UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FÁBIO HALBEN GUERRA LEAL

AVALIAÇÃO DA VIABILIDADE DE SPIKING NEURAL NETWORKS EM DISPOSITIVOS MÓVEIS PARA DIAGNÓSTICO DE CÂNCER DE PELE

BAURU
Novembro/2025

FÁBIO HALBEN GUERRA LEAL

AVALIAÇÃO DA VIABILIDADE DE SPIKING NEURAL NETWORKS EM DISPOSITIVOS MÓVEIS PARA DIAGNÓSTICO DE CÂNCER DE PELE

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. Leandro Aparecido Passos Junior

BAURU
Novembro/2025

L435a

Leal, Fábio
Avaliação da viabilidade de spiking neural networks em dispositivos móveis para diagnóstico de câncer de pele / Fábio Leal. -- Bauru, 2025
49 f. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru
Orientador: Leandro Aparecido Passos Junior

1. Redes neurais. 2. Inteligência artificial. 3. Melanoma. 4. Spiking Neural Network. 5. Convolutional Neural Networ. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Fábio Halben Guerra Leal

Avaliação da Viabilidade de Spiking Neural Networks em Dispositivos Móveis Para Diagnóstico de Câncer de Pele

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Leandro Aparecido Passos Junior
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Profa. Dra. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Prof. Dr. Douglas Rodrigues
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, 11 de Novembro de 2025.

Dedico este trabalho a todos os meus familiares e amigos que me acompanharam até aqui.

Agradecimentos

Agradeço primeiramente aos meus pais, Marcia Halben Guerra e Roberto Carlos Leal, e ao meu irmão Téo Halben Guerra Leal, por terem me apoiado e me acompanhado durante toda a minha vida. Agradeço à minha vó Lilo e a todos os meus tios,tias, primos e primas pelo o amor.

Agradeço a todos os meus amigos que conheci em Bauru e que me acompanharam nesta fase da minha vida, em especial meus colegas Christian Barry, Franco, Vini, Sofia, Rufino, Yoshio, Daniel, Luciano, Leo Azuma, Henrique e Luana.

Agradeço aos meus amigos de São Paulo, com quem mantive contato mesmo morando do outro lado do estado: Luiz Henrique Assumpção, Gabo, Luiz, Bernardo, Astolfo, Brunão, Bia, Debo, Ana, Murilo, Enzo, Wilson, Enrico, Jão e Iago.

Agradeço ao meu orientador, Prof. Dr. Leandro Aparecido Passos Junior, pelo auxílio e apoio.

Agradeço a todos os professores, desde a educação básica até a faculdade, que contribuíram para a minha formação; sem eles, eu não seria quem sou hoje.

"Despite everything, it's still you."
Toby Fox, em UNDERTALE (2015)

Resumo

Este trabalho investiga a viabilidade de Spiking Neural Networks (SNNs) em dispositivos móveis para apoio ao diagnóstico de câncer de pele, comparando-as com uma arquitetura convolucional convencional em condições controladas. Utilizou-se um conjunto de dados derivado da ISIC, disponível no Kaggle, estratificado em treino (5,000 benignas e 4,605 malignas) e teste (500/500), com pré-processamento padronizado (redimensionamento para 224×224, conversão para tons de cinza e normalização em $[0, 1]$). A CNN, inspirada na AlexNet, foi treinada por 20 épocas com seleção do melhor ponto via F1-macro; seu desempenho no teste atingiu acurácia de $90{,}6\%$, com precisão e *recall* próximos de $0{,}9$, e teve o gasto energético medido com o Zeus. A SNN espelhou a arquitetura da CNN, substituindo ativações por camadas LIF ($\beta = 0{,}8$) ao longo de 10 *time steps* e treinada com *surrogate gradients*; no teste, obteve $89{,}3\%$ de acurácia e $F1 = 0{,}8997$ ($\approx 1{,}3$ p.p. abaixo da CNN). Quanto à energia, registrou-se na CNN um gasto de $\approx 25{,}300$ J durante o treinamento e ≈ 111 J durante a fase de testes. A SNN não foi medida no hardware convencional, pois seus ganhos se manifestam sobretudo em hardware neuromórfico, no qual a literatura sugere uma economia de energia melhor em comparação à CNN. Em síntese, os resultados indicam que SNNs são podem ser viáveis quando a restrição energética é crítica e há acesso a hardware neuromórfico, embora persista um pequeno decréscimo de acurácia e desafios de engenharia.

Palavras-chave: Spiking Neural Networks; Convolutional Neural Networks; Redes Neurais Artificiais; diagnóstico médico; câncer de pele; dispositivos móveis; eficiência energética.

Abstract

This work investigates the feasibility of Spiking Neural Networks (SNNs) on mobile devices to support skin cancer diagnosis, comparing them with a conventional convolutional architecture under controlled conditions. An ISIC-derived dataset, available on Kaggle, was used, stratified into training (5,000 benign and 4,605 malignant) and test (500/500), with standardized preprocessing (resizing to 224×224, conversion to grayscale and normalization in $[0, 1]$). The CNN, inspired by AlexNet, was trained for 20 epochs with best checkpoint selection via F1-macro; its test performance reached $90{,}6\%$ accuracy, with precision and recall close to $0{,}9$, and had energy consumption measured with Zeus. The SNN mirrored the CNN architecture, replacing activations with LIF layers ($\beta = 0{,}8$) over 10 *time steps* and trained with *surrogate gradients*; in the tests, it obtained $89{,}3\%$ accuracy and $F1 = 0{,}8997$ ($\approx 1{,}3$ p.p. below the CNN). Regarding energy, the CNN recorded a consumption of $\approx 25{,}300$ J during training and ≈ 111 J during the testing phase. The SNN was not measured on conventional hardware, because its gains manifest mainly on neuromorphic hardware, in which the literature suggests that SNNs are generally more energy efficient when compared to their CNN counterpart. In summary, the results indicate that SNNs are may be viable when the energy constraint is critical and there is access to neuromorphic hardware, although a small decrease in accuracy and engineering challenges persist.

Keywords: Spiking Neural Networks; Convolutional Neural Networks; Artificial Neural Networks; Medical diagnosis; Skin cancer; Mobile devices; Power efficiency.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

SNN	Spiking Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
IF	Integrate and Fire
LIF	Leaky Integrate and Fire

Sumário

1 Introdução

Redes neurais artificiais (ANNs, do inglês *Artificial Neural Network*) são modelos matemáticos inspirados em observações de sistemas biológicos. Uma ANN pode ser descrita como um mapeamento de um espaço de entrada para um espaço de saída, análogo ao conceito de funções matemáticas (PRIDDY; KELLER, 2005). O que torna as ANNs particularmente relevantes é sua capacidade de resolver uma ampla classe de problemas de reconhecimento de padrões (WALCZAK, 2019). Graças a essa versatilidade, elas vêm se consolidando como solução em diversos domínios, como negócios (Tkáč & Verner, 2016; Wong, Lai & Lam, 2000), engenharia (Ali et al., 2015; Bansal, 2006) e medicina (Reggia, 1993; Yardimci, 2009).

Apesar de suas vantagens, as ANNs apresentam um elevado custo energético. Em aplicações que exigem alta eficiência computacional e energética como por exemplo veículos autônomos, drones e robôs colaborativos, esse consumo se torna um limitador (YAMAZAKI et al., 2022). Para mitigar esse problema, pesquisadores aproximaram ainda mais o modelo artificial do funcionamento do cérebro humano, que, apesar de suas inúmeras funções, consome cerca de 25 watts (KANDEL; SCHWARTZ, 1985). Nesse contexto surgem as *Spiking Neural Networks* (SNNs).

As SNNs são compostas por neurônios pulsantes (*spiking neurons*), cujo mecanismo de comunicação se inspira diretamente no sistema nervoso biológico: a informação é codificada no momento preciso de um pulso (*spike*) e/ou na sequência temporal de pulsos (GHOSH-DASTIDAR; ADELI, 2009). Em geral, a adoção de *spiking neurons* reduz significativamente o gasto energético da rede quando comparada às ANNs; por outro lado, a acurácia tende a ser inferior em grandes conjuntos de dados (YAMAZAKI et al., 2022), e o projeto/treinamento das SNNs costuma ser mais complexo, o que pode dificultar sua adoção.

Como exposto, as ANNs têm sido amplamente empregadas em diferentes áreas. À medida que novas arquiteturas são propostas, cresce o interesse em avaliar se elas podem desafiar a dominância das redes tradicionais em aplicações específicas. As SNNs não são exceção; contudo, por se tratar de uma abordagem relativamente recente, ainda há poucas análises sistemáticas sobre sua viabilidade e aplicabilidade no contexto médico. Diante desse cenário, este trabalho investiga a viabilidade de SNNs em dispositivos móveis para o diagnóstico de câncer de pele, considerando suas vantagens e limitações.

1.1 Problemática

As redes neurais artificiais, especialmente as redes neurais profundas (DNNs, do inglês *Deep Neural Networks*), enfrentam desafios significativos relacionados ao consumo de energia,

desafio que se intensifica em aplicações embarcadas e móveis. Um exemplo emblemático ocorre em dispositivos de Internet das Coisas (IoT). Em 2016, um experimento com óculos inteligentes da Google, equipados com uma GPU (*Graphics Processing Unit*) considerada estado da arte à época, executou uma DNN para reconhecimento de objetos. Assumindo-se a GPU totalmente dedicada à inferência e a bateria alocada exclusivamente para essa tarefa, estimou-se uma autonomia de apenas 25 minutos (VENKATARAMANI; ROY; RAGHUNATHAN, 2016). Tal resultado é claramente insatisfatório para o uso prático dos óculos inteligentes, pois demonstra que o uso contínuo da rede neural torna inviável a utilização prolongada do dispositivo.

Em instituições de saúde, o uso de ANNs já é corrente (Reggia, 1993; Yardimci, 2009), apesar do alto consumo energético. Entretanto, em regiões remotas ou carentes, onde o acesso a tecnologia e infraestrutura tende a ser restrito, e a demanda por diagnósticos e análises via dispositivos móveis pode ser maior, essa limitação pode tornar-se crítica. Nesses cenários, o consumo elevado pode inviabilizar o uso sustentado de ANNs e, em situações extremas, pode representar a diferença entre êxito ou fracasso de uma intervenção clínica.

Embora existam estudos de análises e aplicações de SNNs na área médica (Choi (2024), Xiaoxue et al. (2023)), grande parte desses trabalhos descreve apenas prós e contras no nível da aplicação específica, sem discutir, de forma comparativa e sistemática, em quais condições as SNNs superam modelos tradicionais. Falta, portanto, uma investigação orientada à viabilidade, técnica e energética, em casos de uso onde as vantagens das SNNs (p. ex., a eficiência energética) possam ter peso maior que suas desvantagens.

1.2 Justificativa

Sendo as SNNs uma tecnologia relativamente recente, ainda são escassos os estudos que avaliam sua aplicabilidade para além do âmbito teórico. No contexto médico, especialmente em soluções portáteis como ferramentas de apoio ao diagnóstico, as experiências práticas existentes tendem a aplicar a tecnologia sem uma análise abrangente de seus benefícios e malefícios de sua utilização em um setor tão sensível.

Câncer de pele é uma das causas de morte mais comuns no mundo, e muitos realizam o diagnostico visualmente, observando a olho nu as lesões, isso pode ser propicio ao erro humano. Uma das populares aplicações de ANN e mais especificamente CNN são para auxiliar em diagnósticos que dependem de analises visuais de imagens de testes médicos como o raio X por exemplo ou de lesões como é no caso do câncer de pele.

Ademais, um aplicativo para diagnóstico de câncer de pele é uma aplicação ideal para analisar a viabilidade de SNN, pois o diagnóstico visual, baseado nas marcas na pele, pode gerar erros. Uma inteligência artificial que permita a realização do diagnóstico de forma remota pode exemplificar onde as SNNs podem se destacar, principalmente se aplicada em contextos em que as vantagens das SNN podem se destacar, como em áreas mais necessitadas, onde

o acesso a tecnologia e a infraestrutura é limitada e o uso de uma ferramenta que pode ser utilizada em um dispositivo móvel se torna mais útil.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo principal deste trabalho é analisar a viabilidade de Spiking Neural Networks (SNN) em dispositivos móveis para o diagnóstico de câncer de pele, comparando-as com modelos tradicionais de Redes Neurais Artificiais (ANN). Como resultado, serão realizadas comparações rigorosas entre um modelo SNN e outro ANN, ambos desenvolvidos especificamente para esta pesquisa.

