UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Alvaro Javier Zamudio Sanchez

Classificação de imagens coletadas de câmeras de evento por meio de SNNs *(Spiking Neural Networks)*

Porto Alegre

2023

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Alvaro Javier Zamudio Sanchez

Classificação de imagens coletadas de câmeras de evento por meio de SNNs *(Spiking Neural Networks)*

Projeto de Diplomação II, apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Engenheiro Eletricista

UFRGS

Orientador: Prof. Dr. Tiago Oliveira Weber

Porto Alegre
2023

Alvaro Javier Zamudio Sanchez

Classificação de imagens coletadas de câmeras de evento por meio de SNNs *(Spiking Neural Networks)*

Projeto de Diplomação II, apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Engenheiro Eletricista

BANCA EXAMINADORA

Prof. Dr. Paulo Francisco Butzen
UFRGS

Dr. Edinei Santin

Prof. Dr. Tiago Oliveira Weber
Orientador - UFRGS

Aprovado em 13 de Setembro de 2023.

Resumo

Desafios no campo da visão computacional têm sido abordadas de forma consistente por meio da utilização de Redes Neurais Convolucionais (CNNs). Essas arquiteturas consistem em neurônios interconectados capazes de extrair mapas de características complexas de imagens de entrada, o que viabiliza a classificação autônoma e eficaz de imagens. A aplicação destas redes neurais voltadas para o processo de detecção e classificação de objetos tem particular importância no âmbito dos veículos autônomos, uma vez que o reconhecimento de objetos em imagens é fundamental para a percepção espacial. Embora as Redes Neurais Convolucionais (CNNs) tenham consolidado sua posição como uma das principais arquiteturas líderes no estado da arte, sua aplicabilidade em máquinas com recursos limitados, como veículos autônomos, encontra certas limitações, uma vez que a implementação deste tipo de modelo exige um consumo considerável de memória e energia. Para contornar essas restrições, uma abordagem alternativa envolve a incorporação de metodologias inspiradas em princípios biológicos, aproveitando sua eficiência energética e menor ocupação de memória. Uma estratégia inicial envolve a utilização de câmeras baseadas em eventos em conjunto com a adoção de SNNs *(Redes Neurais Pulsantes)*, o que resulta numa resolução mais elevada, uma latência reduzida além de uma alocação de memória minimizada, em contraste com métodos e dispositivos convencionais. Para a utilização dos dados fornecidos pelas câmeras de eventos, estes dados serão reestruturados na forma de *Voxel Grids*, contendo as dimensões do tempo, posição dos píxeis e os canais das polaridades que sinalizam a ocorrência de ativação dos píxeis. Ao construir o modelo de uma SNN, é necessário utilizar o método de *Surrogate Gradient* para viabilizar o aprendizado de máquina, e assim, tornar-lo apto de adaptar arquiteturas convencionais, desde uma CNN com poucas camadas até modelos mais robustos como a arquitetura VGG-11. Os bancos de dados utilizados para os experimentos neste estudo foram todos capturados por câmeras de eventos, sendo o *DVS-Gesture* e *N-CARS* de origem puramente neuromórfica, e o *CIFAR10-DVS* uma adaptação da base *CIFAR10*. A taxa de acertos obtidos durante a classificação de imagens foram de 86.5%, 88.0%, 65.9% respectivamente, medindo uma atividade não nula de em torno de 10% durante todo o processo.

Palavras-chave: Redes Neurais, DVS, Camêras de Evento, SNNs, LIF, Surrogate Gradient.

Abstract

Challenges in the field of computer vision have been consistently addressed through the use of Convolutional Neural Networks (CNNs). These architectures consist of interconnected neurons capable of extracting complex feature maps from input images, enabling autonomous and effective image classification. The application of these neural networks for object detection and classification is particularly important in the context of autonomous vehicles, as object recognition in images is crucial for spatial perception. Although Convolutional Neural Networks (CNNs) have established their position as one of the leading state-of-the-art architectures, their applicability in resource-constrained machines such as autonomous vehicles faces certain limitations, as the implementation of this type of model requires a considerable consumption of memory and energy. To overcome these restrictions, an alternative approach involves incorporating methodologies inspired by biological principles, leveraging their energy efficiency and lower memory footprint. An initial strategy involves the use of event-based cameras in conjunction with the adoption of Spiking Neural Networks (SNNs), resulting in higher resolution, reduced latency, and minimized memory allocation compared to conventional methods and devices. To use the data provided by event cameras, this data will be structured in the form of Voxel Grids, containing dimensions of time, pixel positions, and polarity channels signaling pixel activation events. When constructing an SNN model, it is necessary to use the Surrogate Gradient method to enable machine learning and make it capable of adapting to conventional architectures, from a CNN with few layers to more robust models such as the VGG-11 architecture. The databases used for experiments in this study were all captured by event cameras, with DVS-Gesture and N-CARS originating purely from neuromorphic sources, and CIFAR10-DVS being an adaptation of the CIFAR10 dataset. The accuracy rates obtained during image classification were 86.5%, 88.0%, and 65.9%, respectively, with a non-null activity measuring around 10% throughout the process.

Keywords: Neural Networks, DVS, Event-Cameras, Spiking Neural Networks, LIF, Surrogate Gradient

Lista de Figuras

Figura 1 – Exemplo de circuito Interno das Câmeras de evento. 13
Figura 2 – Retorno da câmera de evento em função do tempo para cada pixel. . . 14
Figura 3 – Representação de eventos em *Voxel Grids*. 17
Figura 4 – Representação de uma rede neural. 18
Figura 5 – Representação de uma rede neural convolucional. 19
Figura 6 – Representação de uma VGG-11. 20
Figura 7 – Representação de uma SNN formada por LIFs. 22
Figura 8 – Heaviside step function. 24
Figura 9 – Sigmoid & Heaviside Function. 25
Figura 10 – Variação da performance em função de σ. 25
Figura 11 – Exemplos de instâncias da base *DVS-Gesture*. 31
Figura 12 – Sistema de coleção e interpretação de Imagens. 32
Figura 13 – Exemplos das duas classes encontradas em *N-CARS*. 32
Figura 14 – Geração do banco de dados *N-CARS*. 33
Figura 15 – Exemplos de amostras encontradas em *CIFAR10-DVS* junto com os rótulos. 34
Figura 16 – Processo de transformação da base. 35
Figura 17 – Procedimento geral da metodologia. 41
Figura 18 – Voxel Grid. 42
Figura 19 – $Sigmoid(x)$ & $Atan(x)$. 43
Figura 20 – $Sigmoid(x, \sigma = \beta)$. 44
Figura 21 – Camadas da SNN Convolucional. 45
Figura 22 – Distribuição de Classes N-Cars. 47
Figura 23 – Distribuição de classes em *DVS-Gesture*. 48
Figura 24 – Distribuição de Classes em CIFAR10-DVS. 48
Figura 25 – Efeito da escolha da duração Δt de cada *Time-step*. 50
Figura 26 – Efeito da escolha da duração Δt de cada *Time-step*, opções reduzidas. . 50
Figura 27 – Taxa de acerto e perdas em função das *epochs*. 53
Figura 28 – Taxa de acerto em função de T. 53
Figura 29 – Matriz de confusão. 54
Figura 30 – Taxa de acerto e perdas em função das *epochs*. 55
Figura 31 – taxa de acerto em função de T. 55
Figura 32 – Matriz de confusão. 56

Lista de Tabelas

Tabela 1 – Modelos das câmeras de eventos. 40
Tabela 2 – Opção dos hiper-parâmetros. 49
Tabela 3 – Resultados dos hiper-parâmetros. 49
Tabela 4 – Efeito da camda Batch-Normalization. 51
Tabela 5 – Efeito da Função de Otimização. 51
Tabela 6 – Comparação de Resultados N-CARS. 52
Tabela 7 – Comparação de resultados DVS-Gesture. 54
Tabela 8 – Comparação de Resultados CIFAR10-DVS. 56
Tabela 9 – Comparação de Resultados CNNs vs SNNs, ... 57
Tabela 10 – Projeção do Custo Energético. 58
Tabela 11 – Taxa de Atividades Non-Zero em SNNs para cada marco de tempo T. . 58

Sumário

1 INTRODUÇÃO 9
1.1 Objetivo Geral 10
1.2 Objetivos Específicos 10
1.3 Justificativa 10

2 FUNDAMENTAÇÃO TEÓRICA E REVISÃO BIBLIOGRÁFICA . . 12
2.1 Câmeras de Evento 12
2.1.1 Principio de Funcionamento 12
2.1.2 Aquisição de Sinal 14
2.1.3 Voxel Grid 15
2.2 Aprendizado de Máquina 17
2.2.1 ANNs (*Artificial Neural Networks*) 17
2.2.2 CNNs (*Convolutional Neural Networks*) 18
2.2.2.1 VGG11 19
2.2.3 SNNs (*Spiking Neural Networks*) 20
2.2.4 LIFs para construção de SNNs 21
2.2.4.1 Surrogate Gradient 23
2.3 Métricas de Consumo Computacional em ANNs e SNNs 26
2.3.1 Custo Operacional 26
2.3.2 Custo de Alocação de Memoria 27
2.3.3 Custo de Ocupação de Memória 28
2.4 Bases de dados 30
2.4.1 DVS-Gesture 31
2.4.2 N-CARS 32
2.4.3 CIFAR10-DVS 33
2.5 Trabalhos Relacionados 35
2.5.1 Câmeras de Evento 35
2.5.2 SNNs 36

3 METODOLOGIA 38
3.1 Materiais e ferramentas 38
3.1.1 Ferramentas computacionais 38
3.1.2 Hardware Utilizado 38
3.2 Procedimento 40
3.2.1 Estudo e Divisão das Bases de Dados 41
3.2.2 Tratamento das Base de Dados 41

3.2.3 Escolha e Otimização de Hiper-parâmetros . . . 43
3.2.4 Seleção das Topografias e Escolha da Topografia Final . . . 44
3.2.5 Estimativa de Consumo de Potência . . . 46

4 ANÁLISE DE RESULTADOS . . . 47
4.1 Estudo estatístico das bases de dados . . . 47
4.2 Estudo da Otimização de uma SNN . . . 48
4.2.1 Hiper-parâmetros . . . 49
4.2.2 Resolução e *Time-Steps* . . . 49
4.2.3 Batch-Normalization . . . 50
4.2.4 Desempenho da rede em função da *loss function* . . . 51
4.3 Analise e Comparação dos Resultados Finais Obtidos . . . 52
4.3.1 N-CARS . . . 52
4.3.2 DVS-Gesture . . . 52
4.3.3 CIFAR10-DVS . . . 54
4.4 Estudo de Desempenho da Rede e Gasto Energético . . . 57

5 CONCLUSÕES . . . 60
5.1 Trabalhos Futuros . . . 60

REFERÊNCIAS BIBLIOGRÁFICAS . . . 62

1 Introdução

Redes neurais convolucionais (CNNs) atualmente são a solução para problemas de visão computacional graças aos seus impressionantes resultados quando submetidas a reconhecimento de imagens. Essas redes são compostas por camadas de unidades de neurônios artificiais, que se conectam por pesos e realizam operações matemáticas para reconhecer padrões e características nos dados de entrada. No entanto, a implementação desse tipo de rede em hardware convencional (CPU/GPU) resulta em um alto consumo de energia, dificultando a sua integração em sistemas embarcados que precisam ter o seu consumo energético reduzido. Em um carro, por exemplo, os algoritmos embarcados têm restrições muito altas em termos de energia, latência e precisão.

Para projetar algoritmos de visão computacional energeticamente mais eficientes, é possível seguir uma abordagem inspirada na neurociência, onde câmeras de eventos (traduzido do inglês *Event-cameras*) e *Spiking Neural Networks* (SNNs) passariam a compor todo o processo de captação, modelagem e interpretação dos dados.

As câmeras de eventos são dispositivos de aquisição de imagem construídos com fotossensores, que registram eventos dinâmicos de mudanças de intensidade de iluminação, ao invés de gerar imagens completas como as câmeras tradicionais, pois elas funcionam medindo a variação do fluxo de carga elétrica entre o sinal de luz em cada pixel, ao invés de medir diretamente a intensidade de luz. Quando ocorre uma mudança de iluminação no pixel, é gerado o que é chamado de um evento, que é registrado junto com sua posição e horário. Esses eventos são então utilizados para reconstruir uma imagem. As SNNs, por sua vez, são uma variante das redes neurais tradicionais que se baseiam em um modelo mais próximo do processamento neural biológico. Ao invés de usar sinais contínuos, elas utilizam sinais de pulso, chamados de *spikes* para representar e transmitir informações, e os neurônios desta rede neural são configurados de maneira a imitar a forma como o cérebro processa informações.

É inegável que as câmeras de eventos têm muitos benefícios a oferecer em comparação com câmeras convencionais, como baixa potência, elevada faixa dinâmica, alta resolução temporal e latência reduzida, entretanto, é importante destacar também que o potencial das câmeras de eventos apenas é possível alcançar por se tratar de uma nova tecnologia, que trabalha com um formato de dados cuja implementação e processamento se dão por mecanismos totalmente diferentes dos que atualmente conhecemos. Além disso, o alto custo das câmeras de eventos é um fator limitante para a sua utilização global, o que faz com que haja poucos dados disponíveis para trabalhar e consequentemente não existem vastas bibliotecas prontas para o uso de treinamento de algoritmos de *Machine*

Learning, como existem em câmeras convencionais. Assim, é um mercado que ainda não está plenamente desenvolvido, o que acarreta num reduzido número de algoritmos de *Machine Learning* testados ou desenvolvidos para lidar com essa tecnologia.

1.1 Objetivo Geral

O objetivo geral do projeto é criar um sistema, baseado em câmeras de evento, capaz de realizar a classificação computacional de imagens por meio de SNNs, otimizando assim, o consumo energético de todo o processo ao mesmo tempo que se mantém próximo da taxa de acerto de redes neurais convencionais.

1.2 Objetivos Específicos

- Estudar o funcionamento e características de câmeras baseadas em evento.
- Estudar o modelo de dados que câmeras baseadas em evento transmitem dentro do contexto de 3 bancos de dados voltados para classificação de imagens: *DVS Gesture*, *N-CARS*, *CIFAR10-DVS*.
- Desenvolver algoritmos usando SNNs para a classificação de imagens, e comparar desempenho com outras técnicas de reconhecimento de imagem.
- Analisar resultados e comparar com outros trabalhos da literatura.

1.3 Justificativa

A justificativa para o desenvolvimento voltado para a integração de câmeras de evento com algoritmos capazes de processar e moldar os dados extraídos deste dispositivo se dá pelo alto potencial que este tipo de tecnologia oferece para o mercado de veículos autônomos, visto que as câmeras de evento funcionam de maneira dinâmica, e oferecem para o sistema uma melhor qualidade de imagem, e sem desfoque quando sujeito ao movimento da câmera, assim, contornam com facilidade problemas qualitativos encontrados em imagens de câmeras convencionais, dos quais alguns são citados abaixo:

- Alta resolução temporal, visto que mudanças de iluminação são constantes, logo, como essas mudanças podem ser captadas num intervalo de tempo na ordem de microssegundos (μs), seria o equivalente a obter uma taxa de quadros por segundo na ordem de 10^6;
- Baixa Latência na ordem de centenas de micro segundos, assim, oferece um aumento na velocidade de reação do sistema embarcado;

- Baixo consumo de energia na ordem de $10mW$ contra centenas de mW de câmeras convencionais;
- Alta faixa dinâmica: $> 120dB$ contra $60dB$ de câmeras convencionais, o que permite obter alta qualidade de imagens tanto em ambientes muito iluminados, como também em ambientes muito escuros, afinal, as câmeras de evento são extremamente sensíveis à mudança de iluminação e não à sua intensidade;

Adicionalmente, a integração das Redes Neurais do tipo SNNs a esse contexto decorre da inexistência da exigência de um pré-processamento, uma vez que ambas as tecnologias operam sobre a mesma estrutura de dados. Além disso, as SNNs demonstram uma notável capacidade para explorar eficazmente as características espaço-temporais que são inerentes às saídas das câmeras baseadas em eventos.

2 Fundamentação Teórica e Revisão Bibliográfica

Neste capítulo, serão discutidos e analisados temas e trabalhos pertinentes para o desenvolvimento do projeto. Primeiro será revisado o princípio das câmeras de evento, e posteriormente analisaremos técnicas de aprendizado de máquina, tanto convencionais, como as CNNs, assim como também técnicas inspiradas na neurociência, que é o caso das SNNs sendo um dos principais tópico abordados durante este projeto.

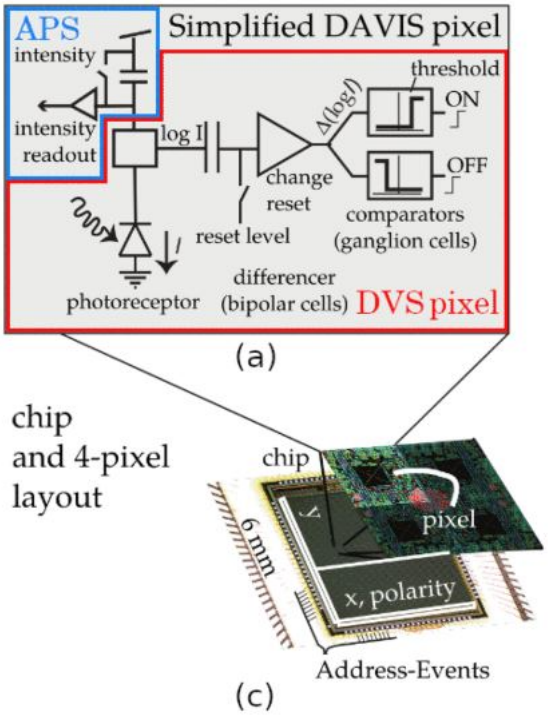
2.1 Câmeras de Evento

Câmeras de Evento, como os modelos de sensores de visão dinâmica (DVS), são diferentes das câmeras convencionais, pois respondem a mudanças logarítmicas de intensidade luminosa de maneira dinâmica e independente para cada pixel, esta intensidade luminosa é denotada como "I" e sua unidade é candela (cd). Em vez de capturar imagens completas a uma taxa fixa, as câmeras de evento produzem uma sequência variável de "eventos" digitais, onde cada evento representa uma mudança de intensidade luminosa detectada pelos pixels em um determinado momento. Essa abordagem permite que as câmeras de eventos tenham uma latência muito baixa e um alto alcance dinâmico, tornando-as ideais para aplicações em tempo real como tarefas de visão computacional.

