivo de caracterizar as formas e frequências destes sinais, e possui como fundamento a análise de Fourier.

Jean-Baptiste Joseph Fourier (1768-1830), foi um físico francês que, em trabalhos sobre a condução de calor, provou matematicamente que qualquer função contínua em um determinado intervalo pode ser descrita por uma soma de senos e cossenos com diferentes amplitudes e frequências, a chamada série de Fourier. Com base nestes trabalhos, surge a transformação de Fourier, método que se resume a representar uma função qualquer no espaço das frequências, facilitando o estudo de sua periodicidade.

No contexto do processamento de sinais, a transformada discreta de Fourier, ou DFT (sigla para Discrete Fourier Transform), pode ser definida na forma exponencial complexa como

$$X[\nu] = \frac{1}{2N} \sum_{n=0}^{2N-1} x[n] \exp\left(\frac{i\pi n\nu}{N}\right),$$

sendo $x[n]$ e $X[\nu]$ as formas discretas da função $f(t)$ e de sua transformação de Fourier contínua $F(\omega)$, respectivamente. Esta discretização é realizada considerando um conjunto de $2N$ valores de tempo, $t_n = nT/2N$ para o intervalo $(0, T)$. Desse modo, é

introduzido o espaço recíproco com $\omega_\nu = 2\pi\nu/T$, fornecendo $x[n] = f(t_n)$ e $X[\nu] = F(\omega_\nu)$ sendo n e ν índices inteiros (Arfken; Weber, 2017).

A transformada rápida de Fourier, ou FFT (sigla para Fast Fourier Transform), proposta em 1965 por J. W. Cooley e John Tukey, consiste em um algoritmo que reduz a complexidade computacional na aplicação da transformação de Fourier.

O espectro de um sinal, é o nome dado à sua representação no espaço das frequências. Ainda que poderoso, o algoritmo FFT nem sempre é uma boa opção na construção do espectro de sinais, principalmente quando existem processos estocásticas envolvidos na geração do sinal. Em outras palavras, os processos estocásticos estão relacionados com variáveis aleatórias, que acarretam no aparecimento de ruídos nos sinais. Por conseguinte, o PSD (power spectral density), ou densidade espectral de potência, auxilia na obtenção do espectro, amenizando efeitos indesejados causados por eventuais ruídos. Segundo Richter *et al.* (2022, p. 56), o PSD de um sinal pode ser obtido pela transformada discreta de Fourier da sua auto-correlação.

4.2.3 Noções básicas sobre processamento de imagens

Uma imagem digital é representada por um conjunto ordenado de elementos, chamados pixels, que armazenam níveis de cor. Imagens bidimensionais são assim chamadas por conterem pixels dispostos ao longo de duas dimensões, embora estes pixels possam armazenar até três valores (na maioria dos casos), chamados canais.

Tendo isso em vista, cada canal fornece o nível de determinada cor, sendo elas o vermelho (red), o azul (blue) e o verde (green), daí a sigla RGB. Para fins deste trabalho, entretanto, serão exploradas imagens monocromáticas, ou seja, com apenas um canal (tons de cinza). Algebricamente, uma imagem monocromática podem ser representada como um sinal bidimensional $I[m, n]$, que informa a intensidade de cor presente no pixel localizado na linha m e coluna n , sendo limitado ao intervalo $0 \leq I[m, n] \leq 1$.

Nesse contexto, Goodfellow, Bengio e Courville (2016, p. 332) define a convolução em uma imagem como

$$(I * K)[j, k] = \sum_m \sum_n I[j - m, k - n]K[m, n],$$

sendo K o chamado núcleo de convolução (ou kernel de convolução), que, dentre outras coisas, pode ser utilizado na aplicação de filtros na imagem (vide a representação na Figura 2).

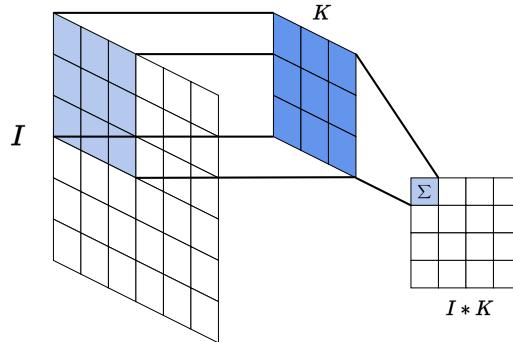


Figura 2: Representação da convolução entre uma imagem I e um kernel K .

4.3 Introdução às redes neurais artificiais

Redes neurais artificiais são algoritmos de inteligência artificial desenvolvidos para realizarem processamento de informações de forma análoga ao cérebro humano. Introduzidas pela primeira vez em 1943 pelo matemático Walter Pitts e pelo neurofisiologista Warren McCulloch no trabalho de título “A logical calculus of the ideas immanent in nervous activity” (Géron, 2019). Neste, McCulloch e Pitts apresentaram um modelo matemático simplificado sobre a propagação de sinais através dos neurônios, explicando como uma rede destes elementos é capaz de realizar cálculos complexos.

4.3.1 Fundamentos biológicos

Neurônios são os elementos estruturais do cérebro, consistem nas células responsáveis pela propagação de sinais na rede neural. Cada neurônio é dividido em três partes: corpo celular, dendritos e axônio. O corpo celular é a parte do neurônio que contém o núcleo celular e a maior parte das organelas. Os dendritos são filamentos celulares responsáveis por receber sinais que serão processados pelo neurônio. O axônio, por sua vez, consiste em apenas um filamento que pode se estender por comprimentos até dezenas de milhares de vezes maiores que o corpo do neurônio (Géron, 2019). Em uma de suas extremidades, o axônio é conectado ao corpo do neurônio, e na outra se ramifica em estruturas chamadas telodendros, onde, na ponta desses ramos, encontram-se os terminais sinápticos.

Basicamente, um determinado neurônio consegue captar sinais de outros neurônios por meio de reações eletroquímicas nas sinapses, entre os neurônios. Em geral, as sinapses possuem capacidades de transmissão de sinais diferentes, de modo que algumas conexões são priorizadas. Ao receber os sinais pelos dendritos, o neurônio tem a possibilidade de emitir um sinal como resposta, ou não. Isso acontecerá se os sinais recebidos ultrapassarem o limiar de ativação, gerando um sinal que é transmitido pelo axônio

até os telodendros e, por fim, atingindo outros neurônios. Algumas propriedades físicas do axônio, tais como alta resistência elétrica e uma grande capacidade de armazenamento, são responsáveis por reduzir, ou até anular, a amplitude do sinal emitido pelo corpo celular (Haykin, 2007).