1.3.2 Objetivos Específicos

- Desenvolver e treinar tanto a ANN quanto a SNN.
- Pesquisar e utilizar o melhor método de medir gasto energético possível.

2 Fundamentação Teórica

2.1 *Artificial Neural Networks* e *Convolutional Neural Networks*

2.1.1 Classificadores

Classificadores são modelos que aprendem um mapeamento de um espaço de atributos (características extraídas dos dados) para um conjunto finito de rótulos, com o objetivo de atribuir, a cada nova amostra, a classe mais provável. Em termos gerais, o processo envolve estimar uma função de decisão que particiona o espaço de entrada em regiões associadas a cada classe, buscando minimizar um erro esperado sob uma distribuição desconhecida dos dados. A escolha de atributos e o critério de decisão adotado influenciam diretamente a capacidade do classificador de generalizar para exemplos não vistos.

Do ponto de vista estatístico, diferentes famílias de classificadores implementam hipóteses distintas sobre a estrutura dos dados. Métodos por "vizinhança", por exemplo, assumem que amostras de classes semelhantes tendem a estar próximas no espaço de atributos, a regra do vizinho mais próximo é um caso clássico, com propriedades assintóticas conhecidas em termos de erro esperado (COVER; HART, 1967). Já métodos de margens máximas procuram separar classes por fronteiras com maior folga, reduzindo a probabilidade de erro de generalização, abordagem formalizada pelos *Support Vector Machines* (CORTES; VAPNIK, 1995).

Outra linha importante são os classificadores de conjunto (*ensemble*), que combinam múltiplos modelos fracos para produzir um modelo forte. *Random Forests* constroem uma coleção de árvores de decisão e agregam suas previsões para reduzir variância e melhorar a robustez (BREIMAN, 2001). Em paralelo, métodos aditivos e de *boosting* otimizam iterativamente funções de perda ao incorporar novos componentes ao modelo, oferecendo garantias e interpretações estatísticas para a melhoria de desempenho (FRIEDMAN, 2001).

Na prática, o desempenho de um classificador depende não apenas do algoritmo, mas também de fatores como qualidade e quantidade de dados, balanceamento entre classes, seleção de atributos, regularização e protocolo de validação.Além disso, aspectos de interpretabilidade, tempo de inferência e requisitos computacionais costumam orientar a escolha do classificador em cenários reais.

2.1.2 Redes Neurais Convolucionais

As *Convolutional Neural Networks* (CNNs) são um tipo de ANN amplamente utilizado em tarefas de classificação. Elas utilizam convoluções, operações matemáticas que permitem a extração automática de características diretamente dos dados, reduzindo a necessidade de

extração manual de *features* característico de métodos mais antigos. A entrada de uma CNN, em geral, possui estrutura matricial (p. ex. imagens) ((HE et al., 2015)).

Assim como outras redes neurais, uma CNN é composta por uma camada de entrada, uma ou mais camadas escondidas e uma camada de saída. Nas camadas escondidas, pelo menos uma deve ser convolucional, responsável por extrair características por meio da aplicação de filtros aos dados de entrada, esses filtros identificam padrões presentes na matriz.

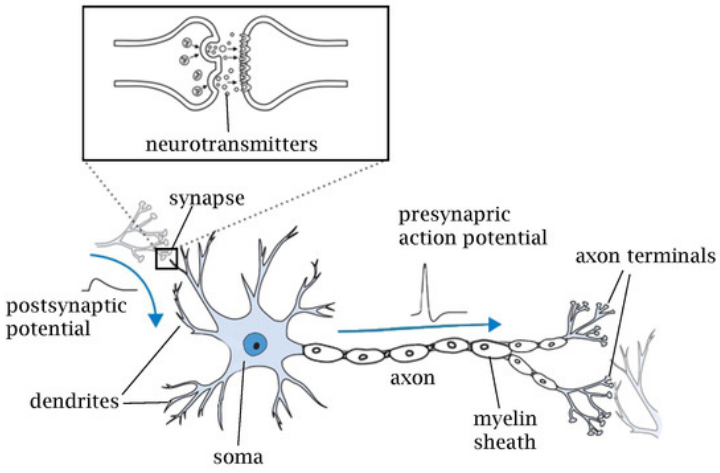
Outro tipo de camada importante em CNNs são as camadas de *pooling*. Seu objetivo é reduzir a dimensionalidade espacial (altura e largura) da entrada enquanto preserva as características mais relevantes detectadas pelas camadas convolucionais. Os dois tipos mais comuns são *max pooling* e *average pooling*. No *max pooling*, seleciona-se o maior valor dentro de cada janela da matriz de entrada, produzindo uma saída com dimensões menores que tende a reter as características mais proeminentes. Já o *average pooling* realiza procedimento análogo, mas calcula a média dos valores na janela, gerando uma saída reduzida que preserva informações médias da região.

2.2 *Spiking Neural Networks*

2.2.1 *Spiking Neuron* e o modelo *Leaky Integrate And Fire*

Os neurônios são as unidades básicas de processamento do sistema nervoso; eles processam informação por meio de sinais eletroquímicos que se propagam na forma de potenciais de ação. A Figura 1 ilustra a estrutura típica de um neurônio com seus quatro componentes principais: dendritos, soma (corpo celular), axônio e sinapses (YAMAZAKI et al., 2022). Dois neurônios conectam-se por uma sinapse: o neurônio que envia a informação é chamado de pré-sináptico, e o que a recebe, de pós-sináptico. Em termos funcionais, uma sinapse "mais forte" tende a produzir potenciais pós-sinápticos de maior amplitude no neurônio pós-sináptico, aumentando a probabilidade de disparo.

Figura 1 – Modelo do neurônio biológico

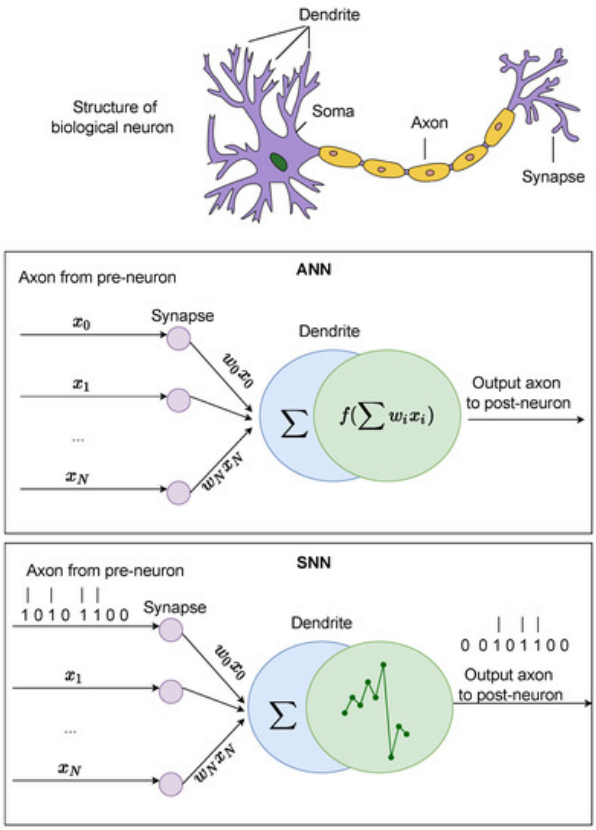


Fonte: (ESHRAGHIAN et al., 2023)

Os dendritos podem ser entendidos como a "entrada" do neurônio: receptores membranares ligam-se aos neurotransmissores liberados pelo neurônio pré-sináptico, convertendo sinais químicos em correntes e potenciais elétricos. O soma é o corpo celular onde os potenciais de membrana, gerados por sinais sinápticos, são integrados temporal e espacialmente (YAMAZAKI et al., 2022). O axônio pode ser visto como a "saída" do neurônio: ele conduz o potencial de ação até suas terminações, cuja arborização estabelece sinapses com outras células.

A maior parte das ANNs usa um modelo de neurônio artificial que, embora inspirado no neurônio biológico, simplifica fortemente o processo. Em um neurônio artificial típico, os sinais de entrada são representados por números (geralmente um vetor), cada componente associado a um peso específico, computa-se então uma combinação linear das entradas ponderadas e adiciona-se um viés (bias). Em seguida, aplica-se uma função de ativação (também chamada de função de transferência) à soma, produzindo a saída do neurônio (SUZUKI, 2011). Em termos funcionais, pode-se traçar uma analogia aproximada: as entradas corresponderiam ao papel dos dendritos, a integração/transformação da soma ao soma (corpo celular), e a saída ao axônio. Essa analogia, entretanto, é apenas heurística e não captura a dinâmica temporal, os fenômenos não lineares dendríticos nem a natureza pulsada dos potenciais de ação. Uma comparação visual dos diferentes modelos de neurônio é demonstrada na Figura 2.

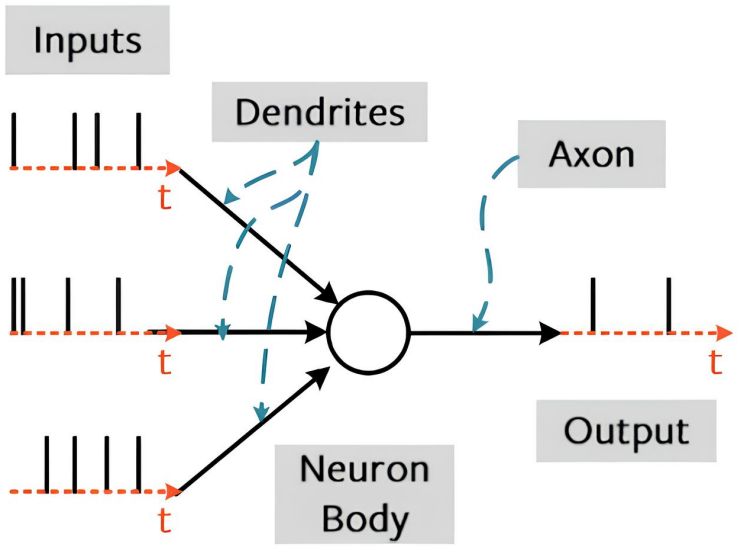
Figura 2 – Comparação entre o neurônio biológico, o neurônio artificial e o *spiking neuron*



Fonte: (YAMAZAKI et al., 2022)

Um modelo de neurônio artificial mais próximo do neurônio biológico é o *spiking neuron*, seu uso é justamente o que diferencia SNNs de ANNs. Assim como nas ANNs, os valores de entrada são multiplicados por seus respectivos pesos e depois somados. A diferença é que, no *spiking neuron*, essa soma contribui para o potencial de membrana do neurônio, denotado por $U(t)$ se o potencial de membrana for suficientemente elevado pela ação das entradas e atingir um limiar θ , o neurônio emite um pulso (*spike*) para suas conexões subsequentes. Contudo, a maior parte das entradas, em contextos biológicos, consiste em pulsos breves de atividade elétrica (eventos sinápticos). É pouco provável que todos cheguem simultaneamente ao corpo do neurônio. Isso indica a presencia de dinâmicas temporais que "sustentam"o potencial da membrana ao longo do tempo (YAMAZAKI et al., 2022). A Figura 3 exemplifica como o neurônio biológico pode ser comparado a o *spiking neuron*.

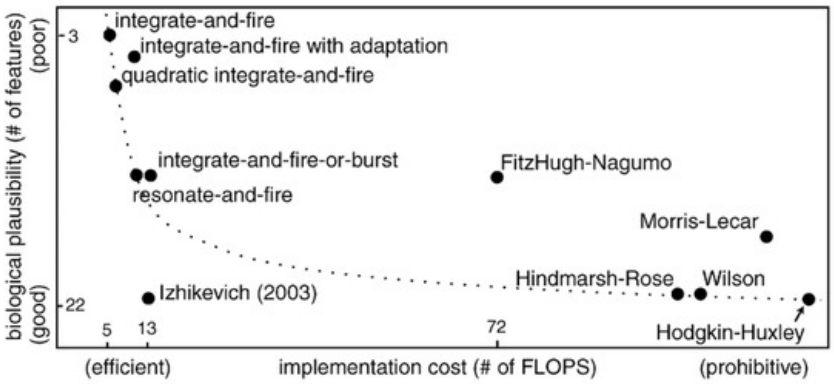
Figura 3 – Spiking neuron comparado com elementos do neurônio biológico



Fonte: (ESHRAGHIAN et al., 2023)

Existem diversos modelos de spiking neurons (demonstrado mais a fundo na Figura 4), cada um com suas vantagens e desvantagens. O modelo mais utilizado é o Leaky Integrate-and-Fire (LIF), em razão de sua simplicidade e do baixo custo computacional, o que frequentemente se traduz em boa eficiência energética em implementações digitais e neuromórficas. Os demais modelos também apresentam benefícios: quando o foco é melhorar o desempenho em tarefas com forte dependência temporal, SNNs recorrentes tendem a ser mais eficazes, se o objetivo é elevar ainda mais o desempenho, mesmo ao custo de eficiência, podem ser empregadas variantes com unidades de memória, como spiking LSTMs. Já quando o foco é a fidelidade biológica, o modelo de Hodgkin–Huxley e o modelo de Izhikevich reproduzem com maior realismo a dinâmica de disparo neuronal, ainda que com maior custo computacional (ESHRAGHIAN et al., 2023).