2.1.1 Principio de Funcionamento

As câmeras de evento são compostas por pixels construídos a partir de fotossensores, onde cada pixel é capaz de memorizar a intensidade logarítmica a cada "evento" capturado e monitora continuamente uma mudança de magnitude suficiente a partir desse valor memorizado. O circuito analógico simplificado da Figura 1 mostra a integração analógica dos pixels do tipo APS (*Active Pixel Sensor*) e DVS (*Dynamic Vision Sensor*), que formam as câmeras híbridas.

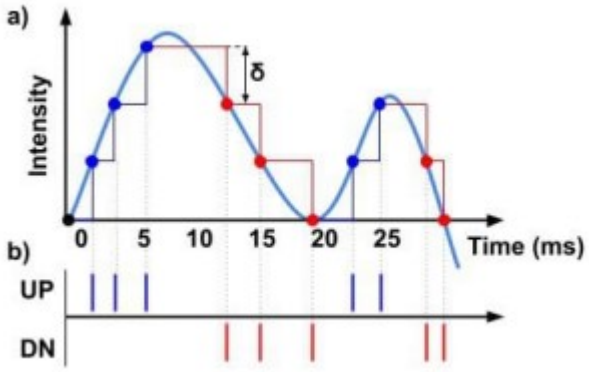
Figura 1 – Exemplo de circuito Interno das Câmeras de evento.



Fonte: (GALLEGO *et al.*, 2022)

Quando a mudança de iluminância detectada excede um *threshold*, o dispositivo envia um evento, que contém informações como a localização do pixel no espaço de resolução da câmera (x, y), além de informar o tempo t e a polaridade da mudança, ou seja, informando se a mudança de iluminância foi positiva ou negativa. A saída de eventos pode ser resumida como está na Figura 2 considerando apenas um pixel. Assim, é importante ressaltar que as câmeras de evento são sensores orientados por variações de dados: sua saída depende da quantidade de movimento na intensidade luminosa na cena, desta forma, quanto mais rápido o movimento, mais eventos por segundo são gerados. Os eventos são registrados com resolução de microssegundos e são transmitidos com latência de sub-mili-segundos, o que faz com que esses sensores tenham a capacidade de reagir rapidamente aos estímulos visuais.

Figura 2 – Retorno da câmera de evento em função do tempo para cada pixel.



Fonte: (JOHANSSON, 2021)

Em virtude desta nova tecnologia, as câmeras de evento possuem diversas vantagens em relação às câmeras convencionais. Em primeiro lugar, elas oferecem alta resolução temporal, com monitoramento rápido das mudanças de iluminância em circuitos analógicos e leitura digital dos eventos equivalente na ordem de kHz. Além disso, cada pixel das câmeras de evento funciona independentemente, sem necessidade de esperar pela ação do conjunto de pixels que compõem o dispositivo, resultando numa latência mínima de cerca de um milissegundo. Isso significa que as câmeras de evento podem capturar movimentos muito rápidos sem sofrer com o borrão de movimento típico das câmeras convencionais.

Outra vantagem é o baixo consumo de energia, pois as câmeras de evento transmitem apenas sinalizações de mudanças de iluminância e removem dados redundantes, permitindo que a energia seja usada apenas para processar pixels em mudança. Enquanto a maioria das câmeras de eventos consomem cerca de 10 mW a 100 mW, câmeras convencionais costumam ter um consumo na ordem de 1 W. Por fim, as câmeras de evento possuem alta faixa dinâmica (*Dynamic Range*), que gira em entorno de 120 dB, que supera notavelmente os 60 dB das câmeras convencionais de alta qualidade. Isso ocorre porque os fotorreceptores dos pixels operam em escala logarítmica e cada pixel funciona de maneira independe. Desta forma, de maneira semelhante às retinas biológicas, os pixels das câmeras de evento podem se adaptar a estímulos vindo de lugares muito escuros ou muito brilhantes, mantendo um alto nível de detalhamento das imagens.

2.1.2 Aquisição de Sinal

Como visto anteriormente, câmeras de evento são compostas por pixels que respondem de maneira independente à mudança logarítmica da intensidade luminosa (L) detectada pelos fotossensores, sendo I a intensidade luminosa medido em candela. Estas variações, ao superarem um certo limite, como mostrado na Equação 1, onde C é uma constante e $p \, \epsilon \, \{-1, 1\}$ informa a polaridade da mudança detectada, podendo ser positiva ou negativa, disparam eventos que podem ser representados pelo agrupamento ϵ descrito

na Equação 3, com (x, y) sendo as coordenadas dos pixels da câmera e t o tempo onde o cada evento foi registrado.

$$L = log(I) \tag{1}$$

$$L(x, y, t) - L(x, y, t - \Delta t) \geq pC \tag{2}$$

$$\epsilon = \left\{e_k\right\}_{k=1}^{N} = \left\{(x_k, y_k, t_k, p_k)\right\}_{k=1}^{N} \tag{3}$$

Assim, para trabalhar com a série de eventos ϵ que vem diretamente das câmeras de eventos, podemos processar cada evento por separado, porém, como a resolução temporal das câmeras de eventos é na ordem de microssegundos, o processamento individual de cada evento potencialmente se tornara pesado, ocupando muita memória e energia. Desta maneira, é preferível agrupar os eventos, e existem diversas técnicas e metodologias para executar este tipo de processamento. Uma das formas mais simples de utilizar dados de eventos para aprendizado supervisionado é acumulando os eventos durante um período de tempo, seja contando o número de mudanças de intensidade luminosa detectadas por pixel ou acumulando suas polaridades, gerando assim um quadro de eventos, também conhecido como histograma 2D. É valido destacar, porém, que este tipo de abordagem requer um certo nível de pré-processamento e altera a natureza dos dados, pois desconsidera o aspecto temporal das câmeras. Uma outra representação popular para dados de eventos utilizada em aprendizado supervisionado de classificação de imagens, são os *Voxel grids*, onde cada *voxel* representa um pixel e um intervalo de tempo. Essa representação preserva melhor as informações temporais, mas exige mais memória e cálculo, ainda assim, ela é a preferida para a utilização de SNNs, já que os eventos podem ser interpretados como pulsos emitidos em um determinado momento, eliminando a necessidade do pré-processamento

2.1.3 Voxel Grid

Como visto anteriormente, as câmeras de disparo assíncrono têm uma alta resolução devido à sua natureza assíncrona, podendo chegar à faixa de microssegundos, o que resultaria numa taxa de disparo equivalente a 1 milhão de quadros por segundo. Isso se deve ao fato de que em vez de capturar imagens inteiras em intervalos fixos, como as câmeras convencionais, essas câmeras capturam mudanças de iluminação em cada pixel em tempo real. Essas mudanças são representadas no formato de dado x, y, t junto com a polaridade, indicando respectivamente a posição do pixel, o tempo da mudança de iluminação detectada, e se a mudança foi positiva ou negativa.

No entanto, esses dados são difíceis de usar diretamente em algoritmos de aprendizado de máquina, pois há um alto volume de fluxo de eventos, o que pode tornar o processo pesado em termos de memória alocada (HE *et al.*, 2020). Para contornar esse problema, os pesquisadores propuseram o uso de *Voxel Grids*, que consistem em agrupar eventos em

matrizes da seguinte forma: os eventos são acumulados em janelas de tempo maiores (Δt segundos totalizando um total de T intervalos de tempo), mantendo a resolução espacial inalterada, essencialmente constituindo uma grade de *voxels* (o que origina o termo *Voxel Grid*). Considerando a gravação de eventos com um total de d segundos de duração, é possível reunir os eventos em $T = d/\Delta t$.

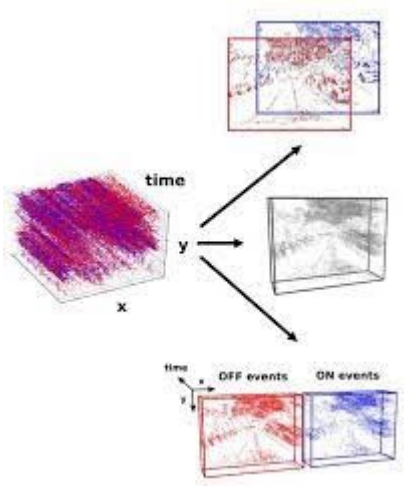
Os dados do evento são armazenados na forma de um tensor $CTHW$ de 4 dimensões, em que C é o número de canais contendo as polaridades de mudança de iluminação, T é o número de etapas de tempo, H e W são a localização dos pixels que transmitem os eventos. Esse método de organização de dados permite que os eventos sejam interpretados por *Spiking Convolutional Networks* de C dimensões onde estas trabalhariam separadamente em cima de cada instante de tempo T. Assim, o *Voxel Grid* consegue manter informações temporais dos eventos e utiliza uma acumulação binária em janelas de tempo, ou seja, não são contados os eventos, apenas se identifica se ao menos um evento ocorreu na janela de tempo Δt. Essa limitação resulta em uma perda de informação, mas é necessária para viabilizar o uso em tempo real do sistema, em que o fluxo de eventos é processado instantaneamente. De maneira geral, essa abordagem equivale a modificar a resolução temporal da câmera de eventos para uma resolução de Δt.

Voxel grids, como ilustrados na Figura 3, na sua essência são construídos utilizando apenas um Canal C, onde se guarda a informação dos dois canais originais que contêm as polaridades, seja contando o número de eventos enviados destes canais para cada janela de tempo T, seja apenas sinalizando que houve pelo menos um evento disparado por meio da lógica *OR* (logica analógica do "OU"). Esta escolha é feita para diminuir a memória consumida durante todo o processo de interpretação de dados.

Neste projeto, para os Datasets mais leves, como N-CARS e DVS-Gesture, optamos por manter o número total de canais originais (C) em sua totalidade (ou seja, dois canais, um para cada polaridade). Isso nos permite sinalizar se pelo menos um evento foi disparado e se esse evento foi positivo ou negativo. Embora essa abordagem consuma mais memória em comparação com o método original, estudos anteriores indicam que a inclusão dos canais de polaridade pode melhorar o desempenho da rede neural em até 12% em tarefas de aprendizado de máquina (GEHRIG *et al.*, 2019).

Por outro lado, no estudo do banco de dados CIFAR10-DVS, optamos por utilizar apenas um canal. Essa decisão visa reduzir o tamanho da rede neural, uma vez que o CIFAR10-DVS é um conjunto de dados mais pesado e requer mais poder computacional para processamento.

Figura 3 – Representação de eventos em *Voxel Grids*.



Fonte: (GEHRIG *et al.*, 2019)

2.2 Aprendizado de Máquina

O aprendizado de máquina é uma das áreas mais promissoras da inteligência artificial, com inúmeras aplicações dentro da indústria e da pesquisa. A ideia central se concentra em ensinar as máquinas a aprender com dados, sem precisar programá-las explicitamente para cada tarefa.

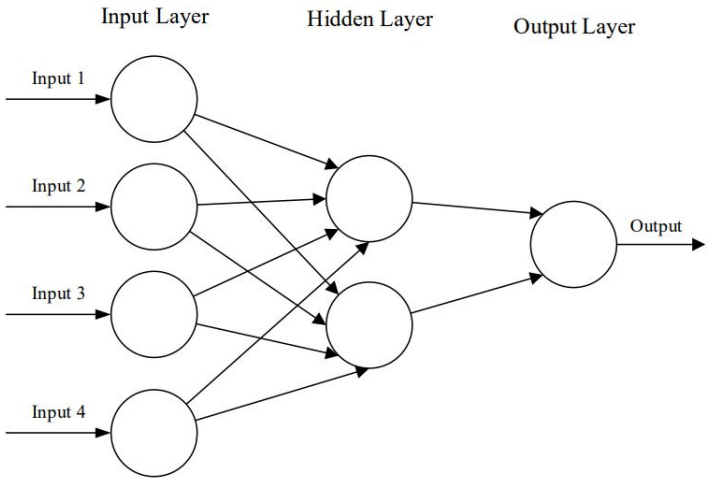
Uma das principais técnicas de aprendizado de máquina são as redes neurais artificiais (*Artificial Neural Networks*), que se inspiram no funcionamento do cérebro humano para realizar tarefas complexas, como reconhecimento de imagens e sons, ou até mesmo previsões de tendências. As redes neurais artificiais são compostas por camadas de ”neurônios” interconectados que processam informações e geram saídas. Essas redes são treinadas em grandes conjuntos de dados para reconhecer padrões e tomar decisões.

2.2.1 ANNs (*Artificial Neural Networks*)

A arquitetura das redes neurais artificiais de maneira geral pode variar desde uma simples camada de neurônios até várias camadas de neurônios interconectados, assim, uma rede neural profunda é aquela que tem várias camadas interconectadas, também chamadas de *layers*. A arquitetura das redes neurais artificiais, assim, é definida pelo número de camadas e neurônios em cada camada. Existem diferentes tipos de ANNs, cada um com uma arquitetura específica e adequada para diferentes tipos de tarefas. Ao considerar as estruturas mais básicas de uma ANN, é possível representar a sua arquitetura como mostra a Figura 4, ilustrando uma camada inicial que recebe os valores de entrada, geralmente na forma de um vetor multidimensional, e estes valores serão distribuídos para as camadas

ocultas por meio de funções matemáticas que podem ser parametrizadas, de maneira que seja obtido o resultado desejado na saída da rede neural.

Figura 4 – Representação de uma rede neural.



Fonte: (O'SHEA; NASH, 2015)

2.2.2 CNNs (*Convolutional Neural Networks*)

As redes neurais convolucionais (CNNs), são um tipo de rede neural frequentemente usadas para análise de imagens, pois elas são capazes de extrair características relevantes de uma imagem, independentemente de sua posição. Essa capacidade é possível graças à arquitetura das CNNs, que consiste em várias camadas de processamento. A primeira destas camadas é a camada de convolução que é a principal camada em uma CNN. Ela consiste em uma série de filtros que deslizam sobre a imagem de entrada para extrair recursos importantes. Cada filtro é uma matriz de pesos que é aplicada a uma parte da imagem de entrada por meio de uma operação de convolução.

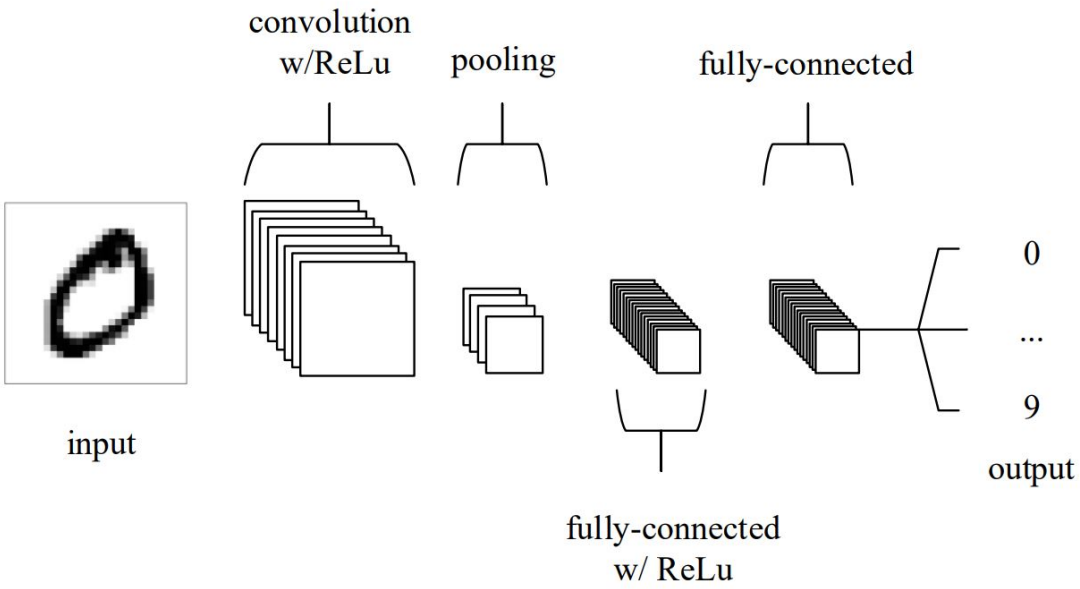
A operação de convolução envolve multiplicar cada elemento da matriz de filtro pelo valor correspondente na imagem de entrada e, em seguida, somar os resultados. O resultado é armazenado em uma nova matriz que, de certa maneira, são imagens em uma escala reduzida que reúne as principais características da imagem original.

Na sequência se encontra a camada de ativação, que é usada para introduzir não-linearidade na rede. A operação de convolução é uma operação linear, portanto, a adição de uma camada de ativação permite que a rede seja capaz de resolver problemas mais complexos.

Outra camada que faz parte da estrutura de uma CNN é a camada de *pooling* que é usada para reduzir a dimensão espacial das matrizes geradas pela primeira camada de convolução. Isso ajuda a reduzir o número de parâmetros na rede e a tornar a computação mais eficiente.

Por último temos uma camada que é semelhante a uma camada densa em uma rede neural clássica, e ela serve para computar o valor final da saída, sendo assim, tornando-se a última camada de toda a rede neural. Na Figura 5 podemos ver uma representação gráfica da arquitetura de uma CNN básica.

Figura 5 – Representação de uma rede neural convolucional.



Fonte: (O'SHEA; NASH, 2015)

2.2.2.1 VGG11

A VGG-11, é um modelo baseado em CNNs, criada a partir dos modelos originais VGG-16 e VGG-19, ela é conhecida por sua simplicidade e eficácia em produzir resultados precisos em tarefas de classificação de imagens. O número "11" em seu nome refere-se ao número total de camadas na arquitetura, como pode ser visto na Figura 6. Ao contrário de algumas redes contemporâneas que se concentravam em criar tipos de camadas novos ou conexões complexas, a VGG-11 adotou uma abordagem direta empilhando múltiplas camadas de filtros convolucionais de tamanho pequeno (tipicamente 3x3), seguidas por camadas de *max-pooling* para reduzir as dimensões espaciais. Essa estrutura repetitiva de filtros pequenos permitiu à VGG aprender características hierárquicas de complexidade crescente.

Figura 6 – Representação de uma VGG-11.