4.3.2 O neurônio artificial

Baseado em modelos de neurônios artificiais semelhantes ao proposto por McCulloch e Pitts, Frank Rosenblatt desenvolveu o chamado perceptron, que se trata de uma das arquiteturas mais simples de redes neurais. O perceptron é a unidade de processamento de informação das redes neurais artificiais, e possui estruturas análogas aos neurônios biológicos.

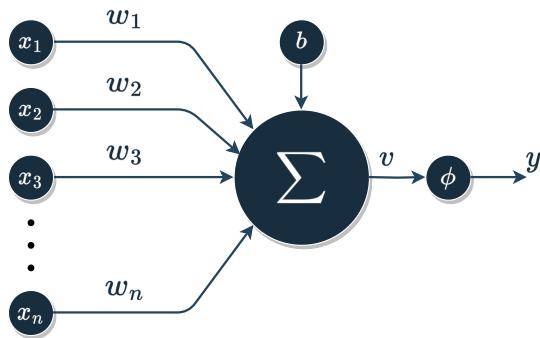


Figura 3: Representação de um perceptron.

De modo semelhante aos neurônios biológicos, o perceptron é composto por um conjunto de entradas, cada entrada caracterizada por um peso (o análogo às sinapses), numa estrutura semelhante aos dendritos. Como representado na Figura 3, um sinal x_k que passa por uma determinada entrada é ponderado por um peso w_k associado àquela entrada. Além disso, na maioria dos casos, é adicionado um parâmetro externo ao neurônio artificial, chamado bias b (ou viés), que é responsável por aplicar uma transformação afim à saída, além de ser o análogo ao limiar de ativação em um neurônio biológico. A soma dos sinais ponderados mais o bias define o campo induzido v ,

$$v = \sum_{k=1}^n x_k w_k + b,$$

que pode ser entendido como o análogo ao sinal emitido pelo corpo celular de um neurônio biológico.

E, por fim, o campo induzido é “filtrado” por uma função, chamada função de

ativação $\phi(v)$, que restringe a saída y do perceptron

$$y = \phi(v),$$

além de fornecer o caráter não-linear ao modelo. Neste contexto, podemos associar a função de ativação ao axônio no neurônio biológico.

A função de ativação pode assumir várias formas dependendo do contexto ao qual o perceptron é aplicado. Algumas das funções mais comuns são: função de Heaviside, ReLU e sigmoide. A função de Heaviside, cujo gráfico está mostrado na Figura 4a, é utilizada em modelos de classificação binária simples e pode ser definida por

$$\phi(v) = \begin{cases} 0, & \text{se } v \leq 0 \\ 1, & \text{se } v > 0 \end{cases}.$$

O modelo de neurônio artificial que utiliza a função de Heaviside como função de ativação é referido na literatura como modelo de McCulloch-Pitts, em homenagem ao trabalho por estes desenvolvido (Géron, 2019; Haykin, 2007). Em seu trabalho, eles demonstraram que uma rede constituída por uma quantidade suficientemente grande destas unidades de processamento, e com conexões sinápticas ajustadas adequadamente, seria capaz, a princípio, de aproximar qualquer função computável (Russell *et al.*, 2010; Haykin, 2007).

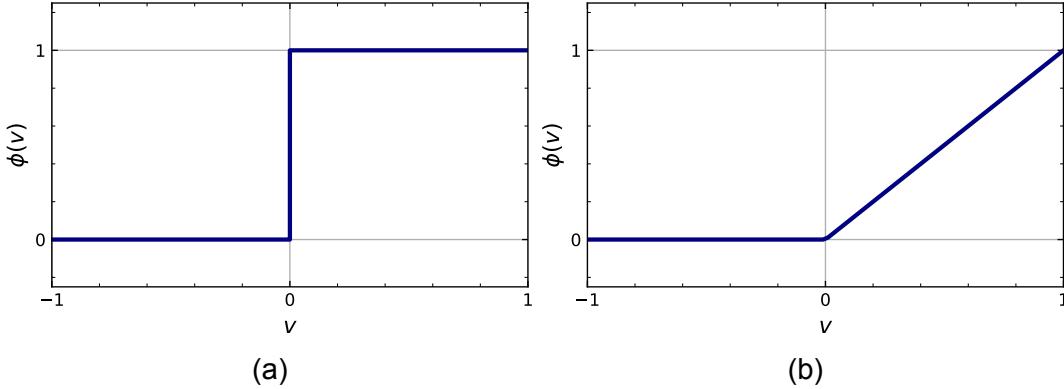


Figura 4: (a) Função de Heaviside. (b) Função ReLU.

A função de ativação ReLU, abreviação para Rectified Linear Unit (unidade linear retificada), é comumente dada na forma

$$\phi(v) = \max(0, v),$$

e está dentre as funções de ativação mais utilizadas atualmente (Figura 4b). O fato da função ReLU não alterar significativamente a saída do perceptron para campos induzi-

dos positivos facilita o aprendizado de uma rede neural profunda, reduzindo o chamado gradiente de fuga (vanish gradient).

A função de ativação sigmoide surge com o objetivo de fornecer uma característica probabilística à saída do perceptron, pois é de suma importância fornecer ao modelo de neurônio artificial um caráter estocástico. Considerando o modelo de McCulloch-Pitts, podemos definir uma $p(v, \alpha)$ que representa a probabilidade do neurônio disparar

$$y = \begin{cases} 1, & \text{com probabilidade } p(v, \alpha) \\ 0, & \text{com probabilidade } 1 - p(v, \alpha) \end{cases},$$

sendo α o parâmetro que controla a incerteza associada ao evento. A função logística é comumente utilizada para definir a probabilidade p :

$$p(v, \alpha) = \frac{1}{1 + e^{-v/\alpha}},$$

pois assume valores num intervalo contínuo entre 0 e 1. Note que na ausência de incertezas, $\alpha \rightarrow 0$, retornamos ao caso determinístico modelado pela função de Heaviside, vide o gráfico na Figura 5. Assim, a função sigmoide é definida como um caso particular da função logística,

$$\phi(v) = p(v, 1) = \frac{1}{1 + e^{-v}}.$$

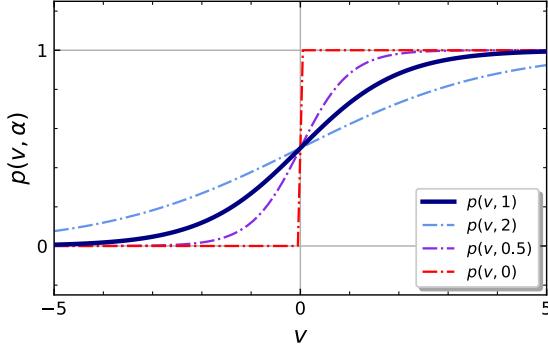


Figura 5: Função logística.