Figura 4 – Uma comparação entre diferentes modelos de *spiking neuron* em termos de custo computacional e fidelidade biológica



Fonte: (YAMAZAKI et al., 2022)

O modelo *Integrate and Fire* (IF) integra a entrada ao longo do tempo até que o potencial de membrana atinja um valor de limiar, ao atingir esse limiar, o neurônio dispara um pulso e é reinicializado. O IF não leva em conta a biofísica detalhada dos íons e canais do neurônio. Sua variante, o LIF, introduz um termo de "vazamento" (leak) que representa a perda passiva de carga pela membrana em direção ao potencial de repouso, aproximando o relaxamento em direção ao equilíbrio.

As dinâmicas do LIF na forma contínua podem ser representadas por:

$$C_m \frac{dv_m}{dt} = -G_L\left(v_m - E_L\right) + I_{\text{syn}}(t) \tag{2.1}$$

$$\text{se } v_m \geq v_\theta, \quad v_m \leftarrow v_{\text{peak}} \quad \text{então} \quad v_m \leftarrow v_{\text{reset}} \tag{2.2}$$

Onde v_θ é a voltagem de limiar, v_{peak} representa o pico do potencial de ação, e v_{reset} é o valor de reinício do potencial de membrana. Quando a voltagem atinge o limiar v_θ, o neurônio dispara um pulso, e a voltagem é reiniciada para 0 durante um período τ_{ref}, que limita a frequência de disparo do neurônio (YAMAZAKI et al., 2022).

Para simulação digital e treinamento de SNNs, normalmente se utiliza uma versão discreta por instantes de tempo (ESHRAGHIAN et al., 2023):

$$U[t] = \underbrace{\beta U[t-1]}_{\text{decaimento}} + \underbrace{WX[t]}_{\text{entrada}} - \underbrace{S_{\text{out}}[t-1]\theta}_{\text{reset}}. \tag{2.3}$$

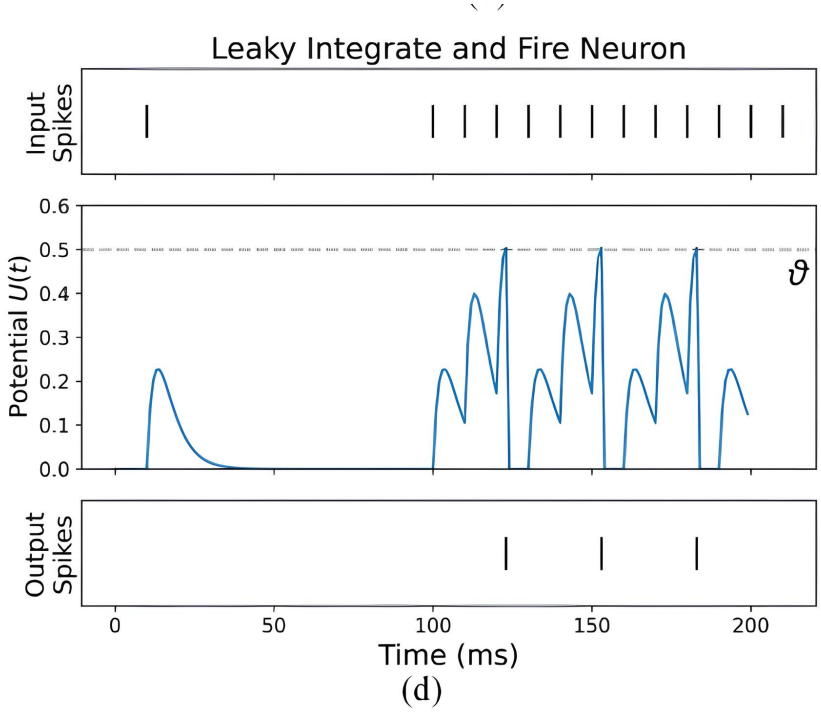
Onde $U[t]$ representa o potencial no passo t, $\beta = e^{-1/\tau}$ representa o fator de decaimento ou a "(constante inversa de tempo)"de $U[t]$, $WX[t]$ é a entrada no momento t multiplicada pelo seu peso W.

$S_{\text{out}}[t] \in \{0, 1\}$ é o pulso de saída gerado pelo neurônio, onde se ativado ($S_{\text{out}} = 1$), o termo de reset subtrai o limear θ do potencial da membrana. Caso contrário, o termo de reset não tem efeito ($S_{\text{out}} = 0$). Resumidamente um pulso é gerado se o potencial da membrana ultrapassar o limear de acordo com a seguinte equação (ESHRAGHIAN et al., 2023):

$$S_{\text{out}}[t] = \begin{cases} 1, & \text{if } U[t] > \theta \\ 0, & \text{caso contrário.} \end{cases} \tag{2.4}$$

Uma simulação do LIF e suas dinâmicas é demonstrada na Figura 5.

Figura 5 – Simulação de LIF demonstrando o potencial da membrana $U(t)$ alcançando o limiar $\theta = 0.5V$, assim gerando um pulso.



Fonte: (ESHRAGHIAN et al., 2023)

2.2.2 *Encoding* e *decoding* de dados em SNNs

Luz é o que se vê quando a retina converte fótons em pulsos de energia. Odores são o que se percebe quando o nariz processa moléculas voláteis em pulsos. Percepções de tato correspondem ao que se sente quando os nervos transformam pressão em pulsos. O cérebro processa informação por meio de pulsos e as SNNs seguem o mesmo princípio (ESHRAGHIAN et al., 2023). Para transformar dados do mundo real em pulsos utilizáveis, empregam-se métodos de *encoding*.

O conjunto de dados MNIST reúne imagens de dígitos manuscritos, com fundo preto e traços brancos (bordas frequentemente em tons de cinza). Por sua simplicidade e difusão na

literatura, será usado como exemplo neste trabalho (DENG, 2012).

A entrada de uma SNN é organizada ao longo de T instantes de tempo (*time steps*), em que cada instante representa uma partição temporal; a soma desses intervalos corresponde à duração total do estímulo. Em cada instante podem ocorrer pulsos (*spikes*), que constituem tanto entradas quanto saídas em SNNs. A maneira como um dado é convertido em pulsos, e como a rede será posteriormente lida (decodificada), é determinada pelo método de *encoding* (e pelo método de *decoding* na saída).

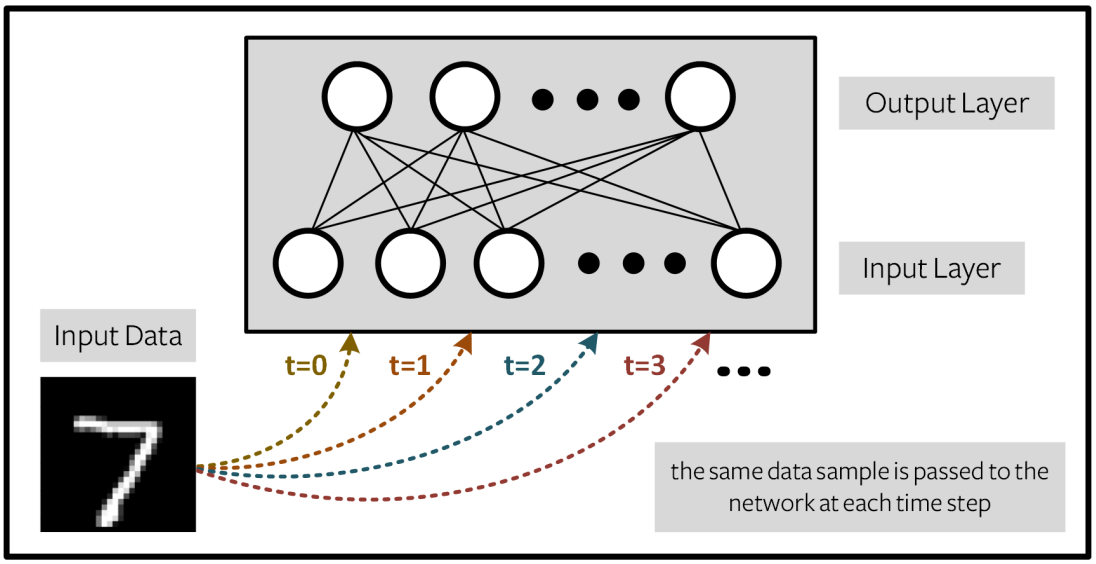
Como alternativa, é possível não aplicar *encoding* pulsado explícito, alimentando a rede com um estímulo constante (por exemplo, corrente/entrada contínua) ao longo de todos os T instantes: isto é, apresentar a mesma matriz de intensidades a cada passo de tempo, como se a imagem fosse um "vídeo estático" (Figura 6). Formalmente, para uma imagem,

$$X \in \mathbb{R}^{m \times n}, X_{ij} \in [0, 1]. \tag{2.5}$$

Em que $X_{ij} = 0$ representa um pixel totalmente preto e $X_{ij} = 1$ representa um pixel totalmente branco. Contudo, essa estratégia não explora plenamente uma das principais vantagens das SNNs que é extrair significado da estrutura temporal.

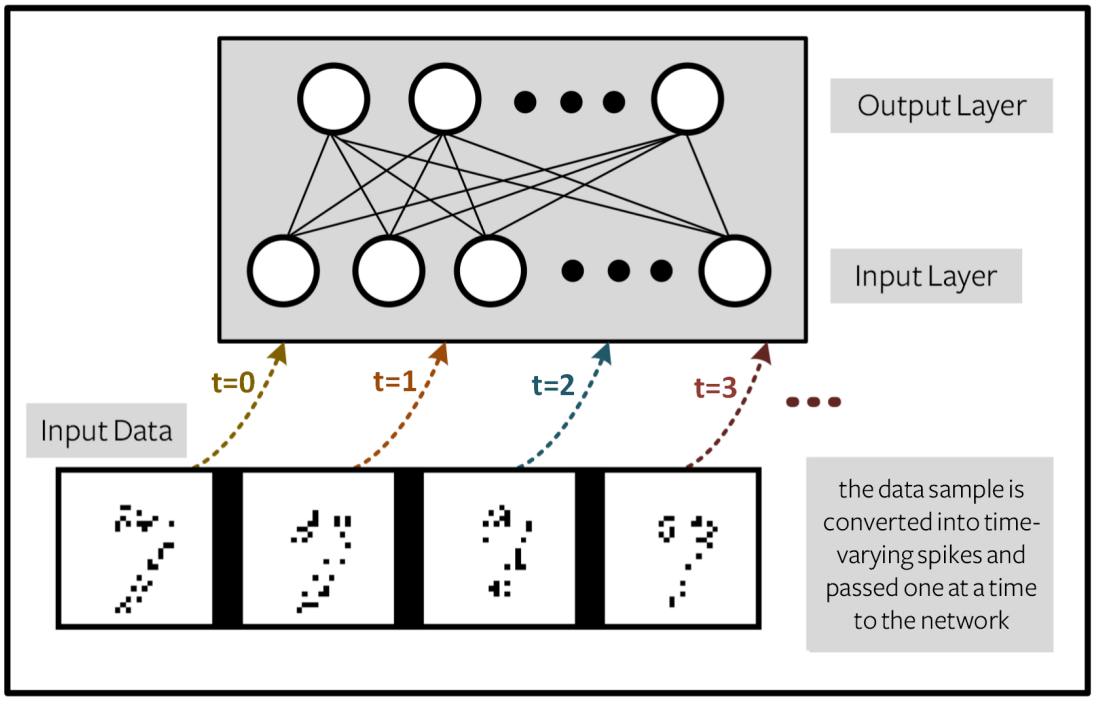
A ideia central do *encoding* é transformar o dado de entrada em uma sequência de pulsos, compatível com os T instantes, em que cada pixel da imagem de entrada é tratado como um valor discreto $X_{ij} \in \{0, 1\}$, ou seja, a imagem MNIST é convertida em uma sequência de pulsos que varia com o tempo e tem relação com a imagem original.