Fonte: Adaptado de (SIMONYAN; ZISSERMAN, 2015)

2.2.3 SNNs (*Spiking Neural Networks*)

As SNNs (Spiking Neural Network) são redes neurais inspiradas na neurociência, pois a sua arquitetura consiste em neurônios, que se comunicam por meio de *spikes*, conectados por pesos ajustáveis e que modelam o aprendizado da máquina. Existem diversos modelos de SNNs, entre os quais se destacam o LIF (*Leaky Integrate-and-Fire*),

ou também um modelo biologicamente mais realista, o neurônio de *Hodgkin-Huxle*. Devido à sua simplicidade, o neurônio da classe LIF é amplamente utilizado, além do fato de que a utilização de modelos mais realistas exigem mais poder de processamento e memoria, o que acaba fugindo das diretrizes de aplicações de baixo consumo potência das SNNs.

Como é o caso das redes neurais tradicionais, é possível ajustar os pesos que conectam os neurônios, no entanto, os *spikes* naturalmente são variáveis discretas e, portanto, não são diferenciáveis, o que impede o uso da recorrente técnica de retro propagação em SNNs. Como resultado, várias regras de aprendizado foram propostas utilizando este tipo de rede neural para contornar este empecilho.

De maneira geral, existem três maneiras de configurar as SNNs, a primeira técnica é com aprendizado de máquina não supervisionado, utilizando o método de STDP (*spike timing dependent plasticity*) onde os pesos atualizados se baseiam nos intervalos de tempo entre os *spikes* gerados pelos neurônios pré-sinápticos (que emitem o sinal) e pós-sinápticos (receptores do sinal), entretanto, esta técnica é limitada quando submetida à aprendizagem de máquina supervisionada de bancos de dados maiores que o MNIST (DENG, 2012).

O segundo método é utilizando aprendizado de máquina supervisionado indireto, por meio da conversão de redes neurais artificiais convencionais em SNNs, e é atualmente o método mais implementado para modelar SNNs. Neste método, as redes neurais convencionais são treinadas e depois convertidas em sua versão SNNs. No entanto, esse treinamento indireto ajuda pouco na revelação de como os SNNs aprendem, uma vez que implementa apenas a fase de inferência em formato SNN, ou seja, o treinamento é em formato convencional e não no formato SNN.

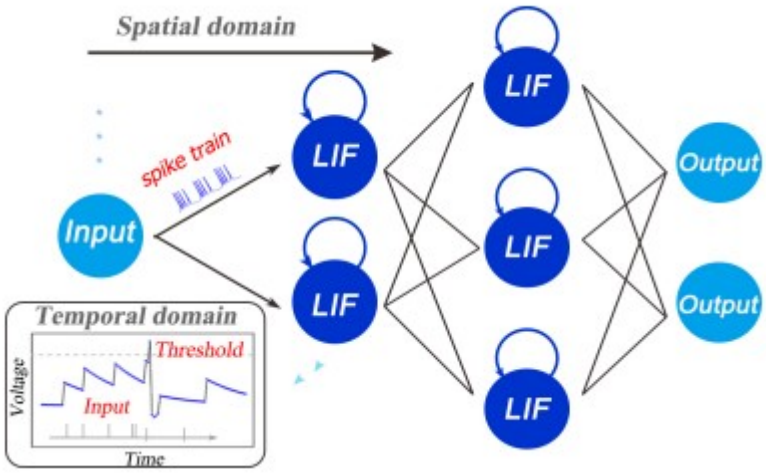
A terceira opção é o método de aprendizado supervisionado direto, que é ideal para a aplicação quando tratamos câmeras de evento, pois a natureza dos dados do algoritmo e do dispositivo se baseiam na neurociência e não há a necessidade de executar um pré-processamento nos dados. Este método, tem sido o tema principal de pesquisas que visam habilitar o aprendizado por meio de retro propagação, pois apesar da natureza discreta do dado não ser uma variável diferençável, o que é essencial para a execução deste método, existem estudos para criar uma função aproximada que possibilite este tipo de aprendizado.

2.2.4 LIFs para construção de SNNs

O modelo LIF, nome que vem da sua origem no termo inglês *Leaky Integrate-and-Fire Model*, é um modelo matemático amplamente utilizado na área de redes neurais que se inspiram na neurociência computacional. As redes neurais de disparo são compostas por neurônios que se comunicam entre si através de sinais elétricos discretos, chamados de *spikes*, que são transmitidos de um neurônio para outro.

O modelo LIF é baseado na ideia de que a atividade elétrica de um neurônio pode ser representada como a carga elétrica em um capacitor, que é atualizada a cada vez que o neurônio recebe um *spike*. A cada atualização, parte da carga elétrica é perdida devido ao vazamento de carga do capacitor, e se a carga restante ultrapassar um determinado limite, o neurônio dispara um *spike* para os neurônios conectados a ele, além de ter a sua carga elétrica reiniciada a um valor pré-determinado.

Figura 7 – Representação de uma SNN formada por LIFs.



Fonte: (WU *et al.*, 2018b)

O LIF é um modelo simples, mas muito eficaz para simular a atividade de neurônios individuais em uma rede neural de disparo e tem como principal vantagem a sua eficiência computacional devido justamente à sua simplicidade, a Equação 4 é a função matemática adotada neste modelo, onde τ é a taxa constante da queda de carga, $u(t)$ é a carga elétrica no neurônio no instante t e $I(t)$ é a corrente de entrada no neurônio, o que virão a ser os *spikes* que provêm de neurônios que fazem parte de camadas antecessoras.

$$\tau \frac{du(t)}{dt} = -u(t) + I(t) \tag{4}$$

Apesar da Equação 4 se tratar de um modelo matemático simples, pode se tornar extremamente complexa ao ser submetida a métodos de retro propagação de erro para o treinamento de várias camadas de neurônios conectadas. Isso ocorre porque o método da retro propagação utiliza técnicas de otimização de parâmetros, como o *gradient descent*, que gera a regra da cadeia para a propagação de erro camada por camada.

Para contornar esse problema e manter a natureza do modelo LIF, assim como o fluxo de informação entre camadas de neurônios, é possível transformar a Equação 4 em um conjunto de equações discretas, como consta na documentação da biblioteca *SpikingJelly* (FANG *et al.*, 2020). Essas equações discretas representam a interação entre os neurônios que compõem uma SNN e podem ser rescritas da seguinte maneira:

$$V[t] - V[t-1] = -\frac{1}{\tau}(V[t-1] - V_{reset}) + X[t] \tag{5}$$

Isolando $V[t]$ obtém-se:

$$V[t] = f(V[t-1], X[t]) = V[t-1] - \frac{1}{\tau}(V[t-1] - V_{reset}) + X[t] \tag{6}$$

E para reiniciar o valor de $V[t]$ ao seu estado inicial, depois do neurônio ser sobrecarregado e mandar um *spike* para as camadas posteriores temos:

$$V[t] = V[t] - V_{threshold} \tag{7}$$

Por último, mas não menos importante, é necessário considerar também a função de ativação, $\Theta(x)$, da SNN representada pela função *heaviside step function* como segue abaixo:

$$\Theta(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{8}$$

Com as equações acima, se constrói toda a base para a arquitetura dos LIFs, assim como também qualquer outro tipo de neurônio orientado à construção de SNNs, uma vez que o comportamento destes neurônios podem ser moldados pelas 3 equações abaixo referente ao carregamento do estado de tensão do neurônio ($H[t]$), o envio de sinal na saída (emissão de *spike* definida por $S[t]$), e também o reinício ao estado inicial do neurônio, respectivamente:

$$H[t] = f(V[t-1], X[t]) \tag{9}$$

$$S[t] = \Theta(H[t] - V_{threshold}) \tag{10}$$

$$V[t] = H[t] - V_{threshold} \cdot S[t] \tag{11}$$

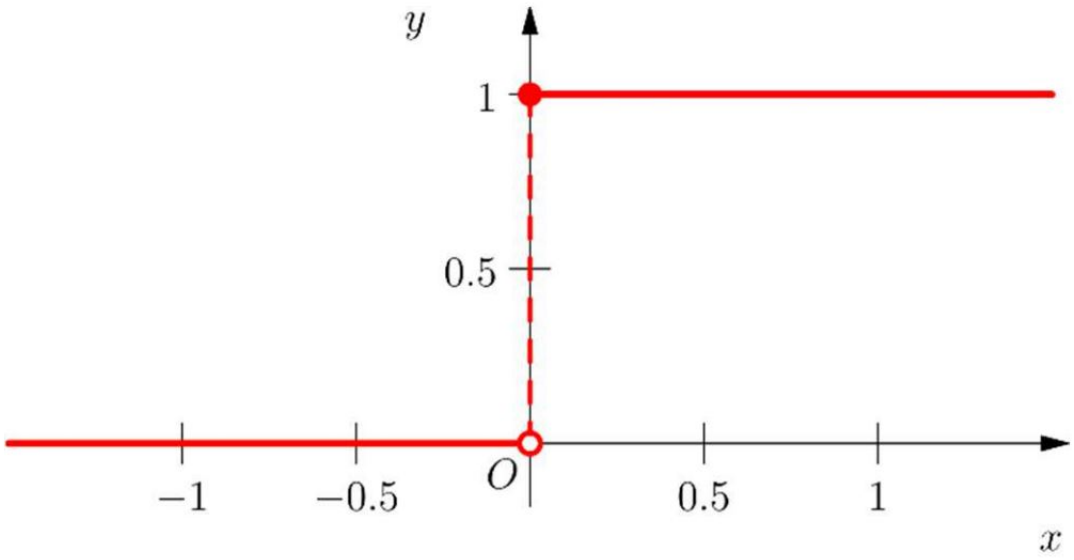
É valido destacar que nas 3 equações acima, $H[t]$ e$V[t]$ representam a mesma variável, porém $H[t]$ representa o estado de carregamento da membrana antes de atingir a tensão máxima $V_{threshold}$ e segue uma função específica $f(V[t-1], X[t])$ para cada modelo de neurônio (o modelo LIF segue a Equação 6), enquanto $V[t]$ representa o estado da membrana logo após ter o seu estado reiniciado e é igual para todos os modelos de neurônios usados na construção de SNNs.

2.2.4.1 Surrogate Gradient

Como visto na Equação 10, a emissão de *spikes* se dá pela função de *Heaviside step function*, a qual não é diferenciável, pois a sua derivada é nula em todos os lugares, exceto

em 0, onde não é definida. Por causa deste ponto de descontinuidade da função, que fica evidente na Figura 8, a otimização do aprendizado de máquina dos neurônios baseada em gradientes e a retro-propagação fica inviável.

Figura 8 – Heaviside step function.

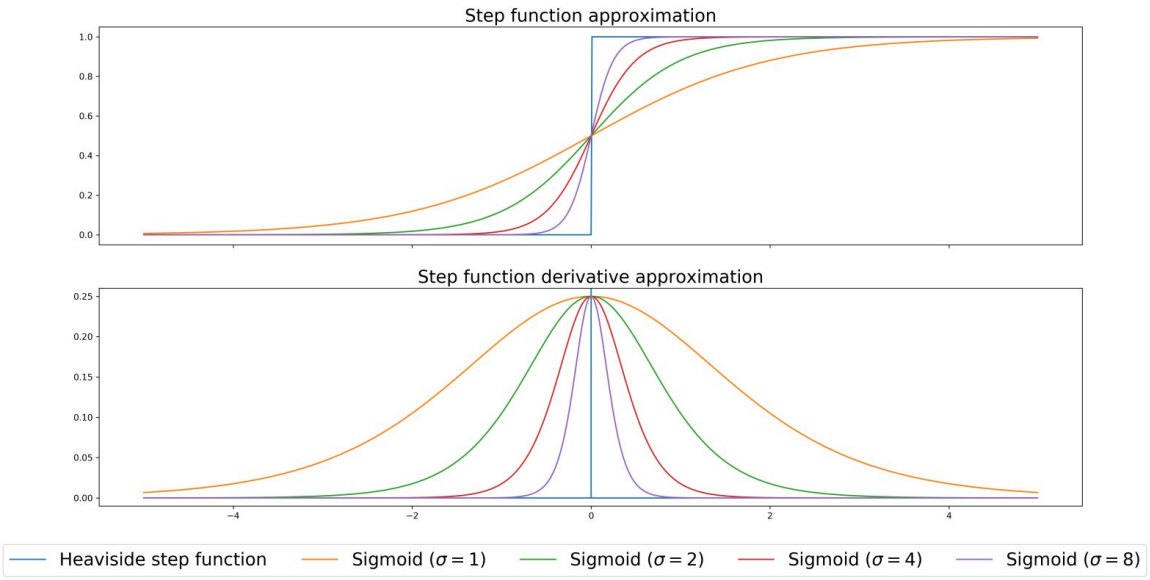


Fonte: (ZAKHAROV *et al.*, 2021)

Vários métodos foram estudados para contornar esta limitação, e o método de *Surrogate Gradient* é o que mais tem se destacado e conseguido viabilizar bons desempenhos em dataset mais complexos e robustos que o MNIST.

Neste método, para viabilizar a retro-propagação por meio do cálculo do gradiente, se substitui, apenas nesta etapa do cálculo do gradiente, a função *Heaviside step function* por alguma outra função semelhante que seja diferenciável, como a função Sigmoide, que tem um comportamento semelhante à da Equação 8 tanto na sua forma integral, assim como na função diferenciável conforme mostrado na Figura 9.

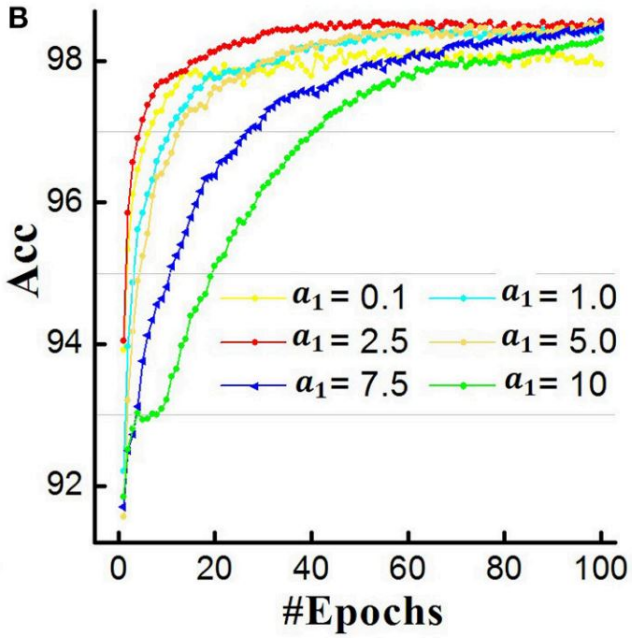
Figura 9 – Sigmoid & Heaviside Function.



Fonte: (ZIMMER *et al.*, 2019)

É valido destacar que o argumento σ da função Sigmoide é arbitrário e serve como um hiper-parâmetro durante o processo de aprendizado de máquina, pois este valor pode impactar diretamente no desempenho da rede neural como mostrado em (WU *et al.*, 2018b), onde foram usados diversos valores de σ para realizar a classificação de imagens do dataset MNIST, e foram obtidos diferentes resultados como pode ser observado na Figura 10.

Figura 10 – Variação da performance em função de σ.



Fonte: (WU *et al.*, 2018b)

2.3 Métricas de Consumo Computacional em ANNs e SNNs

Nesta seção, serão abordadas as métricas delineadas no trabalho de referência (CORDONE, 2022) com o intuito de mensurar o custo computacional associado à execução de tarefas de classificação. Estas tarefas compreendem a utilização de Redes Neurais de Spiking (SNNs) e Redes Neurais Artificiais Convencionais (CNNs). O objetivo principal é a análise subsequente do poder computacional empregado e da precisão alcançada por cada um desses modelos em relação a cada conjunto de dados empregado no projeto. Nesse sentido, serão quantificadas as interações que envolvem as operações das redes neurais, as alocações de memória necessárias durante o processo e o processo que determina os endereços de memória onde as variáveis são alocadas.

2.3.1 Custo Operacional

O custo operacional em relação a uma operação da rede neural se refere à carga computacional associada à realização das operações matemáticas reais, como multiplicações e adições, necessárias para a operação e podem ser expressas em função das métricas de multiplicação-acumulação (MAC) e acúmulo das saídas(ACC). A métrica de operações MAC normalmente quantifica a quantidade de parâmetros envolvidos em cada camada da rede neural, indicando a quantidade de operações de multiplicação e adição realizadas durante o processamento dos dados. Por outro lado, a métrica de operações ACC representa o número de resultados acumulados em cada camada da rede, o que ajuda a avaliar o volume de operações de adição realizadas em cada etapa da computação.

Assim, em uma camada de convolução, a entrada e a saída são compostas por um conjunto de mapas de características, com formatos de $(C_{in} \times H_{in} \times W_{in})$ e $(C_{out} \times H_{out} \times W_{out})$, respectivamente. O número de filtros é definido por C_{out}, e o tamanho de cada filtro é especificado como $C_{in} \times H_{kernel} \times W_{kernel}$. Nesse contexto, as operações de integração de matrizes são realizadas por meio de operações de MAC (Multiplicação-Acumulação) para cada elemento dos filtros de convolução, conforme descrito na primeira linha da Equação 12 Adicionalmente, a integração dos bias dos neurônios é realizada com operações de ACC (Acumulação), uma vez que não requerem a multiplicação com a ativação de entrada. Assim, cada neurônio de saída possui um viés, como mostrado na segunda linha da Equação 12.

$$\begin{aligned} MAC_{OpsConv}^{\mathbf{ANN}} &= C_{in} \times H_{kernel} \times W_{kernel} \times C_{out} \times H_{out} \times W_{out} \\ ACC_{OpsConv}^{\mathbf{ANN}} &= C_{out} \times H_{out} \times W_{out} \end{aligned} \tag{12}$$

Por outro lado, ao se considerar uma SNN, a quantidade de operações realizadas em uma determinada camada depende do número de *spikes* de entrada e saída, denotados como θ_{in} e θ_{out}. Vale ressaltar que os spikes são eventos binários e são integrados usando

operações de ACC (Acumulação), em contraste com o funcionamento das redes neurais convencionais (ANNs).

No que diz respeito aos *spikes* de entrada, cada *spike* desencadeia uma operação de ACC para cada elemento de cada filtro, como indicado no primeiro termo da segunda linha da Equação 13. O segundo termo na equação refere-se à adição de viés a cada potencial da membrana de cada neurônio em cada instante de tempo T e o terceiro termo considera o reset do potencial de membrana a cada emissão de *spike*.