4.3.3 Perceptron multicamadas

Ao desenvolverem seu modelo, McCulloch e Pitts tinham o conhecimento que a utilidade dos neurônios artificiais estava intrinsecamente relacionada com combinações que poderiam ser feitas entre os percetrons, de modo que é necessário arranjá-los em uma estrutura, chamada rede neural (Haykin, 2007). As arquiteturas de redes neurais

mais comuns tem como base na junção de várias camadas de perceptrons, nos quais os dados de entrada dos neurônios de determinada camada provém das saídas dos neurônios da camada anterior.

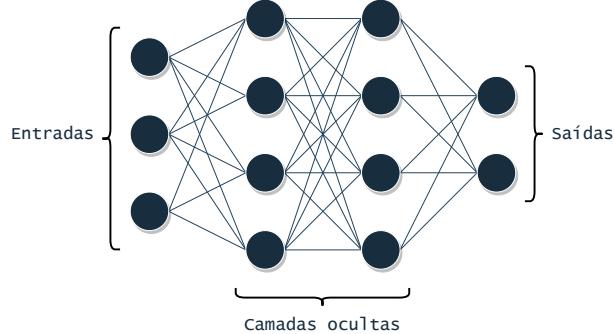


Figura 6: Representação de uma rede Feedforward totalmente conectada.

A rede neural representada na Figura 6 é dita totalmente conectada, pois os neurônios de determinada camada se conectam com todos os neurônios das camadas adjacentes. A adição de camadas ocultas resulta na extração de estatísticas de ordem elevada, tornando valiosas as redes com grandes número de camadas ocultas. A quantidade de camadas ocultas fornece a profundidade (ou deep) da rede, da onde surge o termo Deep Learning (ou aprendizado profundo).

4.3.4 Treinamento de redes neurais

Após definida uma arquitetura, é necessário ajustar os pesos w (e bias) da rede neural para obter o máximo de eficiência. Este processo é chamado treinamento da rede neural, e tem como base a minimização de uma função de custo $J(w)$ (ou função de perda), que avalia o erro da rede. Dentre os métodos mais utilizados, algoritmos baseados em descida de gradiente (ou gradient descent) ocupam lugar de extrema importância, dado que, na maioria dos casos, a minimização da função de custo se configura como um problema de otimização não-convexo.

No contexto das redes neurais, o algoritmo de descida de gradiente consiste em um método numérico de optimização, no qual os pesos da uma função de custo são atualizados em razão do gradiente desta função, buscando minimizá-la. O gradiente da função fornece a “direção” no espaço multidimensional dos pesos, de modo que, partindo de valores iniciais, a descida de gradiente é responsável por alterá-los, levando-os ao ponto no qual a função é mínima.

O optimizador Adam, é um algoritmo que surge na tentativa de reduzir as inconveniências causadas pela não-concavidade da função de custo, sendo resultado da união

de métodos: optimização Momentum e RMSProp. A optimização Momentum, consiste numa extensão da descida de gradiente com a adição de um termo inercial, enquanto o método de RMSProp (abreviação de root mean squared propagation) utiliza a média móvel dos gradientes na atualização da taxa de aprendizado. Ambos os métodos tem por objetivo acelerar e aumentar a eficiência da descida de gradiente.

4.3.5 Generalização e memorização

Após o treinamento, dizemos que uma rede neural é generalizada quando o mapeamento entrada-saída por ela fornecido é correto, em média, para um conjunto de dados diferente daquele utilizado no treinamento. Entretanto, quando a rede neural é exposta excessivamente aos dados de treinamento, acaba memorizando-os, perdendo a capacidade de generalização. Este fenômeno é conhecido como overfitting, ou super-ajuste (Haykin, 2007).

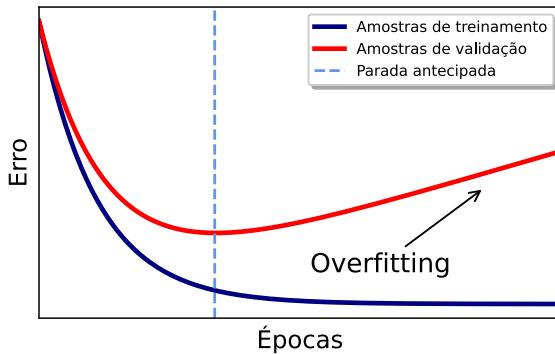


Figura 7: Representação do aprendizado de uma rede neural genérica, mostrando as curvas de erro nos dados de treinamento (em azul) e nos dados de validação (em vermelho) com o passar das épocas de iteração, além da parada antecipada (em ciano) marcando a melhor configuração alcançada.

Assim, é necessário separar os dados disponíveis em amostras de validação e amostras de treinamento, para verificar se a rede neural alcançou o nível esperado de generalização em abas. Como observá-se na Figura 7, após um determinado ponto de mínimo, a curva do erro nas amostras de validação começa a crescer, caracterizando um overfitting. Sendo recomendado utilizar no modelo final os pesos encontrados no ponto de mínimo da curva, técnica conhecida como parada antecipada (ou early stopping).

Além da parada antecipada, existem diversas técnicas que podem ser utilizadas para evitar o overfitting e obter melhores resultados no treinamento de redes neurais artificiais, por exemplo batch normalization e data augmentation. O data augmentation é uma técnica aplicada aos dados de treinamento (e validação), com o objetivo de aumentar o conjunto de dados disponíveis através de algumas transformações (por exemplo trans-

formações geométricas, adição de ruídos e recortes), expondo a rede a mais variações (Yang *et al.*, 2022).