Figura 6 – Exemplo de "vídeo estático" do MNIST



Fonte: <https://github.com/jeshraghian/snntorch/blob/master/docs/_static/img/examples/tutorial1/1_2_1_static.png?raw=true>. Acesso em: 4 out. 2025.

Figura 7 – Exemplo de conversão de MNIST em trens de spikes

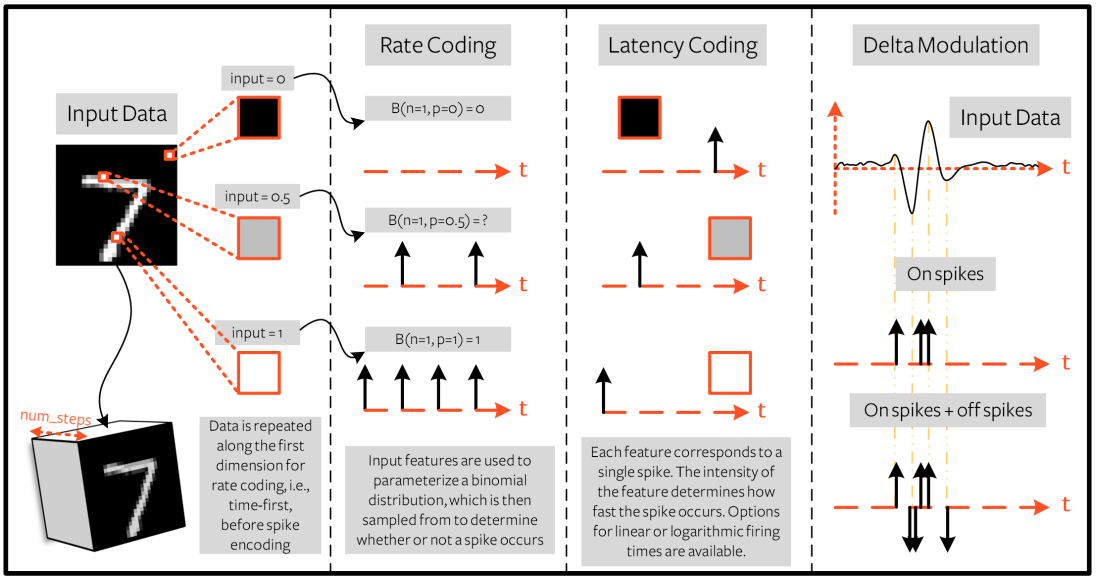


Fonte: <https://github.com/jeshraghian/snntorch/blob/master/docs/_static/img/examples/tutorial1/1_2_1_static.png?raw=true>. Acesso em: 4 out. 2025.

Duas abordagens de *encoding* são particularmente usuais: *rate coding* (Figura 7) e *latency/temporal coding*. No *rate coding*, a intensidade do valor de entrada determina a frequência de disparos, no MNIST pixels escuros (X_{ij} próximo de 0) disparam raramente; pixels claros(X_{ij} próximo de 1) disparam com maior frequência e a saída do neurônio usando *rate coding* é definida pelo o neurônio que tem a maior frequência de pulsos. Já o *latency coding* converte a intensidade dos valores de entrada em um momento/time step especifico que o pulso sera disparado, e a saída do neurônio usando *latency coding* é definida pelo o neurônio que dispara primeiro.

Os dois exemplos são mostrados na Figura 8. No caso do *rate coding*, a intensidade do pixel é convertida em uma distribuição de Bernoulli $R_{ij} \sim B(n, p)$, onde, dado um pixel X_{ij}, sua probabilidade de ocorrer um pulso em *rate coding* é definida da seguinte forma: $P(R_{ij} = 1) = X_{ij} = 1 - P(R_{ij} = 0)$. Já no caso do *latency coding*, a intensidade do pixel define quão cedo o pulso irá ocorrer.

Figura 8 – Comparação visual entre rate e latency coding no MNIST



Fonte: <https://github.com/jeshraghian/snntorch/blob/master/docs/_static/img/examples/tutorial1/1_2_3_spikeconv.png?raw=true>. Acesso em: 4 out. 2025.

2.2.3 STDP *Learning*

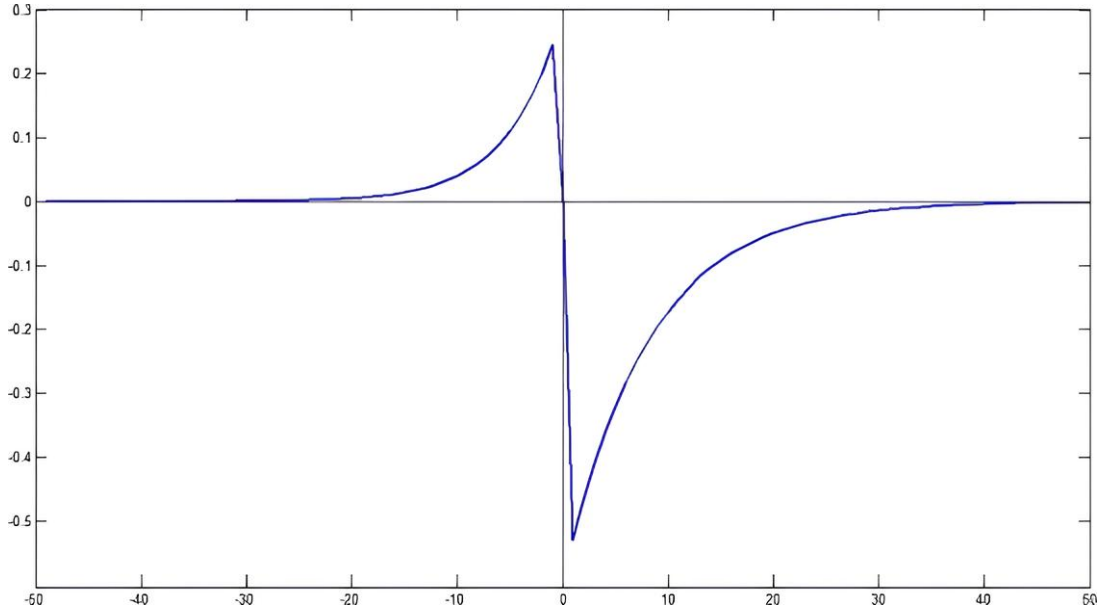
Spike-Timing-Dependent Plasticity (STDP) é um fenômeno descrito por Bi e Poo (BI; POO, 1998). Em experimentos com neurônios do hipocampo de ratos, os autores observaram que, para dois neurônios conectados por uma sinapse, quando o neurônio pré-sináptico e o pós-sináptico disparam em um curto intervalo temporal, o intervalo entre os disparos induz uma modificação na força sináptica. Se define esse intervalo como:

$$\Delta t = t_{\text{post}} - t_{\text{pre}} \tag{2.6}$$

em que Δt é a diferença entre o instante do disparo pós-sináptico (t_{post}) e o instante do disparo pré-sináptico (t_{pre}). O estudo mostrou que, quando $\Delta t > 0$ e aproximadamente $\Delta t \lessapprox 20ms$ ocorre potenciação da sinapse (a força sináptica aumenta). Por outro lado, quando $\Delta t < 0$ e aproximadamente $\Delta t \gtrapprox -20ms$ ocorre depressão sináptica (a força diminui).

Esse fenômeno foi traduzido para ANNs na forma do STDP *learning*, uma regra de aprendizado não supervisionado para SNNs. O algoritmo opera de modo análogo ao observado por Bi e Poo: o peso sináptico artificial varia em função do intervalo temporal Δt entre o disparo pré- e pós-sináptico. A intensidade dessa variação é dada por uma função janela, em que o eixo x representa Δt e o eixo y representa a magnitude (e o sinal) da mudança do peso. Um exemplo dessa função é apresentado na Figura 9.

Figura 9 – Exemplo de função janela de STDP learning



Fonte: (IAKYMCHUK et al., 2015)

Quando se define a função de "reforço" no STDP *learning* (mais precisamente, a função janela de STDP), a curva é normalmente assimétrica, como ilustrado na Figura 9, sua forma pode ser escrita por:

$$STDP(\Delta t) = \Delta w = \begin{cases} A_+ \ \exp^*\left(\dfrac{\Delta t}{\tau_+}\right), & se \sim\sim \Delta t \geq 2, \\ A_- \ \exp^*\left(\dfrac{\Delta t}{\tau_-}\right), & se \sim\sim \Delta t \leq -2, \\ 0, & se \sim\sim -2 < \Delta t < 2 = 0, \end{cases} \tag{2.7}$$

Onde $\Delta t = t_{\text{post}} - t_{\text{pre}}$ é o intervalo entre os pulsos do neurônio pós-sináptico e do neurônio pré-sináptico; A_- e A_+ são constantes (ganhos) que controlam, respectivamente, a depressão (janela negativa) e a potenciação (janela positiva) máximas; e τ_- e τ_+ são constantes de tempo que caracterizam a inclinação/largura da janela em cada lado. A variação de peso na sinapse é definida por:

$$w_{new} = \begin{cases} w_{old} + \sigma \Delta w (w_{max} - w_{old}), & se \Delta w > 0 \\ w_{new} + \sigma \Delta w (w_{old} - w_{min}), & se \Delta w \leq 0 \end{cases} \tag{2.8}$$

em que σ é a taxa de aprendizado, e os pesos são limitados por $w_{max} \geq w \geq w_{min}$ (IAKYMCHUK et al., 2015).

Existem técnicas adicionais que podem ser acopladas ao STDP *learning* para adaptar a regra ao contexto da tarefa, mas com ou sem tais extensões, o STDP preserva sua principal vantagem: aprender de forma não supervisionada a partir da correlação temporal entre *spikes*. Por outro lado, sua principal limitação é a dificuldade de escalar para arquiteturas profundas

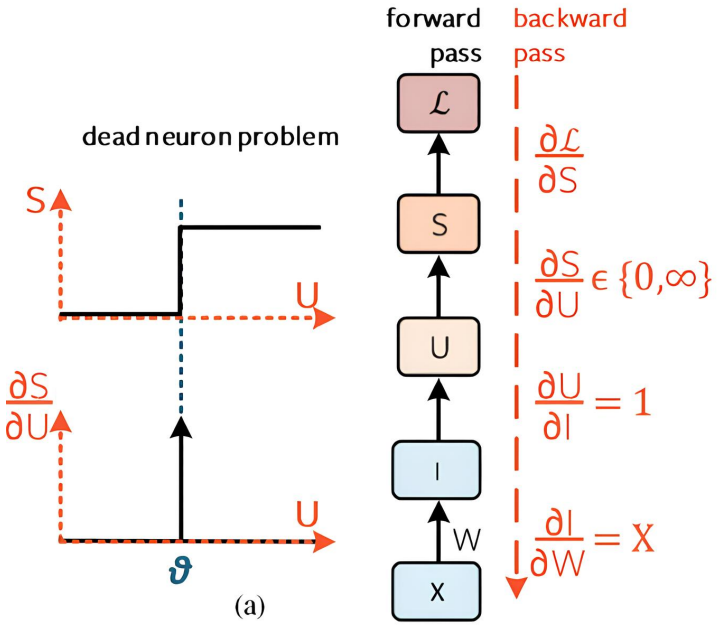
sem mecanismos complementares, sendo mais comum o uso eficiente em redes rasas ou em camadas iniciais.

2.2.4 *Backpropagation* em SNNs

Um dos métodos mais utilizados para treinar SNNs é o *backpropagation* adaptado para esse tipo de rede. Em termos gerais, o *backpropagation* calcula o gradiente da função de perda para cada parâmetro aprendível por meio da regra da cadeia, propagando o erro a partir da última camada até as anteriores. Em seguida, usa-se esse gradiente para atualizar os pesos, ajustando-os na direção de redução da perda. Se o gradiente for 0, não há atualização daquele peso. Nas SNNs, um dos principais obstáculos é a não diferenciabilidade dos pulsos, o que dificulta o uso direto do *backpropagation*. Esse problema da não diferenciabilidade dos pulsos é também conhecida como o problema do "neurônio morto".

Para entender melhor a não diferenciabilidade do disparo no *backpropagation* aplicado a um *spiking neuron* com LIF, considere que a atualização ΔW é aplicada ao peso W. Essa atualização causa o potencial da membrana mudar por ΔU, mas essa mudança no potencial falha em precipitar uma mudança para a presença de pulso do neurônio, em outras palavras, $dS/dU = 0$ para todo U, com exceção do limiar θ, onde $dS/dU \to \infty$. Com isso, o gradiente do pulso pelo potencial da membrana tende a ser nulo ou infinito, tornando a diferenciação inviável por métodos padrão (ESHRAGHIAN et al., 2023). O problema do "neurônio morto" é mostrado visualmente na Figura 10.