Além disso, as SNNs devem incluir um processo de vazamento do potencial de membrana seguindo o comportamento dos neurônios LIFs. Esse vazamento é modelado por operações MAC adicionais para cada neurônio de saída e ocorre a cada passo de tempo T, como ilustrado no primeiro termo da Equação 13. De maneira análoga, pode se chegar nas expressões matemáticas da Equação 14 para englobar as interações de camadas densas.

$$\begin{aligned} MAC_{OpsConv}^{\mathbf{SNN}} =& T \times C_{out} \times H_{out} \times W_{out} \\ ACC_{OpsConv}^{\mathbf{SNN}} =& \theta_{in} \times \frac{H_{kernel}}{S_h} \times \frac{W_{kernel}}{S_w} \times C_{out} \\ &+ T \times C_{out} \times H_{out} \times W_{out} \\ &+ \theta_{out} \end{aligned} \tag{13}$$

$$\begin{aligned} MAC_{OpsFC}^{\mathbf{ANN}} &= N_{in} \times N_{out} \\ ACC_{OpsFC}^{\mathbf{ANN}} &= N_{out} \\ MAC_{OpsFC}^{\mathbf{SNN}} &= N_{out} \times T \\ ACC_{OpsFC}^{\mathbf{SNN}} &= \theta_{in} \times N_{in} \times N_{out} + N_{out} \times T \end{aligned} \tag{14}$$

Assim, como visto em (CORDONE, 2022), com base nas métricas definidas acima, pode se estimar a energia gasta em operações sinápticas em função de E_{ADD} e E_{MUL}, que são o gasto por unidade de operação de cálculo e multiplicação respectivamente.

$$\begin{aligned} E_{Ops}^{\mathbf{ANN}} &= (E_{ADD} + E_{MUL}) \times MAC_{Ops}^{\mathbf{ANN}} + E_{ADD} \times ACC_{Ops}^{\mathbf{ANN}} \\ E_{Ops}^{\mathbf{SNN}} &= (E_{ADD} + E_{MUL}) \times MAC_{Ops}^{\mathbf{SNN}} + E_{ADD} \times ACC_{Ops}^{\mathbf{SNN}} \end{aligned} \tag{15}$$

2.3.2 Custo de Alocação de Memoria

Para dimensionar o custo associado ao processo de endereçamento de variáveis em redes neurais convencionais e do tipo de SNNs é necessário considerar que na abordagem das ANNs, todas as interações de entrada são estimuladas simultaneamente. Por outro lado, nas SNNs, utiliza-se um método de processamento esparso, em que as sinapses são ativadas de forma dispersa ao longo do tempo.

Assim, nas ANNs, um filtro percorre todas as posições possíveis (dependendo do preenchimento e do passo) na amostra de entrada, gerando um mapa de características de saída denso. Nessas convoluções densas, os cálculos são realizados de forma sequencial, permitindo que os endereços sejam calculados ao incrementar um índice em 1. Isso é possível devido à suposição de que a memória está organizada de forma contígua e na mesma ordem em que é processada. Nesse cenário, temos um índice que percorre a entrada, outro índice que percorre a saída e um terceiro índice que percorre os pesos dos parâmetros como mostra a Equação 16.

$$\begin{aligned} ACC^{\mathbf{ANN}}_{AddrConv} =& C_{in} \times H_{in} \times W_{in} \\ &+ C_{out} \times H_{out} \times W_{out} \\ &+ C_{out} \times H_{kernel} \times W_{kernel} \end{aligned} \tag{16}$$

Já nas SNNs, as convoluções esparsas ocorrem de forma assíncrona, sendo desencadeadas pela recepção de *spikes* de entrada. Portanto, as posições dos endereços dos neurônios de precisam ser calculadas a cada vez que um *spike* de entrada é recebido, pois em uma representação esparsa, os cálculos ocorrem de forma não sequencial, sem conhecimento prévio sobre qual posição de saída será afetada por um *spike* de entrada específico. O custo do endereçamento, medido em termos do número de operações de ACC (Acumulação) e MAC (Multiplicação-Acumulação) pode ser visto na Equação 17.

$$\begin{aligned} MAC^{\mathbf{SNN}}_{AddrConv} =& \theta_{in} \times 2 \\ ACC^{\mathbf{SNN}}_{AddrConv} =& \theta_{in} \times C_{out} \times H_{kernel} \times W_{kernel} \end{aligned} \tag{17}$$

$$\begin{aligned} ACC^{\mathbf{ANN}}_{AddrFC} =& N_{in} + N_{out} \\ ACC^{\mathbf{SNN}}_{AddrFC} =& \theta_{in} \times N_{out} \end{aligned} \tag{18}$$

O cálculo que define a energia necessária para viabilizar os processos citados acima é semelhante ao cálculo da energia consumida para a execução das operações das redes neurais, como visto em (CORDONE, 2022).

$$\begin{aligned} E^{\mathbf{ANN}}_{addr} &= (E_{ADD} + E_{MUL}) \times MAC^{\mathbf{ANN}}_{addr} + E_{ADD} \times ACC^{\mathbf{ANN}}_{addr} \\ E^{\mathbf{SNN}}_{addr} &= (E_{ADD} + E_{MUL}) \times MAC^{\mathbf{SNN}}_{addr} + E_{ADD} \times ACC^{\mathbf{SNN}}_{addr} \end{aligned} \tag{19}$$

2.3.3 Custo de Ocupação de Memória

O dimensionamento dos custos associados à utilização de memória envolve considerações adicionais, uma vez que requer não apenas a leitura dos dados, mas também a alocação e o gerenciamento desses dados como veremos abaixo para cada uma das redes neurais.

Começando pelo cálculo do número de operações para realizar a leitura dos atributos de entrada de uma camada de uma ANN, esta variável é definida pela Equação 20, onde, para uma camada convolucional, cada posição de saída corresponde às operações de leitura de todos os canais de entrada e a todas as posições nas quais o filtro da convolução é aplicado. Por outro lado, em uma camada totalmente conectada (*Fully Conected Layer*), uma camada densa, o número de operações de leitura para os dados de entrada é o próprio número de entradas, denominado como N_{in}. Já para as SNNs, como pode ser visto na Equação 21, é necessário realizar a leitura apenas dos *spikes* emitidos uma vez que é uma variável binaria.

$$\begin{aligned} InRead_{Conv}^{\mathbf{ANN}} &= C_{in} \times H_{kernel} \times W_{kernel} \times C_{out} \times H_{out} \times W_{out} \\ InRead_{FC}^{\mathbf{ANN}} &= N_{in} \end{aligned} \tag{20}$$

$$InRead^{\mathbf{SNN}} = \theta_{in} \tag{21}$$

Para a leitura dos parâmetros dos neurônios, observa-se que, no contexto de uma Rede Neural Artificial (ANN) convolucional, cada saída parametrizada envolve uma interação com todos os canais de entrada e com todos os elementos dos *kernels* aplicados, além dos bias dos neurônios correspondentes a cada saída. Em contraste, para uma ANN densamente conectada, a operação se resume a uma combinação linear dos neurônios na camada de entrada com a saída em questão. Isso também inclui a incorporação dos bias dos neurônios, conforme representado pelo segundo termo da Equação 22.

$$\begin{aligned} ParamRead_{Conv}^{\mathbf{ANN}} &= (C_{in} \times H_{kernel} \times W_{kernel} + 1) \times C_{out} \times H_{out} \times W_{out} \\ ParamRead_{FC}^{\mathbf{ANN}} &= N_{in} \times N_{out} + N_{out} \end{aligned} \tag{22}$$

Em uma camada convolucional baseada em *spikes*, cada *spikes* recebido, representado por θ_{in}, desencadeia uma operação de leitura para todos os filtros de saída e para todas as posições de saída associadas (ou seja, aquelas correspondentes às dimensões do kernel). Além disso, os bias dos neurônios relacionados a todas as posições de saída e filtros também necessitam ser lidos como mostra o segundo termo na Equação 23. Por outro lado, em uma camada totalmente conectada, o número de operações de leitura de parâmetros é similar ao de uma Rede Neural Artificial (ANN) convencional, com a exceção de que os pesos são lidos apenas para todos os *spikes* de entrada θ_{in}. Mais uma vez, os bias dos neurônios ainda devem ser lidos para todos os neurônios de saída, representados por N_{out}.

$$\begin{aligned} ParamRead_{Conv}^{\mathbf{SNN}} =& \theta_{in} \times C_{out} \times H_{kernel} \times W_{kernel} \\ &+ C_{out} \times H_{out} \times W_{out} \\ ParamRead_{FC}^{\mathbf{SNN}} =& (\theta_{in} + 1) \times N_{out} \end{aligned} \tag{23}$$

As SNNs, tem uma necessidade exclusiva de, além de realizar a leitura dos parâmetros dos neurônios, também realizar a leitura e modificação do valor referente ao estado da membrana de cada neurônio da rede neural. Assim a Equação 23 se reflete de maneira idêntica para estes dois processos, descritos nas equações 24 e 25 .

$$\begin{aligned} PotRead_{Conv}^{\mathbf{SNN}} =& \theta_{in} \times C_{out} \times H_{kernel} \times W_{kernel} \\ &+ C_{out} \times H_{out} \times W_{out} \\ PotRead_{FC}^{\mathbf{SNN}} =& (\theta_{in} + 1) \times N_{out} \end{aligned} \tag{24}$$

$$\begin{aligned} PotWrite_{Conv}^{\mathbf{SNN}} =& \theta_{in} \times C_{out} \times H_{kernel} \times W_{kernel} \\ &+ C_{out} \times H_{out} \times W_{out} \\ PotWrite_{FC}^{\mathbf{SNN}} =& (\theta_{in} + 1) \times N_{out} \end{aligned} \tag{25}$$

As equações 26 e 27 se referem à atualização de valores nas saídas de cada *layer*, para isso, são contadas apenas as saídas, onde no caso para as SNNs são contados os *spikes* como θ_{out}.

$$\begin{aligned} OutWrite_{Conv}^{\mathbf{ANN}} &= C_{out} \times H_{out} \times W_{out} \\ OutWrite_{FC}^{\mathbf{ANN}} &= N_{out} \end{aligned} \tag{26}$$

$$OutWrite^{\mathbf{SNN}} = \theta_{out} \tag{27}$$

Com a determinação de todas as métricas desta seção, é possível chegar na energia consumida durante a utilização da memória do sistema como segue na equação abaixo.

$$\begin{aligned} E_{mem}^{\mathbf{ANN}} =& (InRead^{\mathbf{ANN}} + ParamRead^{\mathbf{ANN}}) \times E_{ReadRAM} \\ &+ OutWrite^{\mathbf{ANN}} \times E_{WriteRAM} \\ E_{mem}^{\mathbf{SNN}} =& (InRead^{\mathbf{SNN}} + ParamRead^{\mathbf{SNN}} + PotRead^{\mathbf{SNN}}) \times E_{ReadRAM} \\ &+ (OutWrite^{\mathbf{SNN}} + PotWrite^{\mathbf{SNN}}) \times E_{WriteRAM} \end{aligned} \tag{28}$$

2.4 Bases de dados

Nesta seção, serão detalhados os Datasets utilizados para a execução dos testes de desempenho de classificação de imagens praticados por uma SNN. Ao todo foram testados três bancos de dados, sendo dois deles, o *DVS Gesture* e o *N-CARS*, puramente neuromórficos obtidos diretamente de uma câmera de evento, e um terceiro Dataset que é o resultado da transformação da biblioteca de imagens de *CIFAR10* em uma versão DVS, ou seja, no formato de saída das câmeras de eventos

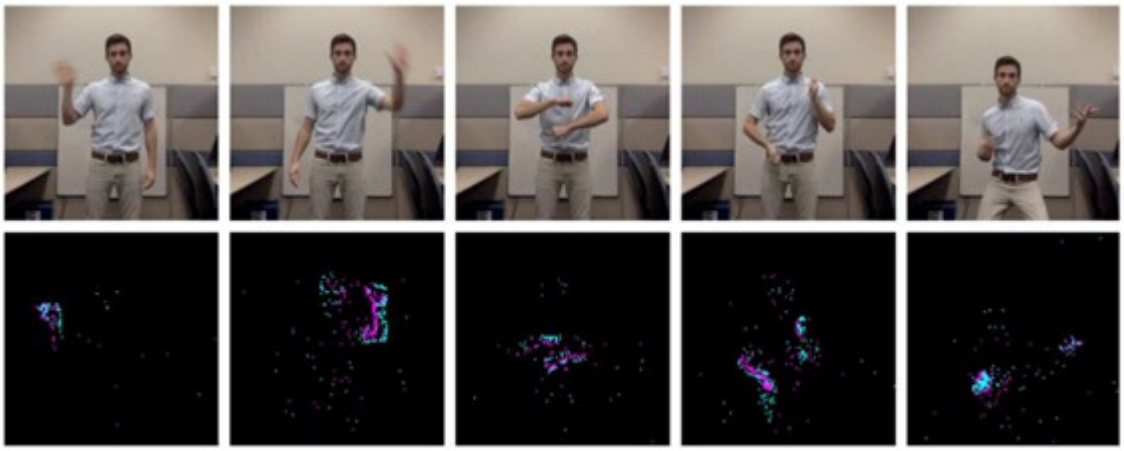
2.4.1 DVS-Gesture

Este banco de dados foi produzido pelo departamento de pesquisa da *IBM Reasearch* (AMIR *et al.*, 2017) utilizando o modelo de câmera de eventos *DVS128* da *iniLabs*, cuja resolução é de 128 x 128 pixels, que oferece uma ampla faixa dinâmica de $120dB$ podendo chegar a enviar 1 milhão de eventos por segundo.

Neste experimento, foram criadas um total de 1,342 instâncias contendo 11 classes de gestos realizados em 3 condições de iluminação diferentes, obtidas com combinações de luz natural, luz fluorescente e luz de LED, especialmente selecionadas para controlar as influências de sombras e flutuações da luz fluorescente sobre os sensores da câmera de evento. Os gestos abrangem uma gama de movimentos (Figura 11), incluindo acenos de mão (com ambos os braços), rotações amplas de braços retos (em ambas as direções, horário e anti-horário), rolamentos do antebraço (para frente e para trás), representações simbólicas de uma guitarra e uma bateria, bem como um gesto rotulado como "Outro"concebido pelo próprio participante.

Cada instância é um elemento que dura até 6 segundos, e são informados todos os eventos em função de (x, y, t, p) sendo a variável t informada em micro-segundos (μs).

Figura 11 – Exemplos de instâncias da base *DVS-Gesture*.



Fonte: (AMIR *et al.*, 2017)

Na Figura 12 é possível identificar os elementos utilizados em (AMIR *et al.*, 2017) para realizar os experimentos, onde uma câmera DVS128, marcada como (1), foi conectada a um microchip neuromórfico, segundo item da figura, produzido pela IBM, chamado de *TrueNorth*, para realizar, em tempo real, a classificação de imagens do dataset DVS-Gesture (AMIR *et al.*, 2017).

Figura 12 – Sistema de coleção e interpretação de Imagens.

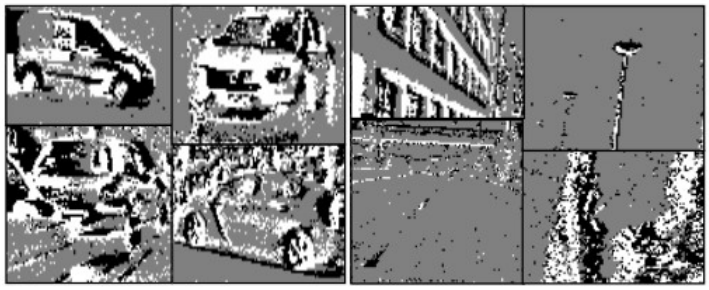
Fonte: Adaptado de (AMIR *et al.*, 2017)

2.4.2 N-CARS

O conjunto de dados N-CARS (SIRONI *et al.*, 2018), criado e disponibilizado pela Prophesee, é uma biblioteca de dados para classificação de imagens composta por 24.000 amostras, cada uma com uma duração de 100 milissegundos (ms), capturadas com uma câmera de eventos do tipo ATIS, modelo Prophesee GEN1, que foi colocada atrás do para-brisa de um carro em movimento durante 80 minutos.

As amostras representam se dividem em apenas duas classes, 12.336 compõem a classe principal de "Carro", e a classe adversa de "Não Carros", que costumam ter ambientes ou planos de fundo, é composta por 11.693 objetos.

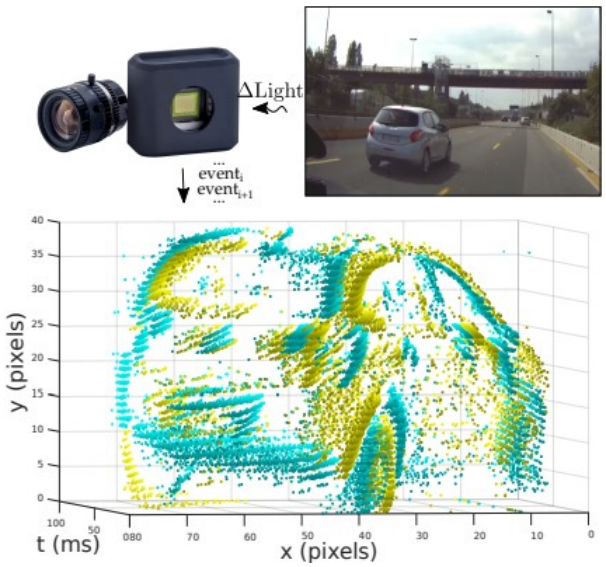
Figura 13 – Exemplos das duas classes encontradas em *N-CARS*.



Fonte: (SIRONI *et al.*, 2018)

As amostras possuem tamanhos variáveis, pois foram obtidas a partir de recortes das imagens originais obtidas pela câmera de resolução de 304 × 240 pixels como mostra a Figura 14.

Figura 14 – Geração do banco de dados *N-CARS*.



Fonte: (SIRONI *et al.*, 2018)

2.4.3 CIFAR10-DVS

A biblioteca *CIFAR10-DVS* (LI *et al.*, 2017) é uma variação do dataset *CIFAR10* criada para expandir a limitada quantidade de datasets obtidos de câmeras de eventos que sejam capazes de proporcionar um certo nível de complexidade para tarefas de classificação de imagens.