4.3.6 Redes neurais convolucionais

Redes neurais convolucionais (ou CNNs, abreviação para convolutional neural networks), pertencem a uma classe de redes neurais especializadas no processamento de imagens. Extremamente importantes na atualidade, as redes neurais convolucionais se baseiam no processo de convolução, introduzido na Subseção 4.2, para filtrar características da imagem e extrair as informações de interesse. Estas redes são estruturadas em camadas, onde cada camada possui um conjunto de filtros, ou kernels de convolução.

Diferente dos perceptrons, os kernels de convolução armazenam os pesos espacialmente, permitindo a extração de padrões de imagens. Após a convolução, uma função de ativação (tipicamente a função ReLU) é aplicada à saída, com a finalidade de fornecer o caráter não-linear do modelo.

Em geral, é comum estruturar uma CNN intercalando camadas de convolução, ativação e pooling. Camadas de pooling são responsáveis por varrer a imagem, aplicando determinada estatística ao longo desta, com a finalidade de fazê-la uma representação invariante por pequenas translações. Este processo reduz o tamanho da imagem original por um fator inversamente proporcional ao tamanho do pooling (ou pooling size) (Goodfellow; Bengio; Courville, 2016). Dentre as camadas de pooling mais comuns encontra-se o max pooling, que reduz a imagem extraíndo o valor máximo na região baseada num filtro quadrado.

5 RESULTADOS

O projeto foi dividido em duas partes principais: construção de um modelo que detecta a área das amostras e construção de uma interface gráfica mobile. Com relação à detecção da área, foram elaborados dois algoritmos: um para detecção da escala pixel-milímetro e outro focado na segmentação da região da pastilha via rede neural convolucional. Ambos os programas escritos na linguagem Python. A interface gráfica, por sua vez, se trata de um aplicativo desenvolvido utilizando a ferramenta Flutter, criada pelo Google, especializada em aplicações mobile. O aplicativo em questão foi escrito em Dart, linguagem base do Flutter.

5.1 O conjunto de dados

Ao longo do projeto, um conjunto de módulos Python foram criados para facilitar

a manipulação e análise dos dados. Estes módulos foram condensados numa pequena biblioteca, nomeada “src” (abreviação de *source*). Dentre os módulos, o “src/data.py” foi implementado, com enfoque no pré-processamento das imagens utilizadas para treinar e testar os modelos.

Em posse dos arquivos da amostra e de sua respectiva máscara, as imagens foram recortadas num formato padrão de 256×256 , centralizado no centro de massa da pastilha. Após isso, as imagens foram aleatoriamente separadas em dois arquivos: “train” para arquivos referentes ao treinamento e “test” para arquivos referentes à validação dos modelos. Assim, foram totalizadas 64 amostras, sendo $\approx 76,6\%$ (49) destinadas ao treinamento e $\approx 23,4\%$ (15) destinadas à validação.

5.2 Detecção da escala pixel-milímetro

O algoritmo de detecção da escala pixel-milímetro foi fundamentado na análise espectral, apresentada na Subseção 4.2, dado o caráter periódico provocado pelo papel milimetrado ao fundo das imagens. Na Figura 8, é exibida uma imagem $\mathcal{I}[m, n]$ como exemplo, evidenciando a periodicidade dos sinais nas projeções das linhas $\mathcal{I}_n[m]$ e colunas $\mathcal{I}_m[n]$. Dessa forma, o problema se resumiu a extrair a frequência ν referente ao tamanho de 1mm (em pixel).

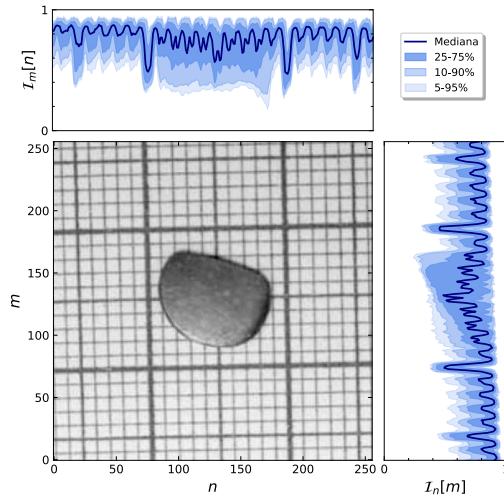


Figura 8: Imagem de exemplo $\mathcal{I}[m, n]$ com as projeções das colunas $\mathcal{I}_m[n]$ e das linhas $\mathcal{I}_n[m]$, revelando o caráter periódico causado pelo papel milimetrado.

Num primeiro momento, houve a análise do espectro FFT das projeções, e verificou-se que um dos picos de frequência consistia naquela referente aos traços do papel milimetrado (vide o exemplo na FFT das colunas $\mathcal{I}_m[n]$ na Figura 9). A frequência ν_{true} destes traços pode ser obtida pelas informações da área real A_{true} e da área em pixels da

pastilha. A área em pixels era encontrada utilizando a máscara $G[m, n]$ de cada amostra, dado que

$$G[m, n] = \begin{cases} 1, & \text{se } \mathcal{I}[m, n] \text{ pertence à pastilha} \\ 0, & \text{se } \mathcal{I}[m, n] \text{ não pertence à pastilha} \end{cases}.$$

Assim, defini-se $\nu_{true} = \sqrt{\alpha_{true}}$, sendo α_{true} a escala pixel-milímetro verdadeira:

$$\alpha_{true} = \frac{\sum G}{A_{true}},$$

onde \sum é a soma sobre todos os elementos.

Ainda que detectável via FFT, houveram dificuldades na extração da frequência de interesse, de modo que foi necessário a implementação de algumas ferramentas de análise de sinais com o objetivo de filtrar os picos do espectro e isolar ν_{true} . Primeiramente, foi preciso reduzir a contribuição de frequências próximas do zero, calculando a derivada discreta da imagem $\frac{\partial}{\partial n} \mathcal{I}_m[n]$ ao longo do eixo que se pretende analisar. Como pode-se observar no exemplo da Figura 9, a derivada faz com que ν_{true} seja destacada no espectro, embora isso também aconteça com alguns picos de altas frequências.