Figura 10 – Problema do "neurônio morto"



Fonte: (ESHRAGHIAN et al., 2023)

Uma alternativa para contornar o problema do "neurônio morto" é o método de

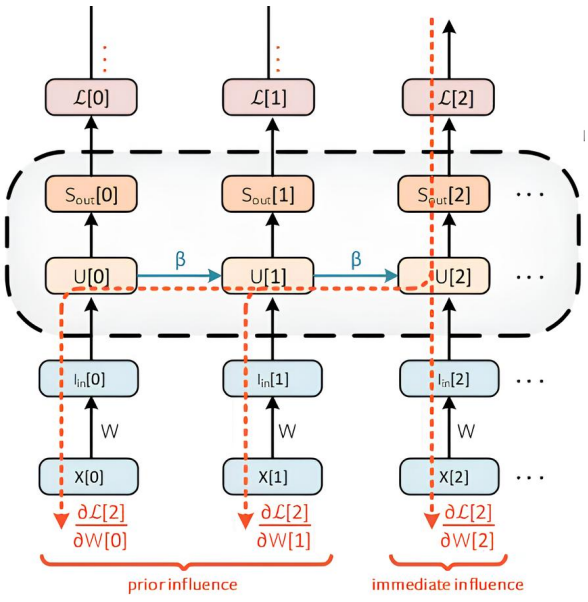
backpropagation using spike times, que foi uma das primeiras propostas para treinar SNNs de múltiplas camadas com *backpropagation*. A intuição é que, embora os pulsos sejam descontínuos, o tempo é contínuo, logo, calcular derivadas dos instantes de disparo em relação aos pesos pode fornecer gradientes úteis (BOHTE; KOK; La Poutré, 2002). Na prática, entretanto, o método mostrou-se sensível a ruídos e a pequenas variações temporais, além de exigir hipóteses restritivas sobre o padrão de pulsos, motivo pelo qual hoje é pouco utilizado.

A alternativa de *backpropagation* mais utilizada em SNNs é a *backpropagation through time* (BPTT). Nesse método, registram-se, para cada *time step* t (Exemplificado na Figura 11 quando $t = 2$), os estados relevantes da rede e entradas/saídas correspondentes. Em seguida, calcula-se o gradiente da perda total em relação aos parâmetros treináveis ($\partial L/\partial W$). Achar o gradiente $\partial L/\partial W$ nos permite o uso de *gradient descent* para treinar a rede.

Considere que o parâmetro W é aplicado em todo *time step*, e a aplicação do peso em um dado *time step* s é dado por $W[s]$. Assuma que a perda instantânea $L[t]$ possa ser calculada em todo *time step* t. Como cada peso afeta a perda presente e as futuras, a contribuição de $W[s]$ em $L[t]$ quando $s = t$ é chamada de influência imediata. Já para $s < t$ o impacto de $W[s]$ em $L[t]$ é chamado de influência anterior. A influência de todos os parâmetros aplicados em perdas presente e futuras são somadas para definir o gradiente global (ESHRAGHIAN et al., 2023) que é definido por:

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_t \frac{\partial \mathcal{L}[t]}{\partial W} = \sum_t \sum_{s \leq t} \frac{\partial \mathcal{L}[t]}{\partial W[s]} \frac{\partial W[s]}{\partial W}. \tag{2.9}$$

Figura 11 – Visualização do cálculo de $\partial L[t]/\partial W$ quando $t = 2$



(a)

Fonte: (ESHRAGHIAN et al., 2023)

Contudo, o problema da não diferenciabilidade dos pulsos permanece quando se aplica BPTT em sua forma "pura", pois $\partial S/\partial U \in \{0, \infty\}$. Para contornar esse impasse, utiliza-se a técnica do gradiente substituto ou *surrogate gradient*. Nela, no *forward* aplica-se o operador de *Heaviside* a $U[t]$ para determinar se o neurônio dispara. Já durante a *backpropagation*, substitui-se o *Heaviside* por uma função contínua $\tilde{S}$(i.e. sigmoid), e usa-se sua derivada como substituta do gradiente: $\partial S/\partial U \leftarrow \partial \tilde{S}/\partial U$ (ESHRAGHIAN et al., 2023).

2.2.5 Conversão ANN-SNN

Uma maneira de contornar problemas de treinamento direto por *backpropagation* em SNNs a não diferenciabilidade do treino de pulsos é utilizar a conversão ANN→SNN, na qual a função de ativação de alta precisão de cada neurônio de uma ANN é mapeada para um esquema de codificação, tipicamente *rate coding* e, em menor grau, *latency coding*. Uma das principais vantagens dessa conversão é que, para classificação de imagens estáticas em conjuntos de dados complexos, costuma-se obter desempenho próximo ou superior a outros métodos específicos de treino de SNNs, com a economia energética concentrada na inferência. Isso ocorre porque o treinamento é realizado no domínio ANN, preservando sua eficiência de otimização, enquanto a execução final se dá como SNN, explorando disparos esparsos e processamento orientado a eventos, a o custo de manter a ineficiência energética do treinamento de uma ANN.

Além de a economia ocorrer principalmente na inferência, a conversão ANN→SNN traz limitações. Em primeiro lugar, sua vantagem é mais pronunciada em tarefas estáticas, quando o problema exige dinâmicas temporais ricas, o método não aproveita todo o potencial de SNNs. Em segundo lugar, a conversão costuma demandar muitos *time steps* para aproximar taxas de disparo com boa precisão, o que aumenta latência e pode reduzir parte do ganho de eficiência. Em terceiro lugar, a conversão é sempre uma aproximação: diferenças na normalização de camadas, o que reduz a acurácia em relação à ANN original. Por fim, a conversão de modelos sequenciais para SNNs ainda é menos explorada, tornando a aplicação direta em tarefas fortemente temporais um desafio aberto (ESHRAGHIAN et al., 2023).

Abordagens de conversão ANN→SNN normalmente partem de uma ANN totalmente treinada para então convertê-la em SNN. Entretanto, existem métodos que impõem restrições durante o treinamento da própria ANN i.e. restringir e depois treinar, de modo a facilitar a conversão tendo em mente propriedades das SNNs e, por vezes, características do hardware-alvo. Embora ambos serem métodos de conversão, diferem no momento e na forma como as restrições são introduzidas, na conversão clássica (*train-then-convert*), treina-se a ANN sem restrições específicas e só depois se realiza o mapeamento para SNN. Já nos métodos restringir-e-treinar (*constrain-then-train*), a ANN é treinada desde o início sob restrições, com hiperparâmetros e limites desejados para a SNN final já definidos no processo de treinamento. Como consequência, se esses parâmetros mudarem após o treinamento, geralmente é necessário retreinar a ANN.

Em termos de desempenho, métodos restringir-e-treinar frequentemente apresentam acurácia superior à conversão direta em tarefas de classificação estática, pois reduzem o descompasso entre o comportamento contínuo da ANN e a dinâmica de disparos da SNN. O custo é um treinamento mais complexo (PFEIFFER; PFEIL, 2018).

3 Metodologia

3.1 Ambiente e Ferramentas

Os modelos de *machine learning* foram desenvolvidos na plataforma Kaggle, em *notebooks* Jupyter escritos na linguagem de programação Python. Diversas bibliotecas especializadas foram utilizadas para manipulação, visualização e cálculo de dados, bem como para o desenvolvimento e a avaliação dos modelos de *machine learning*.

3.1.1 Kaggle

O Kaggle é uma plataforma de ciência de dados que oferece um ambiente gerenciado para *notebooks* Jupyter, armazenamento de dados, versionamento e aceleradores de hardware, facilitando a execução e a reprodutibilidade de experimentos. No projeto, o Kaggle foi utilizado para executar os modelos sem configuração local complexa, permitindo redes mais profundas e uma velocidade de execução melhor. Durante o treinamento e a avaliação, foi habilitado o acelerador de GPU NVIDIA P100, disponibilizado pela plataforma.

3.1.2 Python

A linguagem Python foi escolhida devido à sua facilidade de uso e aprendizado e ao ecossistema de bibliotecas para ciência de dados e *machine learning*. Além disso, o suporte nativo a *notebooks* Jupyter facilita a rastreabilidade de experimentos e a reprodutibilidade, enquanto bibliotecas otimizadas como NumPy e PyTorch permitem execução acelerada em GPU, quando disponível. Abaixo serão citadas as bibliotecas utilizadas e seus usos no projeto.

3.1.2.1 Numpy

O NumPy foi empregado como base para operações numéricas vetorizadas e manipulação de arranjos durante pré-análise e avaliação. Ela aparece na criação/concatenação de vetores, cálculo de estatísticas. Também é utilizado na conversão entre tensores do PyTorch e ndarray para métricas e *plots*.

3.1.2.2 Pandas

O Pandas foi utilizado para organizar metadados do conjunto de imagens em DataFrames, facilitando contagens estratificadas por classe, tabelas de resoluções e sumarizações descritivas que alimentam gráficos. Funções auxiliares constroem um DataFrame a partir da lista de

amostras do *dataset*, permitindo calcular frequências e gerar estatísticas que subsidiam a criação de gráficos.

3.1.2.3 Matplotlib

A Matplotlib foi responsável pela criação de gráficos de evolução de treino/validação (em métricas de *loss*, acurácia e F1-macro) e matrizes de confusão. Esses gráficos foram usados para inspecionar *overfitting/underfitting*, estabilidade de treino e padrões de erro por classe.

3.1.2.4 Scikit-Learn

Scikit-Learn foi utilizado para realizar a divisão estratificada de train/val com a função train_test_split para manter a proporção de classes. Ele também foi utilizado para o cálculo de métricas clássicas de classificação e da matriz de confusão.

3.1.2.5 PyTorch

O PyTorch fornece a infraestrutura principal para aprendizado de máquina em Python, incluindo o tipo de dado tensor, o mecanismo de autograd para cálculo automático de gradientes, e a API de modelagem para compor camadas e redes neurais. Além disso, oferece utilitários de treinamento como otimizadores e funções de perda, carregamento de dados, e suporte a aceleração por GPU. Em síntese, ele é a base que orquestra dados, modelos e otimização de forma eficiente e flexível.

3.1.2.6 SnnTorch

A snnTorch estende o ecossistema do PyTorch para SNNs, disponibilizando neurônios e sinapses com dinâmica temporal como LIF, codificações em *spikes*, e ferramentas para treinamento com *surrogate gradients*. Ela permite montar SNNs usando a mesma filosofia modular do PyTorch, integrando-se aos tensores, autograd, otimizadores e DataLoader. Em termos gerais, a snnTorch viabiliza, dentro do mesmo pipeline do PyTorch, a modelagem e o treinamento de SNNs.

3.1.2.7 Zeus

O Zeus foi integrado ao código da CNN para quantificar tempo e energia consumidos pela GPU durante o treinamento e durante a fase de testes.

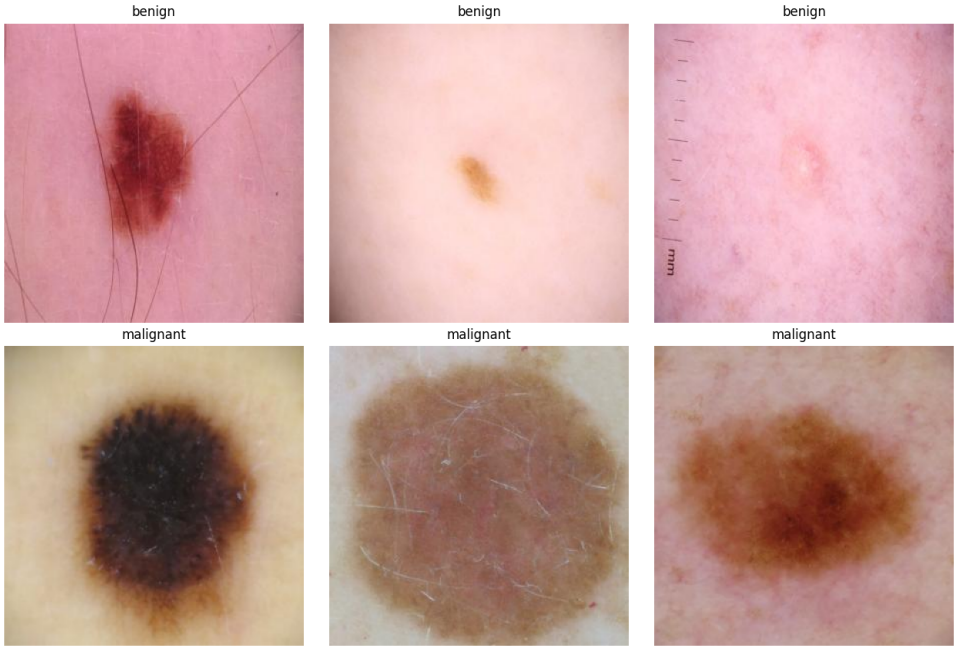
3.2 *Dataset*

A base de dados utilizada para o treinamento e a avaliação dos modelos foi criada a partir de diferentes conjuntos da ISIC (*International Skin Imaging Collaboration*) e disponibilizada

publicamente na plataforma Kaggle (JAVID, 2022). Cada imagem representa uma lesão de pele classificada como benigna ou maligna. O *dataset* é dividido em treino e teste. O conjunto de treino contém 5.000 imagens benignas e 4.605 imagens malignas. O conjunto de teste contém 500 imagens benignas e 500 imagens malignas.