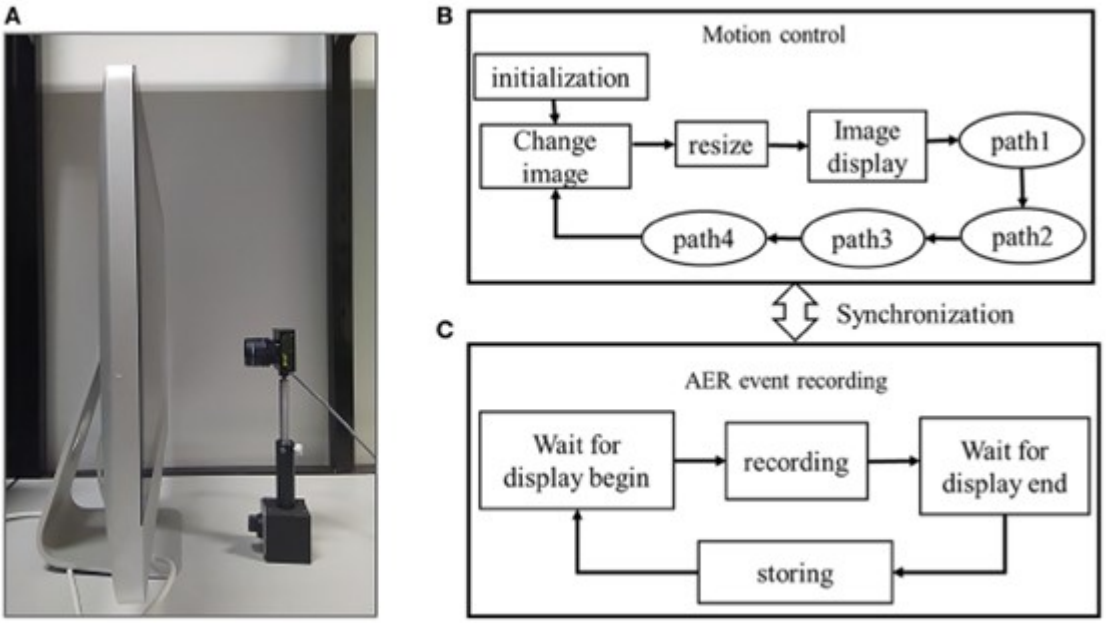
O conjunto original de dados é composto por 60.000 imagens coloridas, de 32 × 32 pixels, contendo 10 classes diferentes, sendo 6.000 imagens para cada classe. A primeira diferença da base convertida se da no número de amostras, pois durante a conversão, foram coletadas 1.000 imagens para cada classe que é equivalente ao conjunto de dados *MNIST-DVS*, porém apresentam estruturas espaço-temporais mais complexas (LI *et al.*, 2017).

Figura 15 – Exemplos de amostras encontradas em *CIFAR10-DVS* junto com os rótulos.

Fonte: (LI *et al.*, 2017)

A resolução do dataset também varia, visto que foi usada uma câmera de evento do tipo DVS (LICHTSTEINER; POSCH; DELBRUCK, 2008) cuja resolução é de 128 x 128 pixels, para isso, as imagens originais foram aumentadas em um computador e gravadas pela câmera assíncrona. Para esta implementação, foi utilizado um movimento repetido em *loop* de maneira lenta e moderada das imagens para que estes movimentos fossem detectados pela câmera neuromórfica como mostra a Figura 16.

Figura 16 – Processo de transformação da base.



Fonte: (LI *et al.*, 2017)

Consequentemente, o conjunto de dados *CIFAR10-DVS* é uma simulação de eventos artificiais, proporcionando uma plataforma propícia para o contínuo aprimoramento dos algoritmos orientados a eventos. A manipulação do deslocamento das imagens é empregada com o propósito de instigar variações na intensidade dentro do âmbito visual da câmera fundamentada em eventos. A emergência de um evento se dá no momento em que a alteração logarítmica local de intensidade excede um limiar preestabelecido. Embora não se trate de um conjunto de dados de natureza estritamente neuromórfica, este repositório apresenta uma ampla gama de categorias que propiciam a investigação tanto das câmeras baseadas em eventos quanto das SNNs, devido à sua complexidade intrínseca mais substancial.

2.5 Trabalhos Relacionados

Nesta seção, serão discutidos os trabalhos relacionados ao tema deste trabalho, que abordam tanto um estudo voltado para a tecnologia das câmeras de eventos, assim como também discutem detalhadamente o processo de construção para configurar um algoritmo do tipo SNN, além dos desafios que este tipo de rede neural apresenta.

2.5.1 Câmeras de Evento

O artigo (GALLEGO *et al.*, 2022) oferece uma extensa introdução às câmeras de evento, explorando a tecnologia empregada na construção desses dispositivos, que utilizam fotossensores, e as diversas aplicações e vantagens que eles oferecem em comparação às câmeras convencionais. O texto discute aplicações como detecção de objetos e reconstrução

de imagens 3D, além de descrever os desafios envolvidos nesses processos. Além disso, o artigo explora opções de processamento de dados provenientes de câmeras de evento, levando em consideração a natureza distinta desse tipo de dado.

Por sua vez, o artigo (GEHRIG *et al.*, 2019) aborda de forma detalhada as opções de pré-processamento dos quadros obtidos por câmeras de evento, trazendo uma comparação dos ganhos e perdas de cada modelo. O texto também discute como esses modelos podem impactar os resultados ao serem utilizados para alimentar algoritmos de aprendizado de máquina.

Em (GEHRIG *et al.*, 2019) se faz um estudo das técnicas de conversão existentes no estado da arte para a extração de eventos, além de propor a sua própria metodologia utilizando matrizes de convoluções, quantizações e projeções, onde cada operação é diferenciável. Assim, ele traz um comparativo dos métodos de extração, destacando o impacto na taxa de acertos aplicados em diferentes datasets, além de realizar uma estimativa de latência de processamento para cada um dos casos.

Esses trabalhos relacionados oferecem uma visão abrangente das aplicações e desafios associados às câmeras de evento e à extração de dados, o que é fundamental para o desenvolvimento de novas soluções em visão computacional orientada à base de eventos.

2.5.2 SNNs

O artigo (WU *et al.*, 2018b) apresenta uma discussão aprofundada dos modelos de SNNs existentes, fornecendo detalhes sobre os atributos de cada neurônio, com destaque para a tensão do neurônio que determina a saída do mesmo (chamada de "s*pikes*"). O texto explora a relação matemática entre esses componentes e camadas de neurônio, apontando a não diferenciabilidade dessa relação matemática como um problema para o aprendizado de máquina. Isso se deve ao fato de que, para ajustar os pesos das camadas dos neurônios, é necessário realizar uma operação de retro-propagação, que depende da diferenciação matemática das operações envolvidas nas camadas da rede neural.

Para contornar isso, nesse artigo, juntamente como em (WU *et al.*, 2018a), propõem alternativas para tornar a função de ativação diferenciável, viabilizando o aprendizado de máquina por meio do calculo do gradiente. Essas propostas foram testadas em tarefas de classificação em banco de dados como o *MNIST*, *N-MNIST* (versão DVS do *MNIST*) e *CIFAR10-DVS*. Os resultados obtidos foram comparados com outros modelos de algoritmos, como LSTM, CNN e outras RNNs. Esses trabalhos relacionados oferecem uma visão abrangente dos desafios e soluções para o uso de SNNs em aprendizado de máquina, e são relevantes para pesquisadores que trabalham com redes neurais.

Além disso, em (CORDONE, 2022) se faz um extenso estudo de SNNs aplicadas tanto à classificação de imagens assim como também à detecção de objetos adaptando

diversas arquiteturas convolucionais. Neste estudo, se propõe uma abordagem diferente de extração de dados, utilizando uma variação do método de *Voxel Grids*, chamada de *Voxel Cubes*, visando a redução de *Time-steps* durante o treinamento da rede neural. Adicionalmente, se faz uma estimativa de gasto computacional, trazendo uma comparação do desempenho de redes neurais convencionais e de redes do tipo SNNs, onde fica em destaque a economia de utilização de recursos gerada quando se opta pela aplicação de uma SNN.

Em (NEFTCI; MOSTAFA; ZENKE, 2019) se faz um estudo para mostrar a equivalência entre Redes Neurais Recorrentes (RNNs) e SNNs, permitindo o aprendizado delas em estruturas populares de Redes Neurais Artificiais usando o *BackPropagation Through Time*. Neste trabalho também foi introduzido o conceito de *Surrogate Gradient* como uma alternativa para viabilizar o aprendizado em Redes Neurais do tipo SNNs. Nesta abordagem, os *spikes* são gerados usando uma função de ativação característica de SNNs, que é uma simples função *Heaviside step function* durante a passagem direta, poém, na passagem de retropropagação, o cálculo do gradiente é determinado usando um gradiente substituto, como por exemplo, o gradiente de uma função sigmoide. Este método se baseia no fato de que a função substituta é suficientemente semelhante à função de *Heaviside step function*, portanto, sua derivada fornece uma aproximação suficiente para o aprendizado com retropropagação.

3 Metodologia

Neste capítulo, serão abordados todos os materiais e processos envolvidos no projeto para realizar a classificação de imagens utilizando imagens do tipo DVS junto com um algoritmo do tipo SNN, o que tem potencial para reduzir o consumo energético e de memória durante todo o processo de captação e interpretação de dados, além de reduzir a latência da captura e interpretação de imagens.

3.1 Materiais e ferramentas

3.1.1 Ferramentas computacionais

Para a construção dos códigos que vão compor o algoritmo de aprendizagem de máquina e o processamento de dados obtidos a partir de câmeras de eventos, foram utilizados duas plataformas, uma delas foi a máquina local com a utilização do Python 3.7.16, e a segunda foi *Google Colab* que conta com a verão de Python 3.10 embutida no seu sistema operacional Ubuntu. Esta ultima plataforma foi utilizada para analisar e experimentar sobre a base CIFAR10-DVS que é um pouco mais complexa e a plataforma oferece a opção de utilização de GPUs, o que agiliza a velocidade do treinamento, além disso oferece diversas bibliotecas embutidas para projetos de ciência de dados, como *Pandas*, *Matplotlib* e *TensoFlow*.

SpikingJelly é uma biblioteca promissora dedicada à construção de *Spiking Neural Networks* (SNNs), cuja arquitetura é fundamentada na ampla biblioteca de *Machine Learning*, *PyTorch*. Assim, são disponibilizadas diversas ferramentas que abrangem tanto a leitura de dados provenientes de câmeras de eventos, no formato DVS, como também o processamento destes dados em redes neurais do tipo SNN.

Uma característica distintiva do *SpikingJelly* é a sua variedade de classes de neurônios neuromórficos, incluindo o *Leaky Integrate-and-Fire* (LIF), o *Parametric LIF*(PLIF), além de outras variações. Essas classes de neurônios desempenham um papel fundamental na composição das camadas da rede neural. O aspecto central que diferencia esses modelos está relacionado à maneira como a tensão da membrana neuronal é dissipada como segue na Equação 9.

3.1.2 Hardware Utilizado

Conforme informado anteriormente, foram utilizadas 2 ambientes virtuais diferentes para a execução dos códigos. Na máquina local, onde foram feitas as aplicações para as bases

N-CARS e*DVS-Gesture* o Hardware utilizado contou com os elementos computacionais listados abaixo:

- *11th Gen Intel(R) Core(TM) i5-1145G7 @ 2.60GHz 1.50 GHz*;
- Memória *RAM* de 16 GB;
- Sem *GPU* dedicada

Enquanto isso, no ambiente execução virtual disponibilizado pela *Google Colab*, onde foram computadas as análises do *CIFAR10-DVS,* o poder computacional é parcialmente escolhido pelo usuário, pois com base nas demandas computacionais, são atribuídas algumas configurações que consigam suportar os requisitos do usuário, assim, neste projeto foi usada a configuração abaixo:

- *CPU Intel Xeon 2.2 GHz*;
- Memória *RAM* de 51 GB;
- Com *GPU* dedicada modelo V100

No que se refere aos modelos de câmeras usados durante a coleção das bases (citadas na Seção 2.4), foram utilizados dois modelos durante a coleção dos 3 datasets, que serão analisados neste projeto, como pode ser conferido na Tabela 1. O modelo produzido pela empresa *iniVation* é um dos primeiros dispositivos desenvolvidos na área, e foi utilizado para a coleção dos dados os bancos e dados DVS-Gesture e CIFAR10-DVS como descrito na Seção 2.4. Nota-se que este modelo, apesar de ser mais antigo, tem um consumo energético menor que os outros citados, pois não conta com uma saída em escala cinza, diferente de outros modelos que proporcionam imagens em sua versão convencional APS (*Active Pixel Sensor*) e também na sua versão DVS (*Dynamic Vision Sensor*), pois isso abre a possibilidade de operar a câmera em ambos formatos, ou numa combinação das duas tecnologias simultaneamente (isto é, uma câmera híbrida). Isso baixa enormemente a barreira para a adoção desta nova tecnologia pela indústria.

Além disso, o modelo ATIS, produzido pela *Prohesse*, já conta com uma maior resolução, além de ter uma latência de 3μs, sendo esta uma das menores dentro dos modelos das câmeras de eventos, também se destaca pela faixa dinâmico (*Dynamic Range*) de 143 dB também sendo um dos maiores quando comparado aos seus competidores. Percebe-se na Tabela 1 que conforme os modelos vão sendo lançados, a areá de cada pixel vai diminuindo, ao passo que a tecnologia desenvolvida consegue produzir este material mais compacto, reduzindo os custos da operação. Assim, a tendência na diminuição da área dos pixels está diretamente ligada também às câmeras híbridas mencionadas anteriormente, pois os pixels do tipo APS estão na faixa de alguns μm^2.

Tabela 1 – Modelos das câmeras de eventos.

Empresa	IniVation	Prophesee	CelePixel
Modelo	DVS128	ATIS	CeleX-V
Dataset Source Device	DVS-Gesture & CIFAR10-DVS	N-CARS	N.A
Ano	2008	2011	2019
Resolução	128 x 128	304 x 240	1280 x 800
Latência (μs)	12μs @ 1klux	3	8
Dynamic Range (dB)	120	143	120
Consumo energético (mW)	23	50-175	400
Área do Chip (mm^2)	6.3 x 6	9.9 x 8.2	14.3 x 11.6
Área do Pixel (μm^2)	40 x 40	30 x 30	9.8 x 9.8
Fonte de Tensão (V)	3.3	1.8 & 3.3	1.2 & 2.5
Gray Scale Output	NO	YES	YES

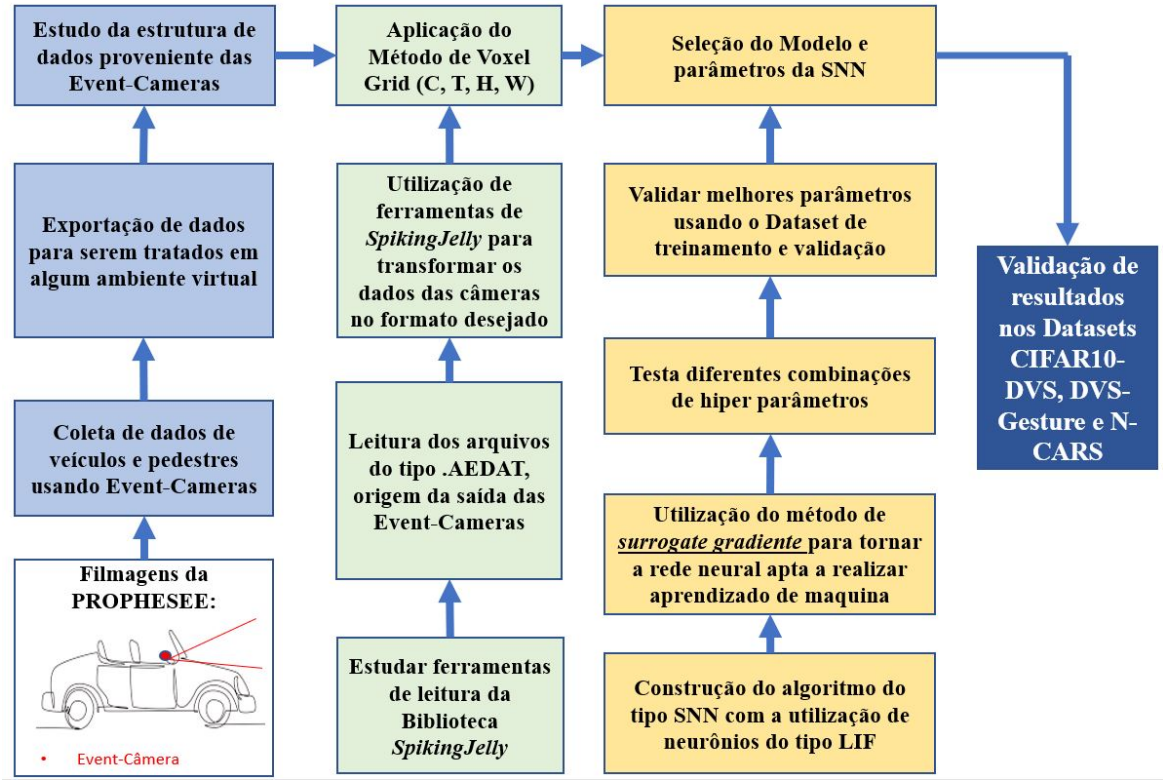
Fonte: O autor.

É importante destacar que durante este projeto, os datasets estudados vieram de fontes externas já produzidas, e o Hardware utilizados para fazer a interpretação de dados foi Hardware convencional, conforme explicitado nas configurações de hardware descritas na Seção 3.1.2. Ainda assim, para implementar este projeto na prática, seria necessário seguir o exemplo da Figura 12.

3.2 Procedimento

Para a implementação do projeto, o processo geral foi semelhante para o estudo dos 3 Datasets, seguindo a estrutura do diagrama da Figura 17 e tendo algumas etapas extras conforme a necessidade para incorporar os experimentos ao mesmo processo que serão detalhadas ao longo deste capítulo.

Figura 17 – Procedimento geral da metodologia.



Fonte: O autor

3.2.1 Estudo e Divisão das Bases de Dados

As bases de dados DVS-Gesture e N-CARs são disponibilizadas com a divisão já determinada dos datasets entre os agrupamentos de treinamento e o de testes, assim, com a ajuda de ferramentas da biblioteca *Torch* (*random split*), o dataset de treinamento é divido entre um sub-set de treinamento e um de validação mantendo a proporção de 80% e 20% respectivamente para executar o aprendizado de máquina ajustando os hiper-parâmetros. Por outro lado, a base CIFAR10-DVS não apresenta nenhuma divisão pre-determinada, assim, optou-se por fazer a divisão dos datasets de treinamento, validação e testes mantendo a proporção de 80%, 10% e 10% respectivamente.