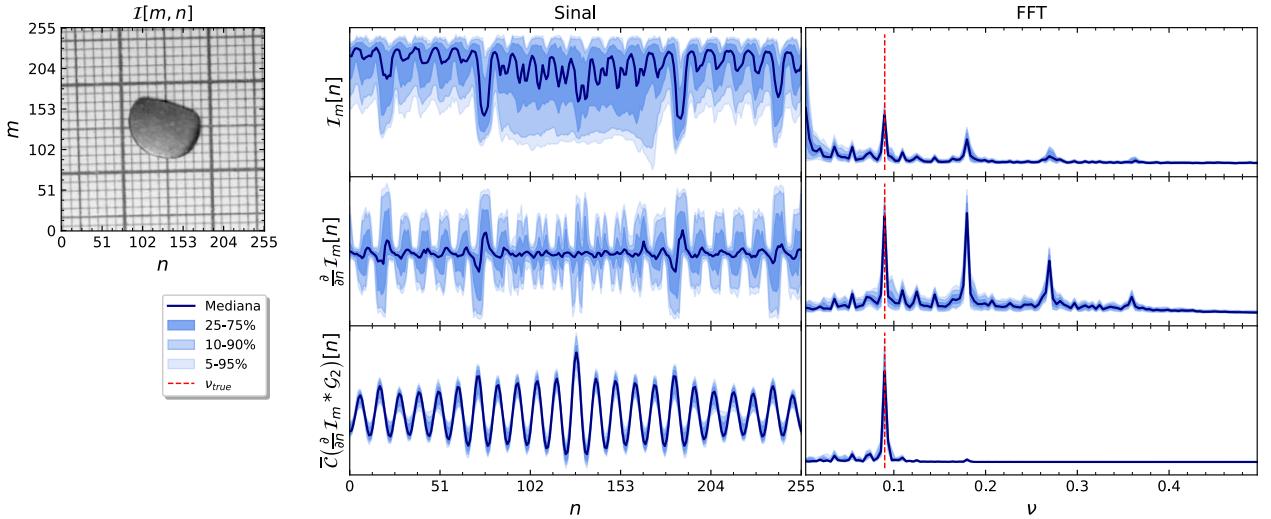


Figura 9: Exemplo do processo de filtragem da frequência ν_{true} de uma imagem, comparando os sinais (projeções horizontais) com a FFT para cada função aplicada.

Tal como explicado na Subseção 4.2, o filtro gaussiano pode ser aplicado para retirar contribuições das altas frequências nos sinais. A aplicação deste filtro é realizada em imagens por meio da convolução com um kernel gaussiano \mathcal{G}_σ , onde σ é um parâmetro que determina a abertura do filtro, também relacionado com o desvio padrão na densidade de probabilidade gaussiana.

Então, realizando a convolução da derivada discreta com o kernel gaussiano

$\frac{\partial}{\partial n} \mathcal{I}_m * \mathcal{G}_\sigma$, os picos de alta frequência eram retirados do espectro. Após isso, foi implementado o PSD, realizando a auto-correlação e a FFT dos sinais (linhas e colunas da imagem). Como pode ser visto no exemplo da Figura 9, por meio destes passos encontrou-se um espectro com apenas um pico, sendo a frequência deste pico o valor estimado da frequência verdadeira.

Em suma, dado a imagem de uma amostra, seguem-se os passos para cada eixo:

- calcula-se a derivada discreta na direção em que se pretende obter a frequência;
- aplica-se o filtro gaussiano, por meio da convolução da derivada com o kernel \mathcal{G}_σ ;
- percorrendo os sinais ao longo do eixo perpendicular, calcula-se o PSD;
- extrai-se a frequência que corresponde ao maior pico do espectro.

Assim, obtém-se ν_x para o eixo horizontal e ν_y para o eixo vertical, e por consequência, a escala estimada $\alpha_{pred} = \nu_x \nu_y \approx \alpha_{true}$.

Buscando minimizar o erro médio na detecção da escala das amostras, o valor do parâmetro σ foi ajustado para $\sigma = 2$, conforme exibido na Figura 10a. Procedendo como o descrito acima para todas as amostras do banco de dados, encontrou-se um erro percentual médio de $\approx (2 \pm 1)\%$. A acurácia do modelo está representada na Figura 10b.

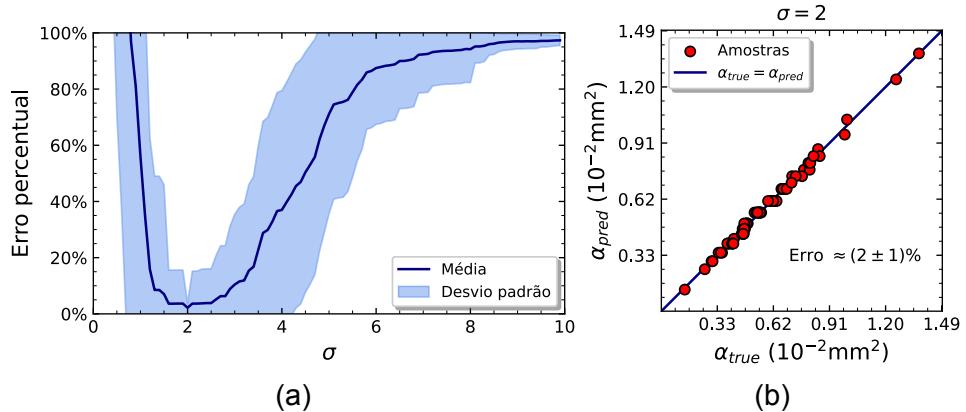


Figura 10: (a) Erro percentual médio da detecção das escalas das amostras em função do parâmetro σ do kernel gassiano, revelando o seu mínimo em $\sigma = 2$. (b) Gráfico de dispersão, representando a acurácia do método ($\sigma = 2$) em estimar a escala pixel-milímetro α_{true} das amostras.

5.3 Segmentação via redes neurais convolucionais

5.3.1 A arquitetura

O modelo de segmentação utilizado neste trabalho consiste numa adaptação da arquitetura de rede neural convolucional proposta por Ronneberger, Fischer e Brox (2015), criada com enfoque na segmentação de imagens médicas. Conhecida como U-Net, esta arquitetura possui seu funcionamento baseado na codificação (ou contração) da imagem entrada em um tensor de valores abstratos, seguido da decodificação (ou expansão) destes valores numa imagem saída (ou máscara, para o caso da segmentação).