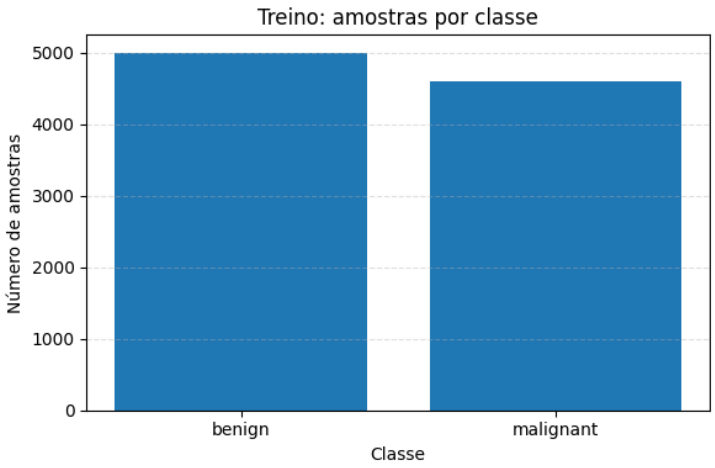
Todas as imagens do conjunto de dados têm resolução de 300×300 pixels. A Figura 12 apresenta três exemplos de cada classe, nessas imagens nenhum filtro ou transformação foi aplicado. A Figura 13 mostra a distribuição de classes no conjunto de treino, e a Figura 14, a distribuição no conjunto de teste.

Figura 12 – Amostra de dados do dataset



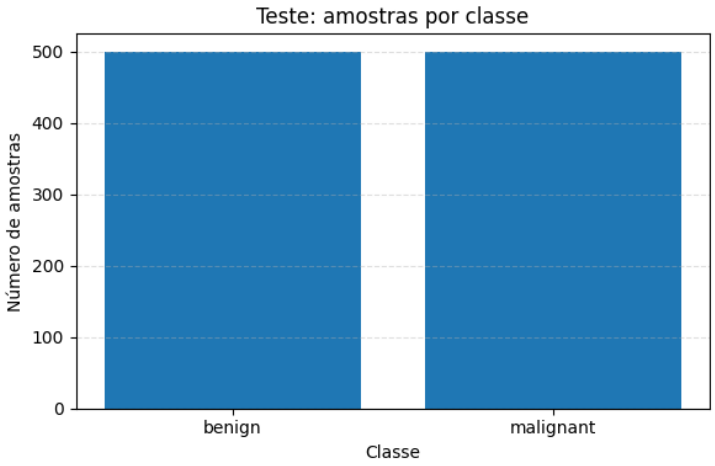
Fonte: Elaborada pelo Autor.

Figura 13 – Distribuição de classes no conjunto de treino



Fonte: Elaborada pelo Autor.

Figura 14 – Distribuição de classes no conjunto de teste



Fonte: Elaborada pelo Autor.

3.3 Pré-Processamento

Por simplicidade, as mesmas transformações foram aplicadas aos dois modelos. Nos conjuntos de treino e de teste, realizou-se o redimensionamento das imagens para uma altura e largura de 224×224 pixels. Para facilitar o treinamento da SNN, as imagens foram transformadas para escala de cinza e seus valores normalizados para o intervalo $[0, 1]$.

A escolha de 224×224 decorre do fato de que ambas as arquiteturas foram inspiradas na AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017), cuja configuração de entrada utiliza dimensões em torno de 224×224.

3.4 Métricas

Legenda:

- $TP =$ Número de verdadeiros positivos.
- $FP =$ Número de falsos positivos.
- $TN =$ Número de verdadeiros negativos.
- $FN =$ Número de falsos negativos.

Acurácia: Mede a proporção de classificações corretas dentre todas as previsões realizadas. É calculada somando todos os acertos e dividindo pelo total de amostras avaliadas

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

Precisão: Quantifica a exatidão das previsões positivas i.e. dentre tudo que o modelo marcou como positivo, qual fração corresponde a casos realmente positivos

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.2}$$

Recall: Mede a cobertura dos casos positivos i.e. dentre todos os exemplos que são positivos no conjunto de dados, qual fração é identificada como tal pelo modelo.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.3}$$

F1-Score: Obtém um único valor a partir da precisão e do recall por meio da média harmônica, atribuindo um balanço entre os dois componentes.

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

F1 Macro: Calcula-se a precisão individual de cada classe k, e depois se toma a média aritmética simples entre as classes. Cada classe contribui igualmente para o resultado, independentemente de sua frequência no conjunto de dados.

$$\text{F1}_k = \frac{2 \cdot \text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k} \quad \Rightarrow \quad \text{F1}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^{K} \text{F1}_k \tag{3.5}$$

Matriz de confusão: A matriz de confusão organiza, em forma de tabela, as contagens de acertos e erros de um classificador ao comparar as classes reais (linhas) com as classes preditas (colunas). Cada célula C_{ij} registra quantas amostras da classe real i foram previstas como classe j. Através da matriz de confusão, é possível obter as variáveis para o cálculo da acurácia, precisão e *Recall*.

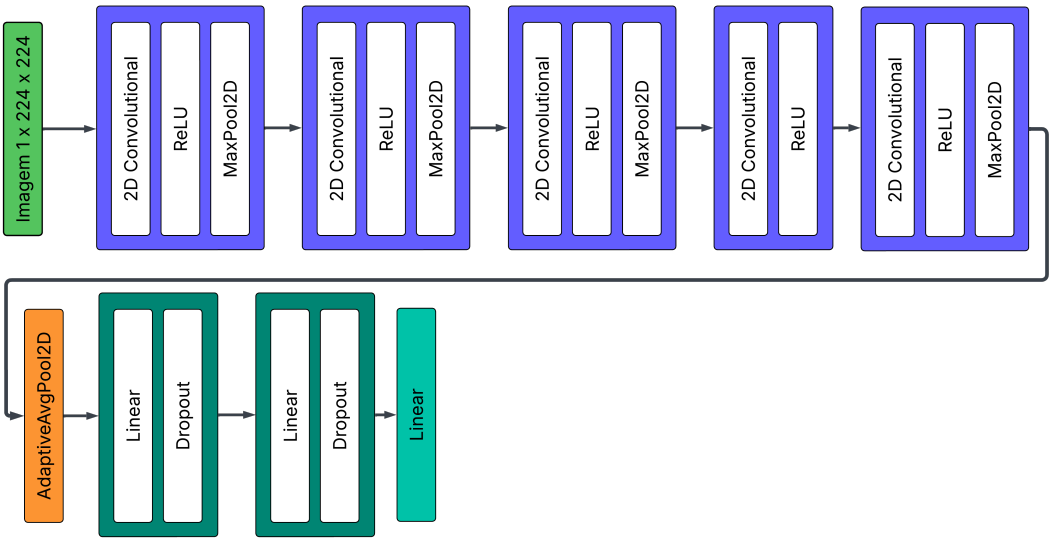
3.5 Modelos Propostos

3.5.1 CNN

A arquitetura foi inspirada na AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017), pela sua boa performance em tarefas de classificação. Pequenas diferenças em relação ao modelo original foram introduzidas por motivos de desempenho (i.e. o número de *input channels* e *output channels* das camadas e algumas camadas lineares foram removidas) e para melhor adequação à tarefa. O *pipeline* de camadas do modelo é apresentado na Figura 15.

Durante o treinamento, utilizou-se um *batch* de 128 elementos, o otimizador Adam com taxa de aprendizado de 1×10^{-4} e decaimento de peso de 1×10^{-5}. A função de perda adotada foi a *Cross-Entropy Loss*. O conjunto de treinamento foi dividido em treino e validação nas proporções de 70% e 30% respectivamente, com *seed* igual a 123.

Figura 15 – Arquitetura CNN



Fonte: Elaborada pelo Autor.

3.5.2 SNN

A arquitetura da SNN foi construída de maneira análoga à da CNN, substituindo-se as ativações por camadas LIF, a arquitetura completa é vista em maior detalhe na Figura 16. As camadas LIF foram configuradas com $\beta = 0.8$.

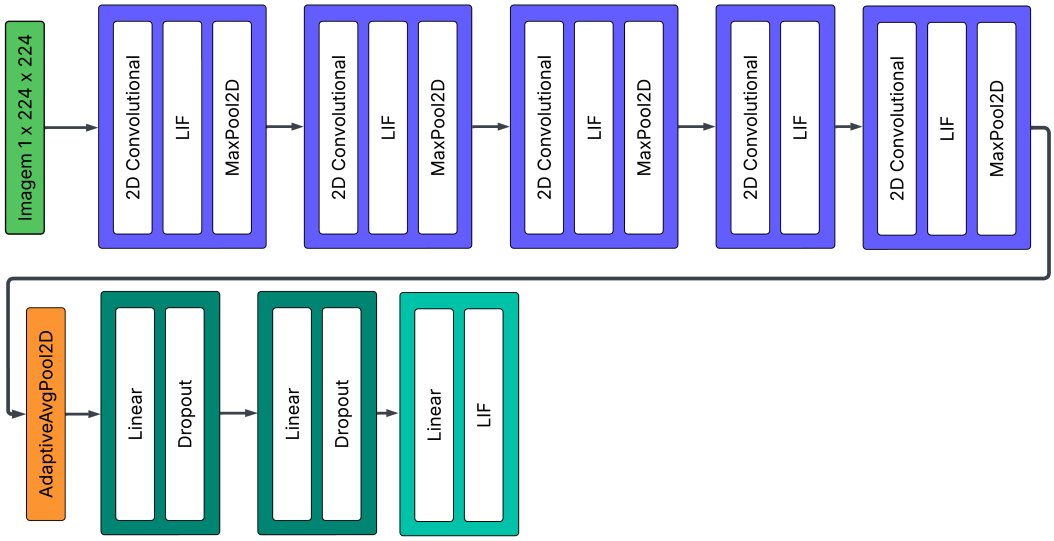
Durante o treinamento, utilizou-se um *batch* de 128 elementos, 10 *time steps*, o otimizador Adam com taxa de aprendizado de 1×10^{-4}. A função de perda adotada foi a *Cross-Entropy Loss*. O conjunto de treinamento foi dividido em treino e validação nas proporções de 70% e 30% respectivamente, com *seed* igual a 123.

Em cada chamada ao *forward*, todo o pipeline de camadas é executado *time_step*

vezes, e em cada iteração deste *loop* são armazenados em *arrays* os estados da última camada (*spikes* e potencial de membrana). Ao final, esses estados são retornados e utilizados no cálculo das métricas e.g. *loss* e acurácia. Para impedir erros as camadas LIF são reinicializadas no começo de toda chamada da função *forward*.

Como o *dataset* é composto por imagens estáticas e a dinâmica temporal não é essencial para esta tarefa (além do benefício energético), não foi definido um método de *encoding* explícito, a entrada é apresentada como um "vídeo estático", o que, na prática, equivale a um *rate coding* trivial. Um exemplo desse desse procedimento é mostrado na Figura 6 da seção 2.2.2. O método de treinamento adotado foi o *backpropagation* em SNNs', por alinhar-se à tarefa, pela complexidade do método, e pelo o menor gasto energético no treinamento e no seu uso(se for utilizado *hardware* neuromórfico).

Figura 16 – Arquitetura SNN



Fonte: Elaborada pelo Autor.

4 Resultados e Discussão

4.1 Treinamento CNN

O treinamento teve duração de 20 épocas. Esse número foi escolhido porque, em testes preliminares, aumentar o total de épocas elevou o risco de *overfitting*. Seria possível treinar por mais tempo mediante redução da taxa de aprendizado ou adoção de uma taxa adaptativa, porém isso aumentaria significativamente o tempo de treinamento, com ganhos marginais nas métricas.