Depois de definir a divisão dos datasets, será feito um estudo estatístico das bases, onde fica em evidência como as classes estão distribuídas para recorte de cada banco de dados, a fim de garantir que nenhuma classe esteja desproporcional em relação aos outros recortes.

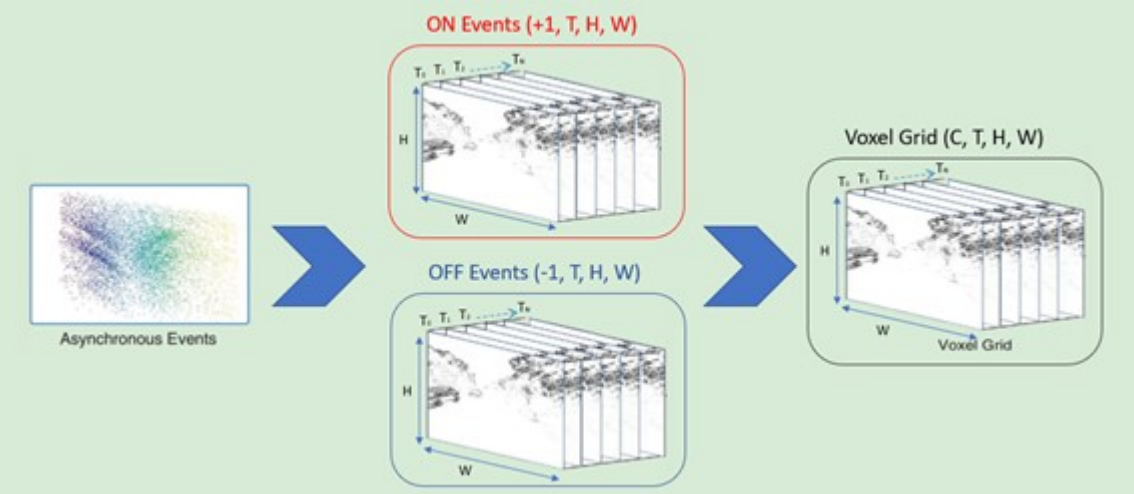
3.2.2 Tratamento das Base de Dados

Como visto na seção 2.1.3, apesar da natureza de dados ser igual, é necessário realizar um pequeno pre-processamento de dados devido à altíssima resolução da câmera de eventos (HE *et al.*, 2020) que pode chegar a micro-segundos (μs). Assim, as 3 bases foram

re-ajustadas para seguirem uma determinada resolução de Δt, na ordem de mili-segundos (ms), utilizando o método de *Voxel Grid*, onde os eventos são colocados no formato (C, H, W, T), sendo C equivalente aos canais (2) que informam as variações positivas e negativas e T o número de passos, que tende a ser maior ao optar por um Δt reduzido. Para isso, é possível utilizar a biblioteca de *SpikingJelly* (FANG *et al.*, 2020), que já conta com algumas ferramentas que automatizam esta transformação diretamente das saídas das câmeras de eventos. Assim, para otimizar a procura de hiper-parâmetros, foi utilizado uma resolução virtual Δt de 50 mili-segundos (ms) na base DVS-Gesture, resultando num menor número de T, enquanto que para a divulgação de resultados com os parâmetros definitivos, foi escolhido um Δt de 10 mili-segundos para aproveitar a alta resolução provinda das câmeras de eventos.

Para o estudo da base CIFAR10-DVS foi utilizado o *Voxel Grid* apenas com um canal, visto que a base de dados é muito pesada em memória, então um canal retém as informações se houve alguma ativação no canal positivo ou negativo da câmera de eventos

Figura 18 – Voxel Grid.



Fonte: O autor

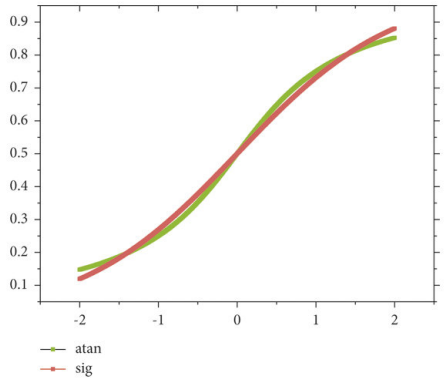
Além disso, como as instâncias do banco de dados N-CARS são de tamanhos H e W variáveis, como explicado na Seção 2.4.2, foi necessário re-dimensionar os tamanhos das amostras para que ficassem em sincronia, já que as entradas da rede neural construída devem ter o mesmo tamanho, desta maneira, uma vez que os datasets DVS-Gesture e CIFAR10-DVS são no formato 128 x 128, optou-se por padronizar este tamanho também para o Dataset N-CARS através do método de interpolação, utilizando a configuração de *"Nearest Neighbor"*.

3.2.3 Escolha e Otimização de Hiper-parâmetros

Para a construção dos LIFs, é necessário definir alguns hiper-parâmetros que vão moldar o comportamento deste neurônio, como a tensão máxima $V_{threshold}$, a taxa de descarga τ, a função utilizada durante o método de *surrogate gradient* já detalhada na Seção 2.2.4.1, além do valor de σ utilizada dentro da função durante o cálculo do gradiente na retro-propagação.

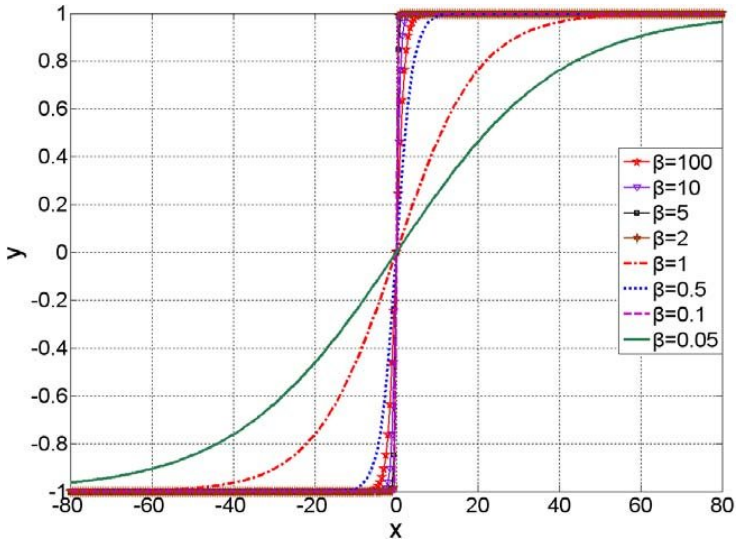
Assim, para a escolha da função utilizada durante o método de *surrogate gradient* foram utilizadas duas funções para os testes, a primeira foi função a *Sigmoid(x, σ)*, amplamente citada como uma função de aproximação da função *heaviside* , e também usamos a função *Atan(x, σ)*, visto que o seu desenho é muito semelhante também como mostra a Figura 19, além disso, para o valor de σ, cuja influência na função de Sigmoide está mostrada na Figura 20, visto que em (CORDONE; MIRAMOND; FERRANTE, 2021) usam $\sigma = 3$, foi decidido fazer a busca de parâmetros tendo $\sigma \epsilon \{3, 5, 9\}$.

Figura 19 – $Sigmoid(x)$ & $Atan(x)$.



Fonte: (XIANGHAN; QUNLI, 2022)

Figura 20 – $Sigmoid(x, \sigma = \beta)$.



Fonte: (YI; LI; ZHAO, 2012)

Por outro lado, para a escolha dos valores de $V_{threshold}$ e τ, foram escolhidos valores que possibilitem a emissão de *spikes*, pois como visto em (PELLEGRINI; ZIMMER; MASQUELIER, 2020), quanto maior forem estes dois valores, menos *spikes* serão emitidos, impossibilitando o aprendizado de máquina. Desta maneira, durante a busca utilizou-se $V_{threshold} \epsilon \{0.3, 0.9, 1.5\}$, e para τ não houve necessidade de escolha manual, visto que foi utilizado uma variação do LIF, chamada de PLIF (*Parametric-LIF*) onde esta variável é um parâmetro otimizado durante aprendizado de máquina. É valido destacar que para realizar a busca dos hiper-parâmetros de maneira otimizada foi utilizada o método de *HyperBand* (LI *et al.*, 2018)

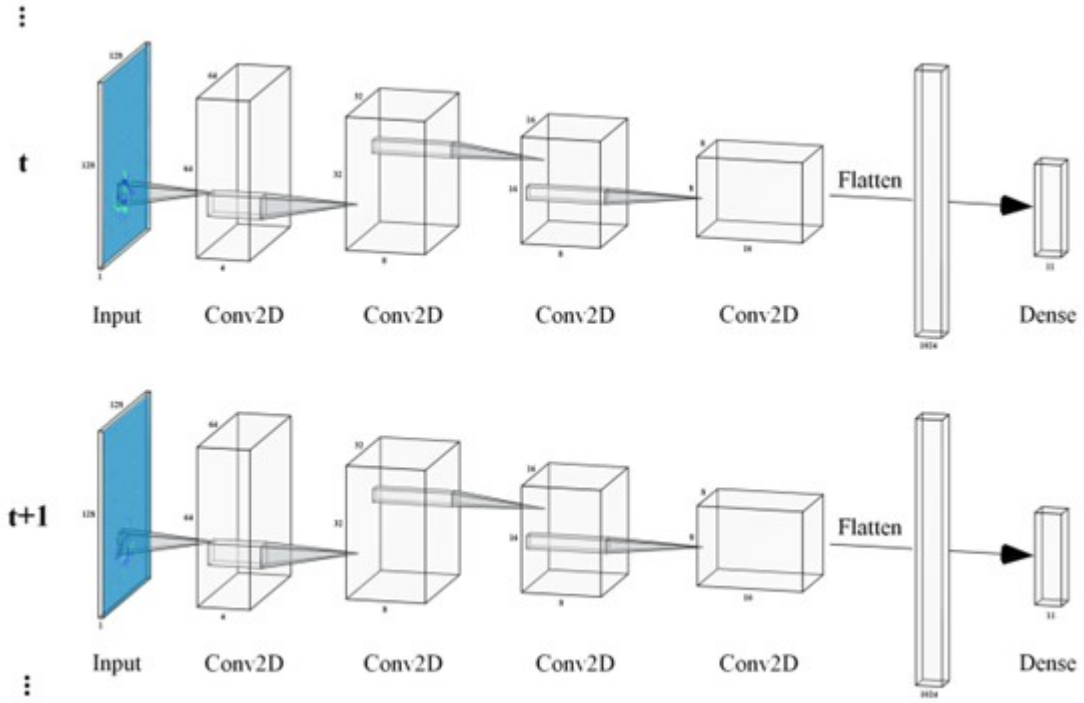
Além desses parâmetros, o otimizador escolhido para o aprendizado de máquina foi o otimizador *Adam* com as configurações pre-determinadas, e a taxa de aprendizado *learning rate* inicial é de 10^{-3}, que vai diminuindo em função das *epochs*, seguindo o método proposto em (LOSHCHILOV; HUTTER, 2017), porém considerando apenas um período. Finalmente, para a função de perda (*loss funciton*) que vai computar os erros da rede neural, foram adotadas as funções *Cross Entropy loss* e *Mean Squared Error*, pois se trata de uma tarefa de classificação de múltiplas classes.

3.2.4 Seleção das Topografias e Escolha da Topografia Final

Foram analisados os trabalhos que tivessem sido testados em cima do DVS-Gesture, por se tratar de um dataset puramente neuromórfico e amplamente conhecido no seu campo de pesquisa, e que também tivessem sido construídos utilizando a lógica de uma rede neural do tipo SNN, desta maneira foram avaliadas os seguintes relatórios: (FANG *et al.*, 2021), (KAISER; MOSTAFA; NEFTCI, 2020), (CORDONE; MIRAMOND; FERRANTE, 2021), (XING; CATERINA; SORAGHAN, 2020), (SHRESTHA; ORCHARD, 2018).

Apesar de todos os trabalhos citados acima conseguirem uma taxa de acerto de mais de 90%, optou-se por utilizar a base estrutural da rede neural utilizada em (CORDONE; MIRAMOND; FERRANTE, 2021), pois conseguiu chegar em resultados semelhantes utilizando muito menos iterações e parâmetros. Para isso, foram utilizados 4 camadas de redes convolucionais, seguidas das camadas dos neurônios PLIFs, um para cada rede convolucional, formando uma rede neural SCNN (*Spiking Convolutional Neural Network*) como pode ser visto na Figura 21, onde se faz uma predição de classificação para cada instante de tempo t.

Figura 21 – Camadas da SNN Convolucional.



Fonte: (CORDONE; MIRAMOND; FERRANTE, 2021)

É importante destacar também que para cada par de camadas SCNN (Spiking + Convolutional Neural Network) foi colocado também um *layer* de *batch normalization* para cada paridade, pois como veremos na seção 4.2.3, é necessário colocar este *layer* para viabilizar o aprendizado de máquina das SNNs, além disso, será analisado como a ausência e a ordem de colocação deste *layer* em relação às SCNN pode impactar o no resultado final da predição.

Por outro lado, no contexto do estudo de classificação de imagens da base CIFAR10-DVS, que apresenta um desafio mais substancial, optou-se por empregar uma derivação da arquitetura VGG11 (Figura 6), denominada *Tiny-VGG11*. Conforme sugere o nome, esta versão é uma abordagem reduzida que incorpora um número inferior de parâmetros. Ela se caracteriza por dispor de uma quantidade de filtros quatro vezes menor em comparação com o modelo original. A concepção do Tiny-VGG11 visa atender a sistemas com capacidade

computacional limitada. Adicionalmente, para a adaptação desta arquitetura em *Spiking Neural Networks* (SNNs), as camadas que normalmente empregam funções de ativação *ReLU* foram substituídas por neurônios do tipo PLIFs.

3.2.5 Estimativa de Consumo de Potência

Para a utilização das equações descritas na Seção 2.3.3, é necessário considerar algumas variáveis definidas pela tecnologia onde os processos são executados. Essas variáveis convertem os números quantificados dos processos das redes neurais em unidades de Joules. Assim, optou-se por utilizar a mesma tecnologia CMOS de 45nm, com variáveis de 32 bits, conforme descrito em (CORDONE, 2022). Nessa tecnologia, os valores são os seguintes:

- O consumo de energia para uma operação de adição (E_{ADD}) é equivalente a $0.1pJ$.
- O consumo de energia para uma operação de multiplicação (E_{MUL}) é equivalente a $3.1pJ$.
- Para determinar os valores de $E_{ReadRAM}$ e $E_{WriteRAM}$, foi utilizado um valor unitário obtido a partir de uma função de interpolação linear dos seguintes valores: $8kB$ ($10pJ$) e $32kB$ ($20pJ$).

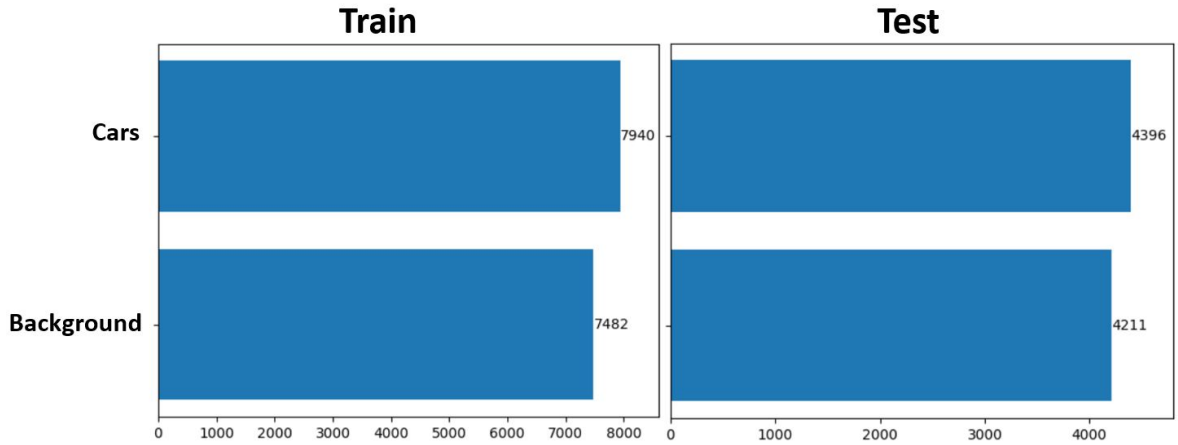
4 Análise de Resultados

Após aplicar os métodos e tratamentos de dados visto na seção da metodologia, foi feita a análise de resultados, conforme segue nas seções.

4.1 Estudo estatístico das bases de dados

A base de dados da N-Cars é a mais simples, visto que têm apenas 2 classes (*Background e Cars*), cujas proporções são mantidas, tanto no set de treinamento, assim como no set de testes como consta na Figura 22, onde aparecem as atribuições dos elementos que totalizam 15,422 e 8,607 ocorrências nos sets de treinamento e testes, respectivamente. Assim, vale ressaltar que a base de treinamento foi novamente dividida em uma sub-classe de treinamento, mantendo 80% das amostras originais, e outra de validação com o restante, o que resultou numa divisão de 51%, 13%, 36% entre os 3 datasets (treinamento, validação e testes).