No modelo original, a rede é constituída de blocos de convolução, separadas por camadas de max pooling (na codificação) ou camadas de deconvolução (na decodificação). Os blocos de convolução são formados de duas camadas de convolução totalmente conectadas, com a mesma quantidade de filtros e ativações do tipo ReLU. Cada bloco de convolução desce ou sobe um nível nas etapas de codificação e decodificação, respectivamente. Estas etapas são simétricas (em modelos simples), ou seja, possuem o mesmo número de níveis. Além disso, são utilizadas camadas de concatenação para unir os níveis de cada etapa.

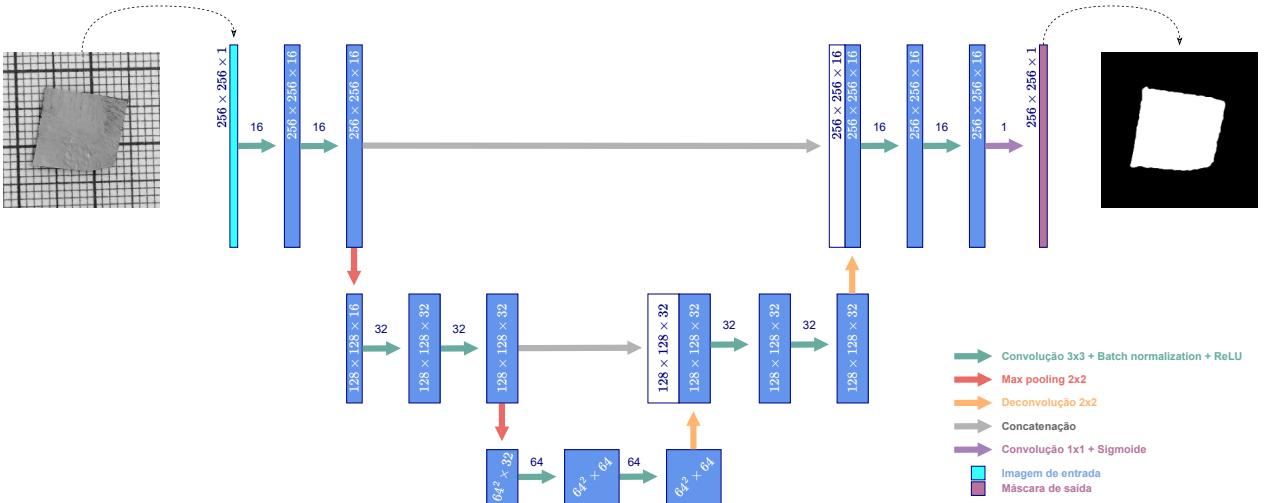


Figura 11: Arquitetura de U-Net utilizada neste trabalho, mostrando um exemplo de segmentação realizada pela rede após o treinamento.

A U-Net aqui aplicada se trata de uma simplificação do modelo original, dado a menor complexidade computacional da tarefa. Assim como representado no diagrama da Figura 11, a rede neural foi construída com 3 níveis. No primeiro nível, os blocos de convolução receberam 16 filtros, enquanto 32 filtros foram aplicados no segundo nível, e 64 ao terceiro. Todas as camadas de convolução foram configuradas com kernels de tamanho 3×3 .

As camadas de max pooling e deconvolução, por sua vez, foram ajustadas para realizarem o redimensionamento com filtros de tamanho 2×2 . Por fim, era aplicada mais uma convolução no resultado do último bloco de convolução, porém com apenas um filtro e ativação do tipo sigmoide, para fornecer a probabilidade de cada pixel pertencer à superfície da pastilha.

A arquitetura apresentada acima foi implementada com o auxílio da biblioteca Tensorflow, uma ferramenta código aberto especializada em machine learning, que permitiu a construção do modelo com extrema facilidade através da sua API funcional. Os códigos referentes à rede neural foram compartmentados no módulo “src/segmentation.py”. Os treinamentos, por sua vez, ocorreram na plataforma Google Colaboratory.

5.3.2 Função de custo e métricas

Assim como explicado na Subseção 4.3, é necessário definir uma função de custo para avaliar o aprendizado do modelo. Então, tendo em vista a ativação do tipo sigmoide na última camada da rede (vide Figura 11), a função binary cross-entropy (entropia cruzada binária). Considerando que $S[m, n]$ corresponda à segmentação realizada pela rede neural, a binary cross-entropy $J_{BCE}(G, S)$ pode ser definida para imagens como

$$J_{BCE}(G, S) = -\frac{1}{N} \sum G \log S + (1 - G) \log (1 - S),$$

sendo \sum a soma sobre de todos os elementos, e N a área total (em pixel) das imagens (Ma *et al.*, 2021).

Além disso, para melhor avaliar o aprendizado da rede, foi aplicada a métrica *IoU* (intersection over union)

$$IoU(G, S) = \frac{G \cap S}{G \cup S} = \frac{\sum GS + (1 - G)(1 - S)}{\sum G(2 - S) - (1 - G)(1 - S)}$$

com a finalidade de extrair informações sobre a segmentação, e também o erro percentual absoluto das áreas E_{area} (em pixel)

$$E_{area}(G, S) = \frac{|\sum G - \sum S|}{\sum G} \cdot 100\%,$$

utilizada para definir a parada antecipada (Ma *et al.*, 2021).

5.3.3 Treinamento do modelo

O treinamento da rede foi realizado utilizando as imagens de treinamento e vali-

dação, já previamente separadas. Porém, para melhorar o aprendizado, foi aplicado uma aumentoção dos dados (data augmentation) via espelhamentos horizontais e verticais, expandindo o conjunto de imagens em quatro vezes, com 196 destinadas ao treinamento e 60 para validação (totalizando 256 amostras). Neste caso, o algoritmo Adam de optimização, já disponibilizado pelo Tensorflow, foi aplicado para ajustar os parâmetros.

A Figura 12 mostra os gráficos da função de custo J_{BCE} e da métrica IoU em função das épocas de treinamento, evidenciando o chamado “Grokking Phenomeon” (fenômeno Grokking). Descrito pela primeira vez por Power *et al.* (2022), o fenômeno Grokking ocorre quando o modelo memoriza o conjunto de treinamento antes de aprender a generalizá-lo, contradizendo o que é observado no overfitting clássico (vide Subseção 4.3.5).