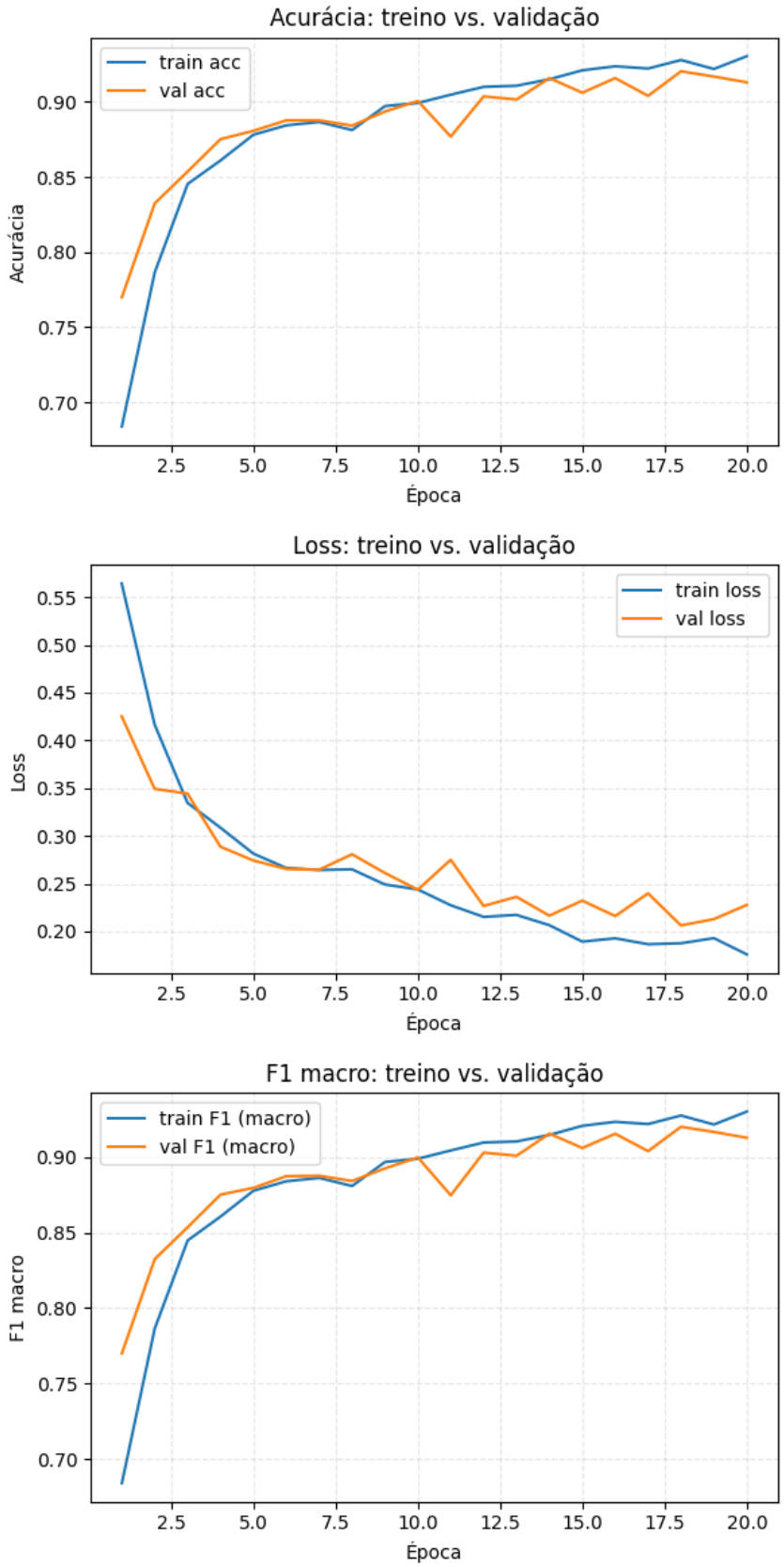
A versão final do modelo foi selecionada com base no maior valor de F1-macro no conjunto de validação. Esse critério foi adotado porque, em tarefas de diagnóstico médico, o F1-macro tende a refletir melhor a qualidade do modelo em cenários com um leve desbalanceamento entre classes.

Durante o período de treinamento e também no teste, utilizou-se o Zeus para mensurar o gasto energético da GPU. O consumo em joules e a duração de cada período estão apresentados na Tabela 1.

Nas cinco primeiras épocas, o modelo apresentou melhora acentuada em todas as métricas, à medida que o número de épocas avançou, o ganho tornou-se mais lento. Ainda assim, observou-se evolução consistente, com pequenas flutuações em algumas épocas, como mostrado nos gráficos das Figuras 17.

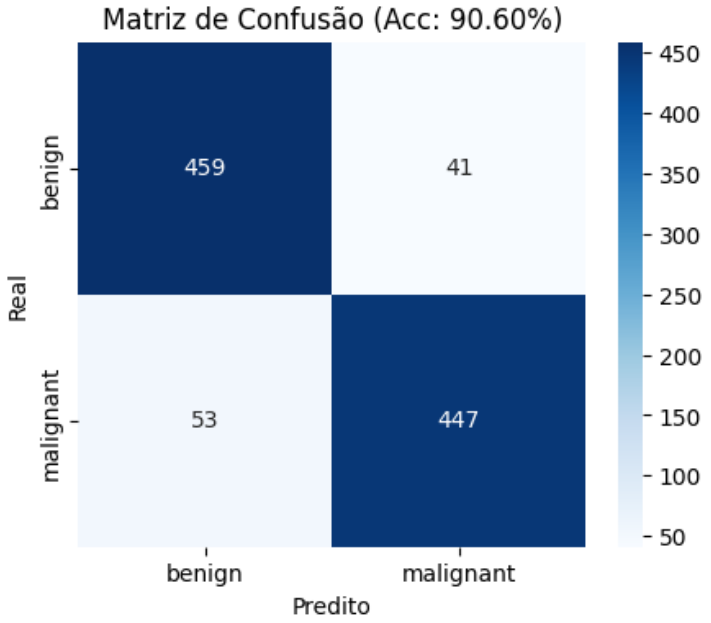
No conjunto de teste, os resultados foram positivos. Como ilustram as Figuras 18 e 19, obteve-se acurácia de 90,6% e valores em torno de 0,9 para precisão e *recall*, indicando que o modelo final é equilibrado.

Figura 17 – Evolução das métricas da CNN ao longo de 20 épocas



Fonte: Elaborada pelo Autor.

Figura 18 – Matriz de confusão da CNN no conjunto teste



Fonte: Elaborada pelo Autor.

Figura 19 – Resultados da CNN no conjunto teste

Classification Report:

	precision	recall	f1-score	support
benign	0.90	0.92	0.91	500
malignant	0.92	0.89	0.90	500
accuracy			0.91	1000
macro avg	0.91	0.91	0.91	1000
weighted avg	0.91	0.91	0.91	1000

Fonte: Elaborada pelo Autor.

4.2 Treinamento SNN

De maneira análoga à CNN, o treinamento da SNN teve 20 épocas, e o melhor modelo foi selecionado com base no F1-macro do conjunto de validação, os motivos para essas escolhas são os mesmos adotados na CNN.

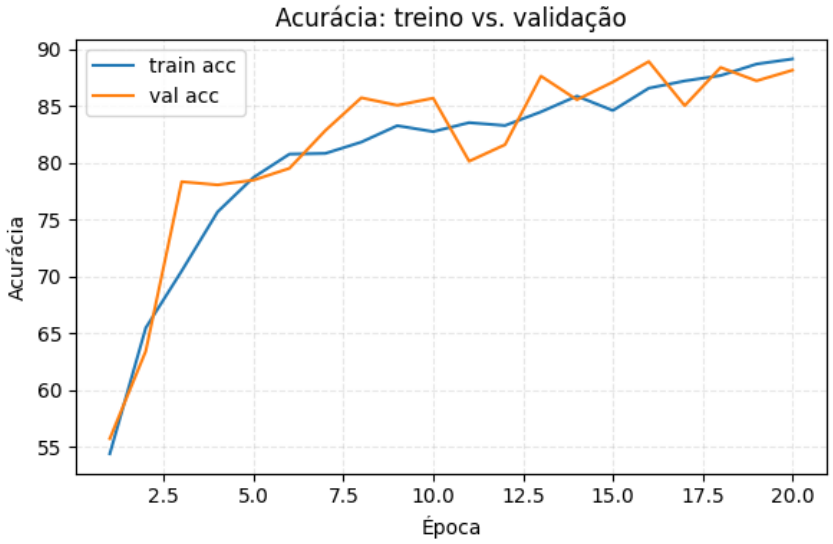
Nas sete primeiras épocas, observou-se rápida melhora em todas as métricas. Nas épocas seguintes, os ganhos tornaram-se mais lentos, ainda que significativos. Os gráficos de validação (Figura 20) não foram tão "suaves" (apresentaram picos e maior variação), esse comportamento poderia ser atenuado aumentando o número de *time steps*. Contudo, essa

opção não foi adotada, pois um *batch* de 128 com 10 *time steps* já aumenta substancialmente o tempo de treinamento quando o processo ocorre fora de hardware neuromórfico.

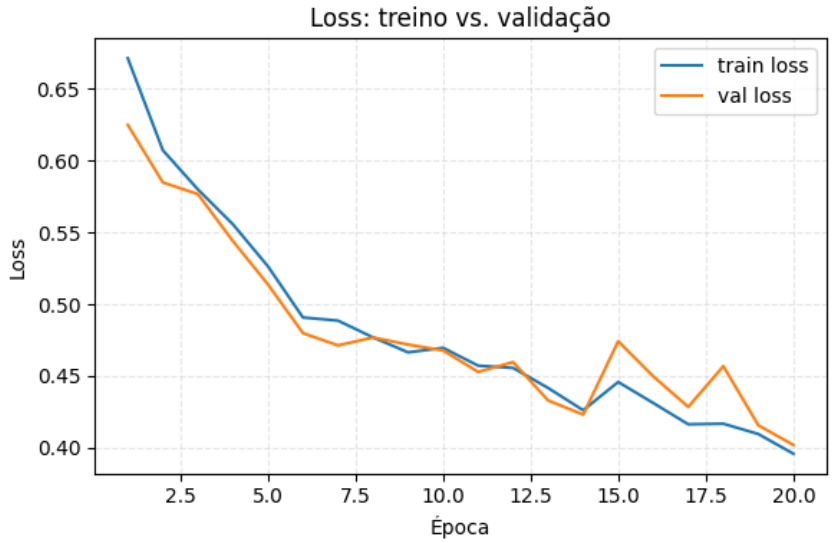
O Zeus não foi utilizado para medir o gasto energético da SNN durante o treinamento e o teste, pois a eficiência energética característica das SNNs manifesta-se principalmente em hardware neuromórfico. Como os experimentos foram executados em hardware convencional, a medição não refletiria o custo energético real. Uma alternativa seria estimar o consumo (LEMAIRE et al., 2022) mas essa estimativa é complexa e depende de suposições adicionais.

As métricas finais ficaram muito próximas às da CNN, com redução média de aproximadamente 1 ponto percentual em quase todas as métricas (Figuras 21 e 22). A SNN apresentou leve viés para a classe benigna, possivelmente devido à sua maioria no conjunto de treino, mas o efeito não foi significativo.