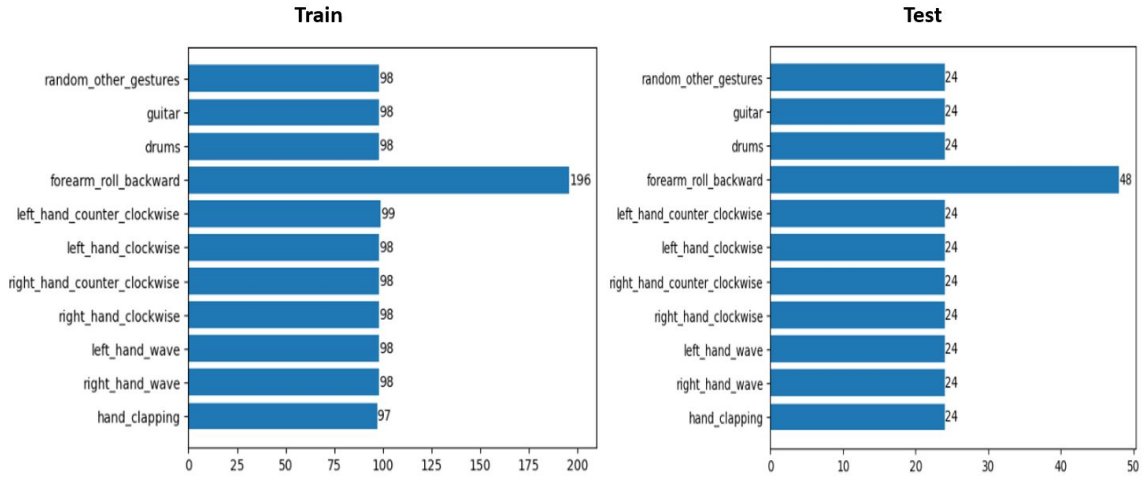
Figura 22 – Distribuição de Classes N-Cars.



Fonte: O autor

Além disso, a outra base puramente neuromórfica do relatório é a base DVS-Gesture, que também já vem com a divisão pre-determinada entre 1,176 amostras destinadas ao treinamento da rede neural, das quais 235 foram usadas como validação de resultados, e mais 288 amostraras para a execução de testes. Esta base contêm uma variedade maior de classes como mostra a Figura 23, onde há uma classe com uma representatividade maior, pois os movimentos representados por uma classe referente ao antebraço a princípio eram divididos pela direção do braço, mas acabaram sendo reportados como uma classe só, como consta na documentação da fonte (AMIR *et al.*, 2017). Apesar disso, a proporção das classes é mantida igual para os dois datasets fornecidos.

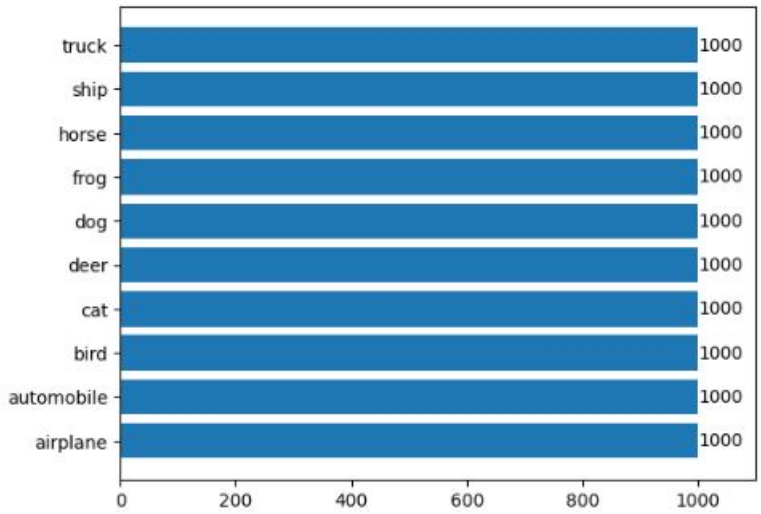
Figura 23 – Distribuição de classes em *DVS-Gesture*.



Fonte: O autor

Por último, a base que mistura complexidade com um número um pouco maior de amostras, é a base CIFAR10-DVS, que contém 1000 amostras das classes do dataset original CIFAR10, sendo o único dataset do projeto que não é fornecido com uma divisão previa de sub-sets, assim, neste projeto, a base foi dividida de maneira aleatória em porções de 80%, 10% e 10% para os agrupamentos de treino, validação e testes.

Figura 24 – Distribuição de Classes em CIFAR10-DVS.



Fonte: O autor

4.2 Estudo da Otimização de uma SNN

Como visto na seção da metodologia, para construir uma SNN é necessário levar em conta vários fatores, desde a utilização de *layers* de *Batch Normalization*, assim como também a escolha de hiper-parâmetros e funções de otimização que vão fornecer uma convergência mais rápida, detalhes que serão revisados nas seções abaixo.

4.2.1 Hiper-parâmetros

Para a escolha dos hiper-parâmetros, foi utilizado o método de procura *HyperBand* onde à medida que todas as combinações vão sendo testadas, as que têm o pior desempenho vão sendo anuladas antes de chegar no número de *epochs* final. Assim os valores iniciais da procura dos hiper-parâmetros pode ser analisada na Tabela abaixo:

Tabela 2 – Opção dos hiper-parâmetros.

Parâmetros	Opções
Initial Tau	[2.]
$V_{threshold}$	$[0.3, 0.9, 1.5]$
Surrogate Function	$[Sigmoid(\sigma), Atan(\sigma)]$
σ	$[3., 5., 9.]$
Função de Otimização	$[MSE, CrossEntropy]$

Fonte: O autor.

É valido destacar que os testes foram feitos nas bases DVS-Gesture e CIFAR10 por representar uma maior complexidade para a tarefa de classificação de imagens, além disso, foram aplicados no máximo 10 *epochs* para a seleção dos hiper-parâmetros. A configuração com o melhor desempenho para cada banco de dados pode ser consultada abaixo, e estas configurações vão ser utilizadas durante os outros experimentos apresentados posteriormente.

Tabela 3 – Resultados dos hiper-parâmetros.

Banco de dados	DVS-Gesture (Simple SCNN)	CIFAR10-DVS (Simple SCNN)
Initial Tau	2.	2.
$V_{threshold}$	0.9	0.9
Surrogate Function	$Sigmoid(\sigma)$	$Atan(\sigma)$
σ	5.	5.
Função de Otimização	MSE	MSE

Fonte: O autor.

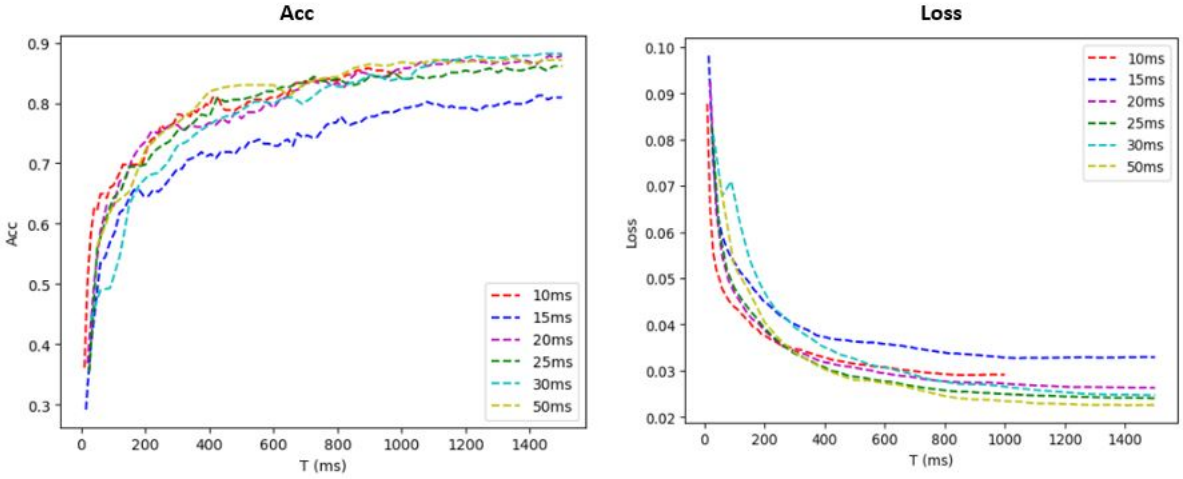
O valor da constante de tempo inicial (τ) não foi alterado, pois é um parâmetro treinável, logo, é alterado automaticamente durante o treinamento da rede neural. Além disso percebe-se que os resultados de ambos os testes é muito parecido, divergindo em apenas a função *Surrogate Function* que não é dos fatores mais importantes visto que a substituição de uma função pela outra oferece resultados muito similares.

4.2.2 Resolução e *Time-Steps*

Como visto na Seção 2.1.3, ao tratarmos os eventos na forma (C, T, H, W), é necessário escolher uma resolução virtual Δt para a leitura dos dados das câmeras de eventos. Na Figura 25, é possível perceber a ação dos neurônios LIFs em diferentes cenários

de Δt é semelhante, onde quanto maior a duração da amostra, maior a taxa de acerto dos resultados, uma vez que o estado dos neurônios num dado instante t vai afetar a saída em momentos posteriores também.

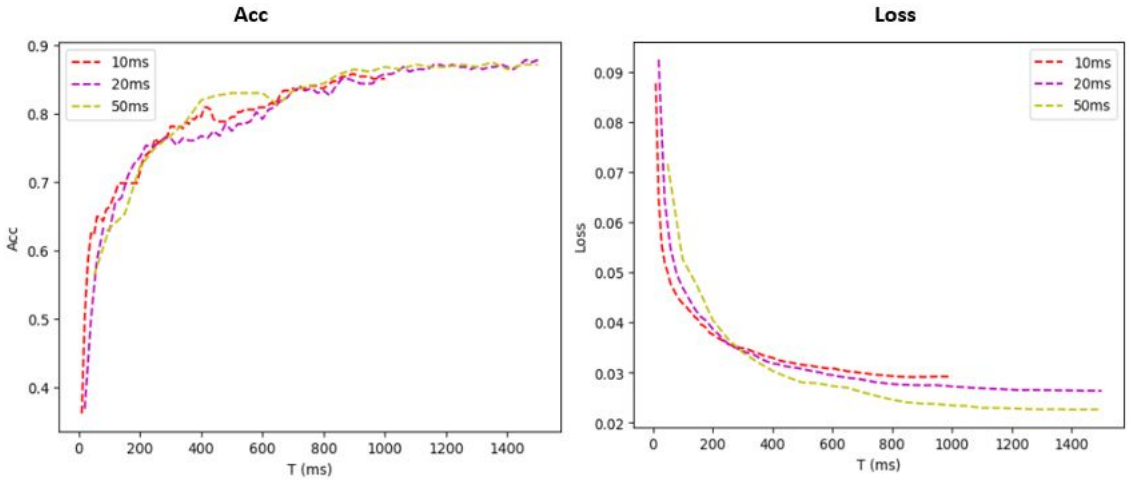
Figura 25 – Efeito da escolha da duração Δt de cada *Time-step*.



Fonte: O autor

Além disso, é possível perceber que nos primeiros instantes de avaliação, SNN treinadas com Δt têm uma melhor avaliação, ao passo que recebem um número maior de quadros em intervalo de tempos menores, enquanto que SNNs treinadas com Δt maiores tendem a se sobressair no longo prazo, uma vez que os *frames* destas SNNs retêm mais informação.

Figura 26 – Efeito da escolha da duração Δt de cada *Time-step*, opções reduzidas.



Fonte: O autor

4.2.3 Batch-Normalization

Ao realizar a classificação de imagens, os 3 Datasets foram submetidos em 3 cenários diferentes em relação ao uso do *layer* de *Batch Normalization*, de maneira que pudesse

ser estudado o efeito do posicionamento desta camada em relação às camadas do tipo SNN. Assim, como pode ser visto na Tabela 4, é imprescindível utilizar este tipo de *layer* para o treinamento e aplicação de SNNs, visto que na ausência deste o treinamento obtém resultados piores em relação aos casos com *Batch-normalization*, Além disso, alguns estudos sugerem que posicionar este *layer* antes das SCNNs pode causar uma melhora no desempenho (CORDONE; MIRAMOND; THIERION, 2022), no entanto, durante a implementação do modelo Tiny-VGG11 este método acabou desfavorecendo o aprendizado de máquina.

Tabela 4 – Efeito da camda Batch-Normalization.

Datasets & SNN Model	BN-SCNN	SCNN-BN	No BN	epochs	$\Delta t(ms)$
N-CARS (Simple SCNN)	**90,36%**	88,93%	88,15%	10	10
DVS-Gesture (Simple SCNN)	**87,15%**	**87,15%**	8,33%	15	30
CIFAR10-DVS (Tiny-VGG11)	15,2%	**64,80%**	10,8%	30	100

Fonte: O autor.

4.2.4 Desempenho da rede em função da *loss function*

A função de perda tem a tarefa de computar os erros baseada em alguma métrica pre-determinada e a partir destes erros, através do cálculo do gradiente, otimizar os parâmetros de aprendizado para melhorar o desempenho da rede neural. Num principio foram utilizadas as funções MSE (*Mean Square Error*) e *Cross Entropy function* por meio da biblioteca *Pytorch* e foram obtidos os resultados abaixo.

Tabela 5 – Efeito da Função de Otimização.

Datasets & SNN Model	MSE	Cross Entropy	epochs	$\Delta t(ms)$	*Time-steps*
N-CARS (Simple SCNN)	**90,36%**	88,61%	10	10	10
DVS-Gesture (Simple SCNN)	**87,15%**	70,14%	15	30	50
CIFAR10-DVS (Tiny-VGG11)	**64,80%**	38,50%	30	100	13

Fonte: O autor.

Observa-se que, ao realizar uma tarefa de classificação de imagens, a função *Mean Squared Error* (MSE) tende a convergir mais rapidamente em comparação com a função de *Cross Entropy*. Essa diferença de convergência pode ser atribuída à maneira como as redes neurais convencionais são configuradas para lidar com a classificação de imagens, pois normalmente, ao realizar a tarefa de classificação de imagens em redes neurais convencionais, a função de ativação *SoftMax* é usada na camada de saída final. No entanto, na prática, ao calcular a função de perda *Cross Entropy*, a função *SoftMax* já é incorporada à função de perda. Isso significa que não é necessário adicionar explicitamente uma camada *SoftMax* à rede neural, uma vez que a função *SoftMax* é tratada durante o cálculo da função de perda.

Essa abordagem evita a duplicação do cálculo da função *SoftMax*, o que poderia atrasar a convergência, como indicado pelos resultados na tabela, pois na prática, o uso de neurônios LIFs (neurônios do tipo Leaky Integrate-and-Fire) é equivalente à aplicação da função *SoftMax* na saída final da rede neural. Isso pode explicar por que a função MSE converge mais rapidamente em comparação com a função de *Cross Entropy* nos testes realizados neste projeto.

4.3 Analise e Comparação dos Resultados Finais Obtidos

4.3.1 N-CARS

O dataset de N-Cars, por ter apenas duas classes, proporcionou menos dificuldade para o treinamento de redes neurais, e por isso foi possível atingir resultados perto dos 90% de taxa de acerto, resultados semelhantes a outros estudos, porém neste projeto foi utilizada uma rede neural muito mais simples, logo com um menor número de parâmetros. É valido destacar que neste relatório, o tamanho das amostras também foram re-dimensionadas para ser de 128 x 128 de resolução, enquanto em outros projetos a resolução foi de 62 x 62. Na Tabela 6 é possível ver os resultados obtidos em outros trabalhos sob a utilização de diferentes formatos extraídos das câmeras de eventos, e por se tratar de um dataset com apenas duas classes, a maioria dos trabalhos atingiu uma taxa de acerto maior que 90%.

Tabela 6 – Comparação de Resultados N-CARS.

Métodos	Event-Data format	N-CARS Acc
HATS	TimeSurface	90,2%
Gabor-SNN	Spike	78,9%
YOLE	Voxel Grid	92,7%
Asynet	Voxel Grid	**94,4%**
VGG11	Voxel Cube	92,4%
Simple SCNN (This Work)	Voxel Grid with two Channels	90,1%

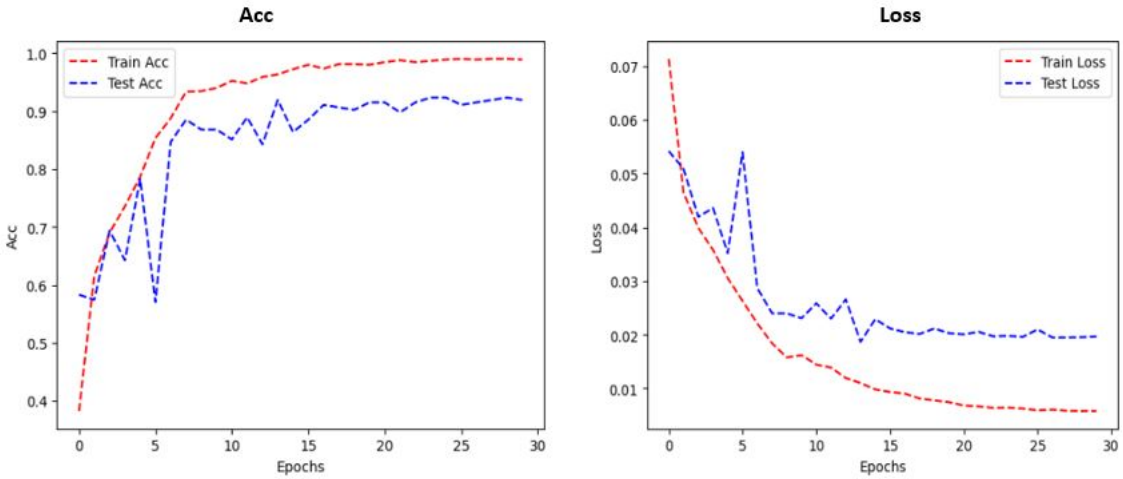
Fonte: O autor.

4.3.2 DVS-Gesture

A base de DVS-Gesture já se torna mais interessante para a análise de resultados por tem uma maior variedade entre as classes a serem classificadas, e também pelas amostras terem uma duração maior que os outros datasets estudados neste projeto, e claro, por ser de origem puramente neuromórfica. Na Figura 27 é possível acompanhar o aprendizado de máquina em função de cada *epoch* através das métricas de taxa de acerto das predições e também do cálculo da função de perda (MSE). Assim, é possível perceber que a rede neural converge quase por completo após 20 *epochs*, já se chegam em ótimos resultados tanto no dataset de treino assim como no de validação. É importante destacar

que, apesar do número reduzido de amostras, foi possível chegar em tais resultados, pois o dataset foi coletado dentro de um laboratório, proporcionando amostras limpas e com muito pouco ruído.