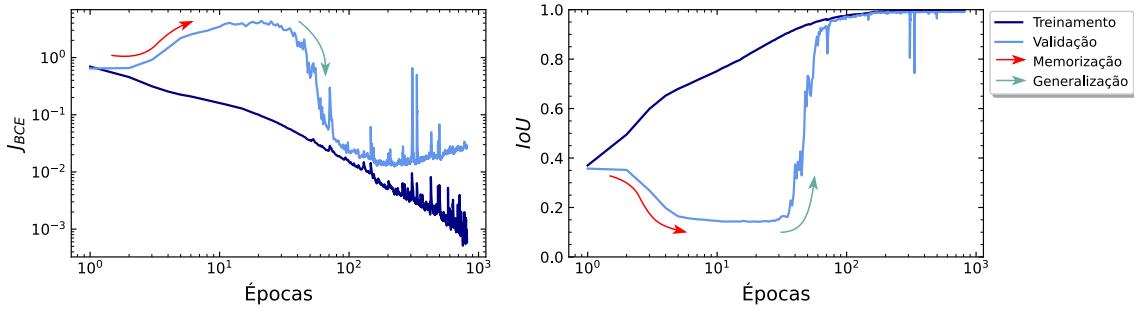


Figura 12: Curvas de aprendizado do modelo após 814 épocas, com a função binary cross-entropy J_{BCE} (à esquerda) e a métrica IoU (à direita) em escala logarítmica, ressaltando o fenômeno Grokking (memorização antes da generalização).

Apesar de presente em todos os treinamentos realizados neste trabalho, o efeito Grokking não influenciou na convergência do modelo. Pois, após começar o processo de generalização, a rede manteve a média do erro percentual por volta de 1% para as amostras de validação, como pode ser analisado na Figura 13a. O mínimo de erro percentual das amostras de validação foi encontrado na época 480 com o valor de $E_{area}^{(validação)} \approx (0.4 \pm 0.4)\%$, cuja acurácia está representada na Figura 13b.

Unindo o resultado obtido pela rede neural com o programa de detecção da escala pixel-milímetro, foi encontrado um erro percentual médio da área (em milímetros) de $\approx (2 \pm 1)\%$, demonstrando a capacidade do modelo em prever a área da superfície das amostras.

5.4 Melhorias na interface gráfica

O aplicativo PAC (Pallet Area Calculator), idealizado em planos de trabalhos anteriores, foi reconstruído em Flutter, como dito anteriormente, o que possibilitou o seu rápido desenvolvimento.

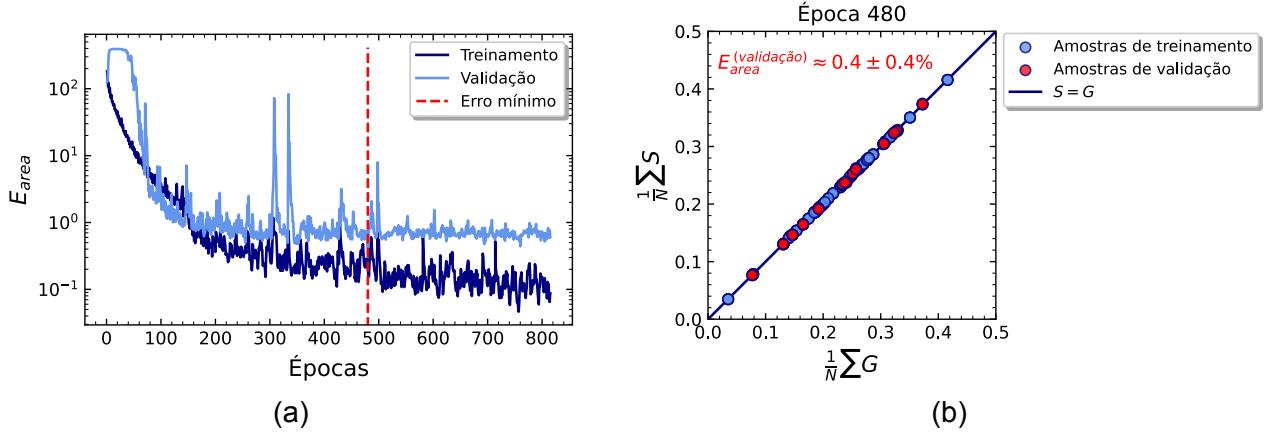


Figura 13: (a) Evolução do erro percentual médio ao longo das épocas de treinamento, além do ponto de parada (em vermelho). (b) Gráfico de dispersão, representando a acurácia do modelo em prever a área (em pixel) das pastilhas.

Como representado no fluxograma da Figura 14a, a interface gráfica (front end) é composta de três etapas: coleta da imagem, pré-processamento (corte e rotação) e exibição dos resultados. Enquanto isso, o back end foi feito em Python por meio da API Flask, sendo responsável por aplicar os modelos de segmentação, detecção da escala e pós-processamento. O pós-processamento consiste na aplicação de alguma manipulação morfológica (remoção de buracos, excessos, binary cossing, etc...), que pode ser aplicada para melhorar a segmentação.