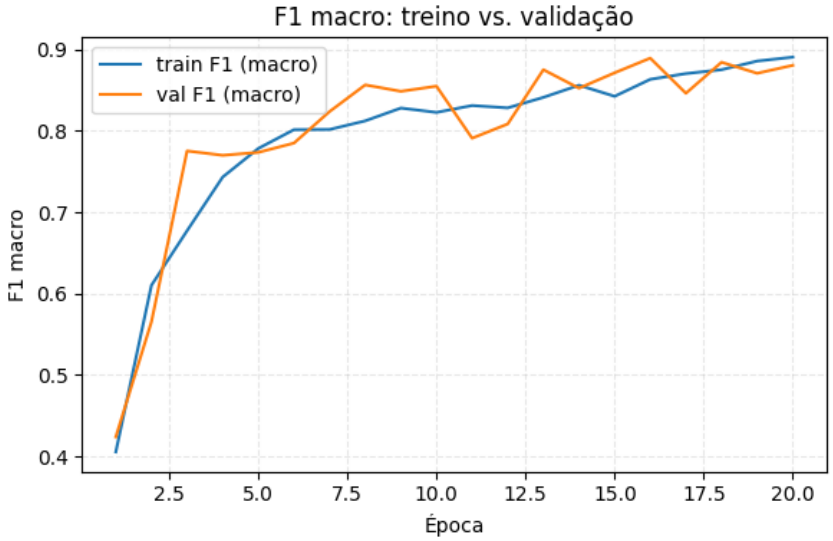
Figura 20 – Evolução das métricas da SNN ao longo de 20 épocas



(a) Acurácia



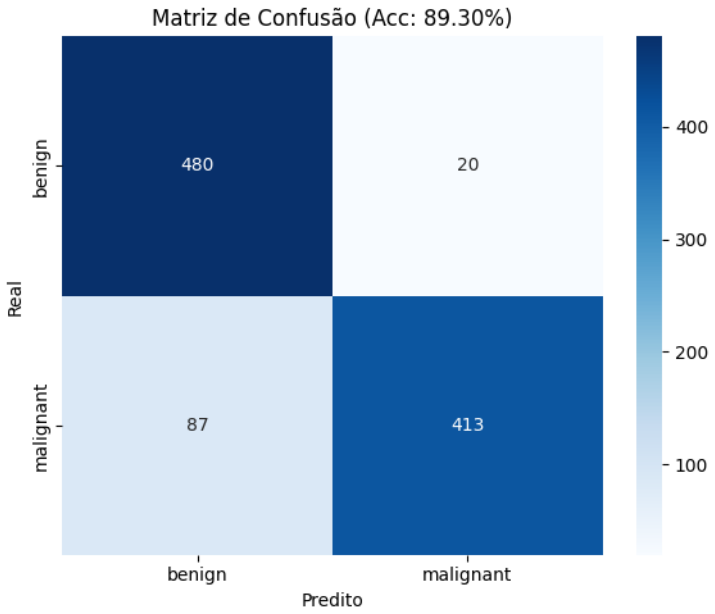
(b) Loss



(c) F1-Macro

Fonte: Elaborada pelo Autor.

Figura 21 – Matriz de confusão da SNN no conjunto teste



Fonte: Elaborada pelo Autor.

Figura 22 – Resultados da SNN no conjunto teste

Classification Report:

	precision	recall	f1-score	support
benign	0.85	0.96	0.90	500
malignant	0.95	0.83	0.89	500
accuracy			0.89	1000
macro avg	0.90	0.89	0.89	1000
weighted avg	0.90	0.89	0.89	1000

Fonte: Elaborada pelo Autor.

4.3 Análise dos Resultados

Tabela 1 – Resultados dos modelos no conjunto de teste.

	Accuracy	Precision	Recall	F1-Score	Energy per Epoch	Energy Consumption on Training	Energy Consumption on Testing
CNN	90.6%	0.918	0.896	0.9071	1 260 J	25 300 J	111 J
SNN	89.3%	0.960	0.8466	0.8997	–	–	–

Fonte: Elaborada pelo Autor.

Como esperado, as métricas da SNN ficaram levemente abaixo das da CNN, com diferença de acurácia de 1,3%. Em questão de perspectivas futuras para tarefas de classificação

de imagens estáticas, a conversão ANN–SNN pode reduzir ainda mais essa diferença. Se for adotado um esquema *constrain-then-train* na ANN antes da conversão, a diferença tende a diminuir ainda mais, pois a rede já é treinada com restrições compatíveis com a SNN.

Embora a métrica de consumo energético da SNN não tenha sido medida diretamente neste estudo, toma-se como referência os resultados de (LEMAIRE et al., 2022), que comparam uma CNN a uma SNN equivalente em uma tarefa de classificação de imagens estáticas (CIFAR-10) e reportam até $8\times$ maior eficiência energética para a SNN. Como o arranjo aqui proposto opera de modo semelhante, é razoável inferir que uma economia significativa de energia seria obtida ao empregar a SNN equivalente, especialmente quando executada em hardware neuromórfico. Ademais, a literatura tipicamente aponta que SNNs, quando realizadas em plataformas neuromórficas, tendem a consumir menos energia do que ANNs em tarefas comparáveis (YAMAZAKI et al., 2022). À luz desses resultados, o *trade-off* entre uma pequena perda de acurácia e um ganho potencial de eficiência energética mostra-se favorável quando a tarefa demanda economia de energia e há acesso ao referido hardware.

Por outro lado, persiste o desafio da complexidade no projeto e treinamento de SNNs. A construção dessas redes requer conhecimento técnico específico, e, por se tratar de uma tecnologia relativamente recente, nem todo avanço obtido em ANNs é imediatamente aplicável em SNNs. Isso pode ampliar diferenças de desempenho (e outras métricas) entre os dois paradigmas se não houver adaptações metodológicas adequadas.

Em suma, as SNNs podem se mostrar viáveis para o diagnóstico de câncer de pele quando há acesso a hardware neuromórfico e a eficiência energética é prioridade e. g. em locais remotos com infraestrutura limitada ou intermitência de energia, nos quais dispositivos móveis com suporte neuromórfico podem viabilizar a aplicação. Embora exista perda de acurácia, ela pode se aproximar da obtida por uma ANN equivalente, o que torna o uso de SNNs atraente nesses cenários.

5 Conclusão

Este trabalho investigou a viabilidade de SNNs em dispositivos móveis para diagnóstico de câncer de pele, contrastando seu desempenho e características com uma rede neural convencional (CNN). O objetivo principal foi comparar, de forma controlada, um arranjo SNN com um arranjo ANN/CNN desenvolvido para a mesma tarefa, considerando não apenas métricas de acurácia, mas também aspectos práticos como custo energético e implicações de implementação.

Empregou-se um conjunto de dados derivado da ISIC, disponibilizado no Kaggle, contendo imagens de lesões benignas e malignas, estratificadas em treino (5.000 benignas e 4.605 malignas) e teste (500/500). Todas as imagens possuem 300×300 pixels, tendo sido redimensionadas para 224×224; adicionalmente, aplicou-se conversão para tons de cinza e normalização em [0,1].

A CNN, inspirada na AlexNet, foi treinada por 20 épocas, com seleção do melhor ponto via F1-macro em validação. Seu desempenho no teste atingiu acurácia de 90,6%, com precisão e *recall* em torno de 0,9. Para a mensuração energética, integrou-se o Zeus ao código da CNN, registrando consumo na fase de treino e de teste. A SNN espelhou a arquitetura da CNN, substituindo ativações por camadas LIF ($\beta = 0,8$), executadas ao longo de 10 time steps por chamada ao forward, adotou-se *backpropagation* com *surrogate gradients*. No teste, a SNN obteve 89,3% de acurácia e F1 de 0,8997, configurando diferença de $\approx 1,3$ p.p. em relação à CNN.

No tocante à eficiência energética, não se mediu diretamente a SNN neste estudo, uma vez que o ganho típico emerge em *hardware* neuromórfico, condição não presente no ambiente experimental. Ainda assim, com base na literatura citada, existe uma expectativa de economia de energia quando a SNN é rodada em *hardware* neuromórfico, o que torna o *trade-off* de uma leve perda de acurácia por ganhos potencialmente substanciais de eficiência particularmente atraente para cenários móveis e remotos.

Em síntese, os resultados indicam que SNNs podem ser tecnicamente viáveis para apoio ao diagnóstico de câncer de pele quando a restrição energética é crítica e existe acesso a *hardware* neuromórfico. Persistem, entretanto, desafios de engenharia e.g. maior complexidade de projeto e maturidade de ferramentas, que podem limitar a adoção ampla no curto prazo. Caminhos promissores incluem estudos de conversão ANN→SNN e estratégias *constrain-then-train* direcionadas ao *hardware* alvo, bem como medições energéticas em dispositivos reais e exploração de tarefas com dinâmica temporal, nas quais SNNs podem expressar vantagens adicionais.

Referências

BI, G.-q.; POO, M.-m. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, Society for Neuroscience, v. 18, n. 24, p. 10464–10472, 1998. ISSN 0270-6474. Disponível em: <https://www.jneurosci.org/content/18/24/10464>. Acesso em: 30 ago. 2025.

BOHTE, S. M.; KOK, J. N.; La Poutré, H. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, v. 48, n. 1, p. 17–37, 2002. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231201006580>. Acesso em: 19 set. 2025.

BREIMAN, L. Random forests. *Mach. Learn.*, Springer Science and Business Media LLC, v. 45, n. 1, p. 5–32, out. 2001.

CHOI, S. H. Spiking neural networks for biomedical signal analysis. *Biomed. Eng. Lett.*, Springer Science and Business Media LLC, v. 14, n. 5, p. 955–966, set. 2024.

CORTES, C.; VAPNIK, V. Support-vector networks. *Mach. Learn.*, Springer Science and Business Media LLC, v. 20, n. 3, p. 273–297, set. 1995.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, v. 13, n. 1, p. 21–27, 1967.

DENG, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, IEEE, v. 29, n. 6, p. 141–142, 2012.

ESHRAGHIAN, J. K.; WARD, M.; NEFTCI, E. O.; WANG, X.; LENZ, G.; DWIVEDI, G.; BENNAMOUN, M.; JEONG, D. S.; LU, W. D. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, v. 111, n. 9, p. 1016–1054, 2023.

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, Institute of Mathematical Statistics, v. 29, n. 5, p. 1189–1232, 2001. ISSN 00905364, 21688966. Disponível em: <http://www.jstor.org/stable/2699986>. Acesso em: 23 out. 2025.

GHOSH-DASTIDAR, S.; ADELI, H. *SPIKING NEURAL NETWORKS*. World Scientific Connect, 2009. Disponível em: <https://www.worldscientific.com/doi/abs/10.1142/S0129065709002002>. Acesso em: 31 jul. 2025.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. *arXiv*, 2015.

IAKYMCHUK, T.; ROSADO-MUÑOZ, A.; GUERRERO-MARTÍNEZ, J. F.; BATALLER-MOMPEÁN, M.; FRANCÉS-VÍLLORA, J. V. Simplified spiking neural network architecture and STDP learning algorithm applied to image classification. *EURASIP J. Image Video Process.*, Springer Science and Business Media LLC, v. 2015, n. 1, dez. 2015.

JAVID, M. H. *Melanoma Skin Cancer Dataset of 10000 Images*. Kaggle, 2022. Disponível em: <https://www.kaggle.com/dsv/3376422>. Acesso em: 24 out. 2025.

KANDEL, E. R.; SCHWARTZ, J. H. *Principles of Neural Science*. [S.l.]: Elsevier, 1985.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 60, n. 6, p. 84–90, maio 2017. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/3065386>. Acesso em: 26 out. 2025.

LEMAIRE, E.; CORDONE, L.; CASTAGNETTI, A.; NOVAC, P.-E.; COURTOIS, J.; MIRAMOND, B. An analytical estimation of spiking neural networks energy efficiency. *Arxiv*, 2022.

PFEIFFER, M.; PFEIL, T. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, Volume 12 - 2018, 2018. ISSN 1662-453X. Disponível em: <https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2018.00774>. Acesso em: 9 set. 2025.

PRIDDY, K. L.; KELLER, P. E. *Artificial Neural Networks, An Introduction*. SPIE, 2005. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=BrnHR7esWmkC&oi=fnd&pg=PP13&dq=artificial+neural+networks+introduction&ots=UA3BUlbCLR&sig=DKjEiHMhJB8eo-Pe1hfv-RfUwSc&redir_esc=y#v=onepage&q=artificial%20neural%20networks%20introduction&f=false>. Acesso em: 1 jul. 2025.

SUZUKI, K. *Artificial Neural Networks*. London: IntechOpen, 2011. Disponível em: <https://doi.org/10.5772/644>. Acesso em: 20 ago. 2025.

VENKATARAMANI, S.; ROY, K.; RAGHUNATHAN, A. *Efficient embedded learning for IoT devices*. IEEE, 2016. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7428029>. Acesso em: 10 ago. 2025.

WALCZAK, S. *Artificial Neural Networks.* IGI Global, 2019. Disponível em: <https://www.igi-global.com/chapter/artificial-neural-networks/213116>. Acesso em: 11 jul. 2025.

XIAOXUE, L.; XIAOFAN, Z.; XIN, Y.; DAN, L.; HE, W.; BOWEN, Z.; BOHAN, Z.; DI, Z.; LIQUN, W. Review of medical data analysis based on spiking neural networks. *Procedia Computer Science*, v. 221, p. 1527–1538, 2023. ISSN 1877-0509. Tenth International Conference on Information Technology and Quantitative Management (ITQM 2023). Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050923008967>. Acesso em: 28 out. 2025.

YAMAZAKI, K.; VO-HO, V.; BULSARA, D.; LE, N. *Spiking Neural Networks and Their Applications: A Review.* MDPI, 2022. Disponível em: <https://www.mdpi.com/2076-3425/12/7/863>. Acesso em: 21 jul. 2025.