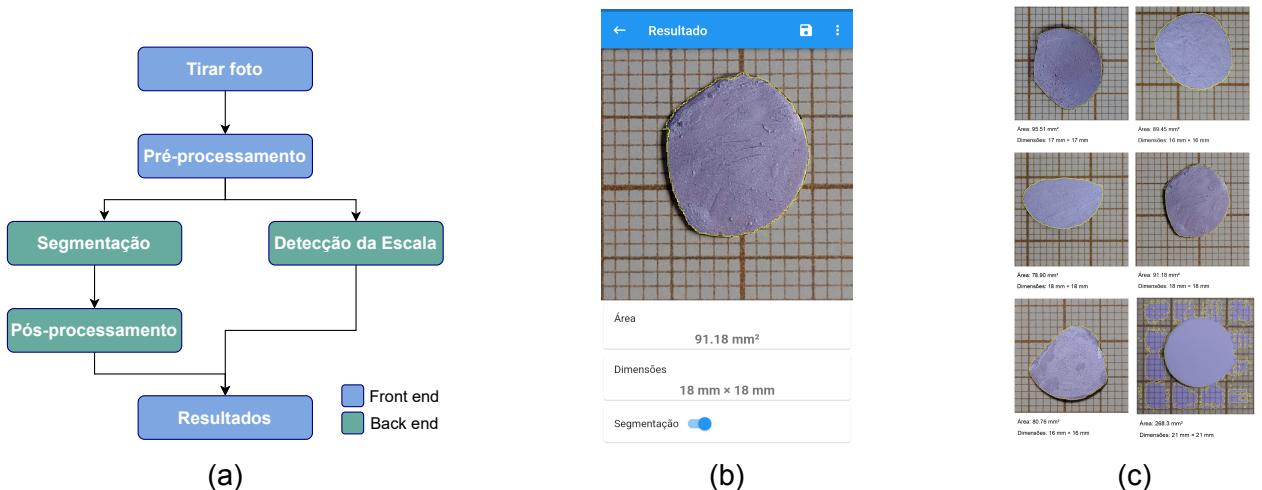


Figura 14: (a) Representação do funcionamento do aplicativo. (b) Resultado da área de uma amostra feita pelo aplicativo. (c) Resultados salvos da área de algumas pastilhas obtidas pelo aplicativo, com um exemplo de falha do modelo.

Na seção de resultados do aplicativo, foram adicionadas algumas informações, como as dimensões do papel milimetrado e a possibilidade de observar a segmentação

(Figura 14b), para ajudar o pesquisador a identificar quando o modelo falhou (Figura 14c). Nesse sentido, é importante que o usuário realize várias medidas com o app, afim de extrair a média dos resultados.

6 CONCLUSÕES

Unindo o algoritmo de detecção da escala pixel-milímetro com o modelo de segmentação, foi obtido um erro médio de $\approx (2 \pm 1)\%$ para amostras de validação. A arquitetura U-Net, ainda que razoavelmente simples, se demonstrou excepcional em segmentar a região das pastilhas, obtendo um erro de $\approx (0.4 \pm 0.4)\%$ no conjunto de validação.

O algoritmo de detecção da escala, por sua vez, provou-se ser uma boa opção para converter a área em pixel para a área em milímetros, com um erro médio de $\approx (2 \pm 1)\%$ para todas as amostras. Contudo, a utilização deste algoritmo requer que o papel milimetrado esteja bem alinhado com a imagem, pois uma pequena angulação pode afetar sua precisão. Porém, estes problemas podem ser facilmente contornados rotacionando as imagens antes de aplicar o algoritmo.

Com relação ao aplicativo, este foi reescrito com o framework Flutter, o que tornou o processo de desenvolvimento mais fluido, além de fornecer uma interface gráfica agradável e simples de se utilizar. Até o momento da escrita deste relatório, o aplicativo está em fase de testes e ainda não está disponível para o público geral. Entretanto, há a previsões para o lançamento de uma versão beta, além de uma versão em desktop.

Desde seu início, o projeto foi salvo num repositório remoto, de maneira que todas as etapas da análise das amostras, da construção dos modelos e do desenvolvimento do aplicativo estão disponíveis em [PAC-project](#).

REFERÊNCIAS BIBLIOGRÁFICAS

- ARFKEN, G.; WEBER, H. **Física Matemática**: Métodos Matemáticos para Engenharia e Física - tradução da 7 edição. [S.I.]: Elsevier Brasil, 2017. ISBN 9788535287776.
- BUONOCORE, A. L. W. *et al.* Varistor behavior in a ternary system based on SnO₂ doped with a hexavalent donor: SnO₂-MnO₂-WO₃. **Journal of Alloys and Compounds**, v. 811, p. 151538, 2019. ISSN 0925-8388.
- CHOUDHURY, D. Teaching the concept of convolution and correlation using Fourier transform. In: LIU, X.; ZHANG, X.-C. (Ed.). **14th Conference on Education and Training in Optics and Photonics: ETOP 2017**. [S.I.]: SPIE, 2017. v. 10452. International Society for Optics and Photonics, 104520y. DOI: [10.1117/12.2267976](https://doi.org/10.1117/12.2267976).

- GÉRON, A. **Mãos à Obra Aprendizado de Máquina com Scikit-Learn & TensorFlow**: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes. [S.I.]: Alta Books, 2019. ISBN 9788550803814.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.I.]: MIT Press, 2016.
- HAYKIN, S. **Redes neurais**: princípios e prática. 2. ed. São Paulo: Bookman Editora, 2007. ISBN 0132733501.
- MA, J. *et al.* Loss odyssey in medical image segmentation. **Medical Image Analysis**, v. 71, p. 102035, 2021. ISSN 1361-8415. DOI: <https://doi.org/10.1016/j.media.2021.102035>.
- OPPENHEIM, A.; WILLSKY, A.; NAWAB, S. **Sinais e Sistemas**. 2. ed. São Paulo: Prentice-Hall, 2010.
- POWER, A. *et al.* **Grokking**: Generalization Beyond Overfitting on Small Algorithmic Datasets. [S.I.: s.n.], 2022. arXiv: [2201.02177 \[cs.LG\]](https://arxiv.org/abs/2201.02177).
- RICHTER, M. *et al.* **Signal Processing and Machine Learning with Applications**. [S.I.]: Springer International Publishing, 2022. ISBN 9783319453712.
- RONNEBERGER, O.; FISCHER, P.; BROX, T. **U-Net**: Convolutional Networks for Biomedical Image Segmentation. [S.I.: s.n.], 2015. arXiv: [1505.04597 \[cs.CV\]](https://arxiv.org/abs/1505.04597).
- RUSSELL, S. *et al.* **Artificial Intelligence**: A Modern Approach. [S.I.]: Prentice Hall, 2010.
- SINFRÔNIO, F. S. M. *et al.* Effect of Co-substitution on the Vibrational, Magnetic, and Dielectric Properties of Copper Ferrites Obtained by Microwave-Assisted Hydrothermal Method. **Journal of Electronic Materials**, v. 47, n. 11, p. 6821–6832, 1 nov. 2018. ISSN 1543-186X. DOI: [10.1007/s11664-018-6598-6](https://doi.org/10.1007/s11664-018-6598-6).
- YANG, S. *et al.* **Image Data Augmentation for Deep Learning**: A Survey. [S.I.: s.n.], 2022. arXiv: [2204.08610 \[cs.CV\]](https://arxiv.org/abs/2204.08610).