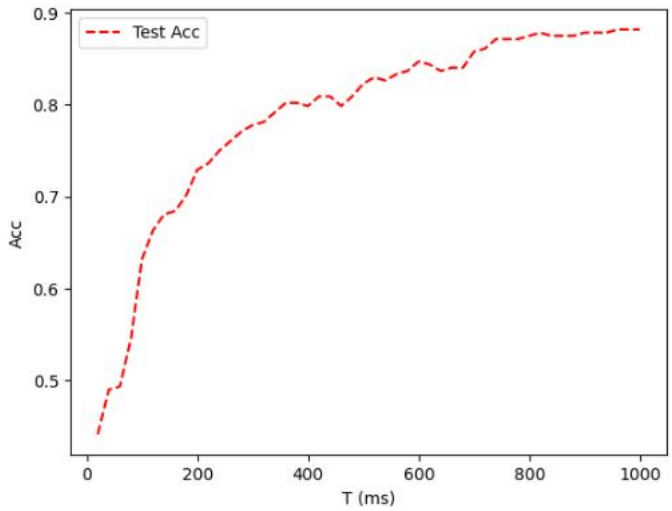
Figura 27 – Taxa de acerto e perdas em função das *epochs*.



Fonte: O autor

Além disso, foi realizado um teste de classificação para cada *Time-Step*, e é possível perceber a atuação dos neurônios LIFs, pois a passo que o tempo vai fornecendo dados, ele vai aumentando o seu desempenho.

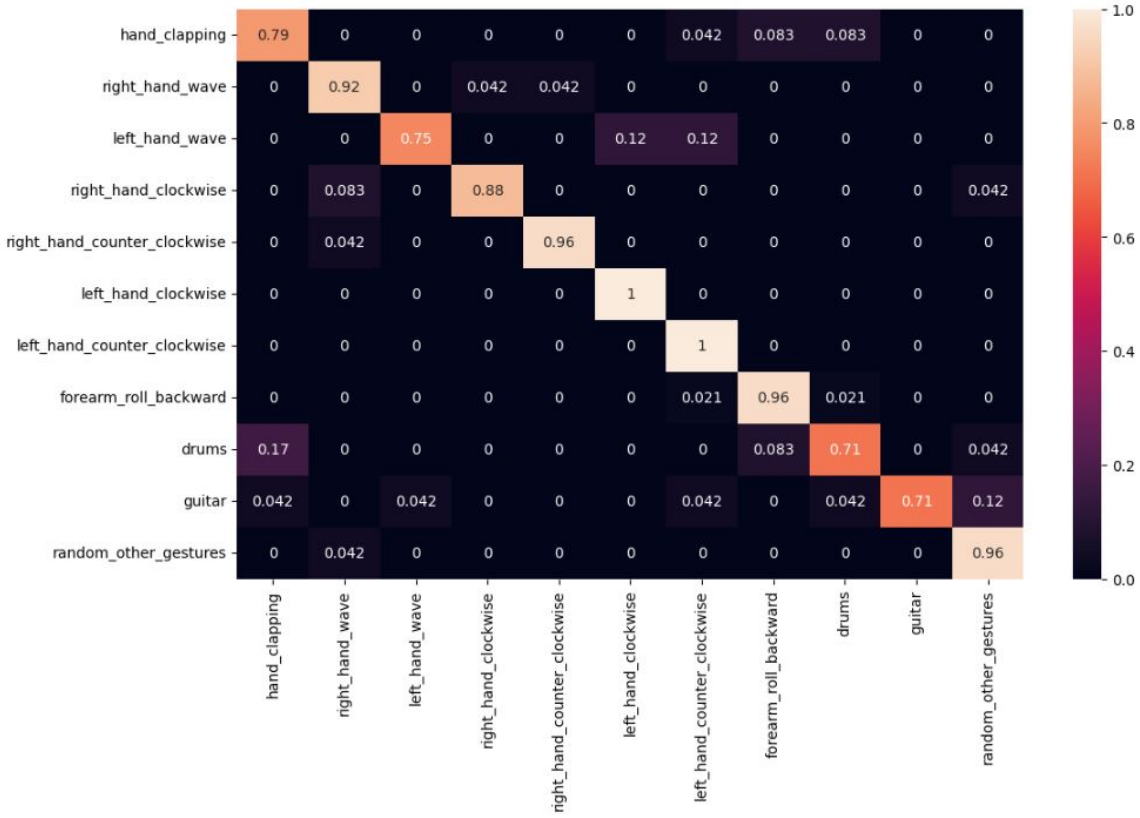
Figura 28 – Taxa de acerto em função de T.



Fonte: O autor

Por último, foi construída uma matriz de confusão contendo a taxa de acerto para todas as classes do dataset, além das predições erradas feita pela rede neural. É possível perceber que os erros se apresentam principalmente em movimentos executados pela mesma mão, e também as classes onde as pessoas tentam imitar um instrumento musical são as que são mais difíceis de serem corretamente detectadas pela rede neural.

Figura 29 – Matriz de confusão.



Fonte: O autor

Na Tabela 7 é possível comparar os resultados deste relatório com outros trabalhos do estado da arte, e percebe-se que foi possível atingir um bom desempenho com um número reduzido de interações durante o treinamento, além de um número reduzido de parâmetros.

Tabela 7 – Comparação de resultados DVS-Gesture.

Métodos	#Param	#Interações	Acc
SLAYER	N.A	$270k$	0,9364
SCRNN	$> 662k$	100 epochs	0,9201
DECOLLE	$> 51k$	$160k$	0,9554
PLIF SNN	$> 1.110k$	$74k$	0,9757
Simple-SCNN (This Work)	$> 210k$	$0,28k$	0,8819

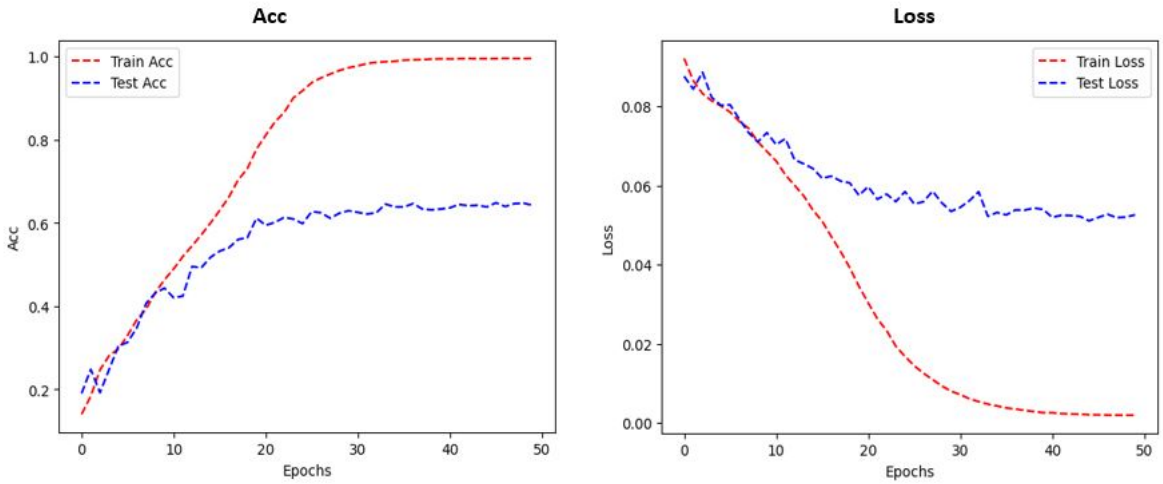
Fonte: O autor.

4.3.3 CIFAR10-DVS

A base CIFAR10-DVS também proporciona boas analises de resultados por ser bem conhecida dentro dos estudos de redes neurais convencionais e também por apresentar elementos do cotidiano, como carros e caminhões que são dois elementos de grande interesse dentro do proposito de carros autônomos e a sua autopercepção do espaço. Na Figura

30 é possível acompanhar o aprendizado de máquina em função de cada *epoch* através das métricas de taxa de acerto das predições e também do cálculo da função de perda (MSE), e diferente dos resultados obtidos com a base de dados de DVS-Gesture, onde os datasets de treino e validação chegam em resultados parecidos, aqui o dataset de validação fica estancado na faixa de 60% de resultados corretos enquanto o dataset de treinamento consegue chegar acima dos 90% depois de 30 *epochs*.

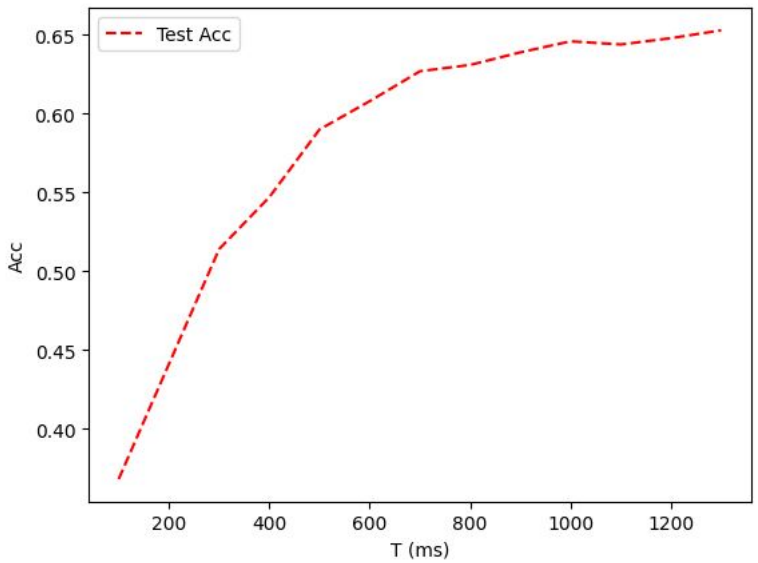
Figura 30 – Taxa de acerto e perdas em função das *epochs*.



Fonte: O autor

Na Figura 31, nota-se mais uma vez como o tempo T é uma variável que afeta no resultado das SNNs, pois o estado dos neurônios influencia na predição de classes de etapas posteriores, aumentando o desempenho da rede neural.

Figura 31 – taxa de acerto em função de T.



Fonte: O autor

Assim, pode se concluir que a SNN implementada foi capaz de realizar o aprendizado de máquina por meio dos neurônios LIFs, porém acabou super ajustando os seus parâmetros ao dataset de treinamento. Isso se deve ao fato da base tratada ser uma base reduzida da base original CIFAR10, o que diminui a variedade de características a ser captada pela rede neural, ainda assim, se ressalta a taxa de acuratividade acima dos 70% para a detecção de veículos terrestres (carros e caminhões).

Figura 32 – Matriz de confusão.

Fonte: O autor

Na Tabela 8 estão listados os resultados deste estudo e de outros relatórios do estado da arte, e nota-se a dificuldade de atingir uma taxa de acertos maior que 70%. Possivelmente por se tratar de um dataset com menos amostras, as redes neurais dão sinais de estarem dentro do quadro de *overfitting*.

Tabela 8 – Comparação de Resultados CIFAR10-DVS.

Métodos	CIFAR10-DVS Acc
Gabor-SNN (SIRONI *et al.*, 2018)	0,245
HATS (SIRONI *et al.*, 2018)	0,524
SPA (LIU *et al.*, 2020)	0,322
Natural SNN (DENG *et al.*, 2020)	0,603
STS-ResNet (SAMADZADEH *et al.*, 2021)	**0,692**
Tiny-VGG11 (This Work)	0,659

Fonte: O autor.

4.4 Estudo de Desempenho da Rede e Gasto Energético

Nesta seção serão analisadas os desempenhos das redes neurais convencionais e SNNs quando as duas utilizam a mesma arquitetura para performar a classificação de imagens. A Tabela 9 mostra para cada um dos datasets, como foi o desempenho de cada uma das classes de rede neurais. É possível perceber que a arquitetura de redes convencionais consegue melhores resultados nas duas bases que são puramente neuromórficas, porém, na base CIFAR10-DVS que possui um grau de complexidade maior, a rede neural SNN obtém melhores resultados.

Tabela 9 – Comparação de Resultados CNNs vs SNNs,

Dataset	Arquitetura	Rede Neural	Param	Acc.	Taxa de Spikes
CIFAR10-DVS	Tiny-VGG11	ANN(Relu)	10,03M	0,406	-
CIFAR10-DVS	Tiny-VGG11	SNN(LIF)	10,03M	**0,659**	0,124
DVS-Gesture	Simple CNN	ANN(Relu)	206,3k	**0,872**	-
DVS-Gesture	Simple SCNN	SNN(LIF)	206,3k	0,865	0,043
N-CARS	Simple CNN	ANN(Relu)	206,3k	**0,9016**	-
N-CARS	Simple SCNN	SNN(LIF)	206,3k	0,880	0,236

Fonte: O autor.

É valido destacar que nos resultados da CNN convencional sobre a base CIFAR10-DVS, a rede neural não tinha chegado na sua convergência total, e com um maior número de *epochs* poderia se aproximar dos resultados da SNNs, que por sua vez já tinha obtido a convergência integral do seu treinamento de parâmetros. Assim, percebe-se que tanto as redes neurais convencionais e as do tipo SNNs são capazes de trabalhar com dados produzidos por câmeras de evento. Desta maneira, o desempenho por si só não justifica a substituição de redes convencionais por SNNs na aplicação de tarefas de classificação de imagens. Por isso, com base nas equações da Seção 2.3.3, foram calculadas estimativas de custos energéticos para executar os processos de cada uma das redes neurais, chegando-se nos valores da Tabela 10.

A diferença significativa no consumo de energia entre as Redes Neurais Artificiais (ANN) e as SNNs nos conjuntos de dados apresentados na tabela pode ser atribuída, em grande parte, à maneira como as equações de custo e uso de memória estão intrinsecamente relacionadas ao número de *spikes* emitidos durante todo o processo. Pois, como podemos observar na tabela 11, a proporção de atividades não nulas (que é equivalente aos *spikes* disparados) em redes neurais, em geral, varia de 20% a 4%. Isso significa que, em ANN, onde o processamento ocorre de maneira contínua, a energia é consumida de maneira mais uniforme, enquanto em SNN, onde os *spikes* são discretos e eventos pontuais, a energia é gasta principalmente durante esses momentos de atividade intensa.

Tabela 10 – Projeção do Custo Energético.

Dataset	E	E_{SNN} (nJ)	E_{ANN} (nJ)	$\frac{E_{ANN}}{E_{SNN}}$
CIFAR10-DVS	Utilização de memória	$6,10e6$	$3,47e7$	5,695
CIFAR10-DVS	Operações sinápticas	$1,80e6$	$7,88e6$	4,368
CIFAR10-DVS	Operações de alocação	$3,11e4$	$1,31e3$	0,042
CIFAR10-DVS	Total	$7,93e6$	$4,26e7$	**5,371**
DVS-Gesture	Utilização de memória	$3,18e4$	$4,04e5$	12,724
DVS-Gesture	Operações sinápticas	$7,95e4$	$9,13e4$	1,149
DVS-Gesture	Operações de alocação	$1,56e2$	$9,50e1$	0,612
DVS-Gesture	Total	$1,11e5$	$4,96e5$	**4,449**
N-CARS	Utilização de memória	$1,15e5$	$4,04e5$	3,517
N-CARS	Operações sinápticas	$2,52e4$	$9,13e4$	3,627
N-CARS	Operações de alocação	$6,87e2$	$9,5e1$	0,139
N-CARS	Total	$1,41e5$	$4,96e5$	**3,520**

Fonte: O autor.

Tabela 11 – Taxa de Atividades Non-Zero em SNNs para cada marco de tempo T.

Simple SCNN Time-steps (T)	N-CARS 10	DVS-Gesture 50	Tiny-VGG11 Time-steps (T)	CIFAR10-DVS 13
Input Images	1.506 (4, 6%)	1.010 (3, 08%)	Input Images	7.013 (42, 81%)
SCNN1	4.875 (30, 71%)	497 (3, 13%)	SCNN1	45.679 (17, 43%)
SCNN2	905 (12, 58%)	327 (4, 55%)	SCNN2	25.061 (19, 12%)
SCNN3	122 (6, 82%)	72 (4, 00%)	SCNN3	17.213 (6, 57%)
SCNN4	135 (17, 23%)	130 (16, 62%)	SCNN4	7.475 (11, 41%)
FC1	76 (29, 90%)	77 (30, 27%)	SCNN5	11.469 (8, 75%)
FC2	1 (49, 78%)	1 (9, 48%)	SCNN6	4.447 (13, 57%)
-	-	-	SCNN7	3, 474 (10, 60%)
-	-	-	SCNN8	433 (5, 30%)
-	-	-	FC1	93 (9, 12%)
-	-	-	FC2	62 (6, 13%)
-	-	-	FC3	1 (9, 47%)
Total da rede	6.114 (23, 61%)	1.105 (4, 26%)	Total da rede	115.413 (12, 44%)

Fonte: O autor.

Essa tendência torna-se clara quando se examina o consumo energético em operações sinápticas, revelando que o gasto de energia nessas operações foi proporcionalmente maior nos testes realizados com a abordagem DVS-Gesture, pois embora seja medido uma baixa taxa de atividades de *spikes* na rede neural durante o processo, a menor entre as 3 aplicações, há também à consideração de um intervalo de tempo T mais amplo na análise em relação às outras bases. Como resultado, ocorreram mais emissões de *spikes*, o que, por conseguinte, não resultou em um ganho de energia tão significativo como inicialmente esperado. Entretanto, vale ressaltar que ainda houve um aumento notável de 12 vezes no ganho de energia em comparação com as redes neurais convencionais, graças ao aproveitamento eficiente da memória.

5 Conclusões

O trabalho apresentou um sistema de classificação de imagens utilizando componentes e redes neurais não convencionais para englobar todo o processo, sendo as câmeras de eventos e as SNNs os elementos principais de todo o processo, obtendo uma taxa de acerto na média de 90% em datastes puramente neuromórficos e 65% para um dataset convertido, porém com um grau de complexidade maior.

Foi demonstrado que é possível adaptar modelos convencionais simples e complexos, como o modelo Tiny-VGG11, a redes neurais do tipo SNNs, desde que seja utilizado o *layer* de *Batch-Normalization* que é essencial para o aprendizado de máquina deste tipo de rede. Além isso, apesar do número de parâmetros ser equivalente às redes neurais convencionais, foi demostrado como há um ganho de consumo energético e de memória durante o processo de classificação de imagens, utilizando 5 vezes menos energia e memória do que métodos convencionais. Isso se deve muito ao fato destas redes neurais, e imagens extraídas das câmeras de eventos, serem de natureza esparsa, o que significa que há poucos valores diferentes de zeros envolvidos durante todo o processo, representando em torno de 10% se considerarmos as 3 bases estudadas.

5.1 Trabalhos Futuros

Apesar das SNNs e das câmeras de evento proporcionarem um potencial ganho de utilização de memória, energia, e diminuição da latência envolvida para coletar e processar as imagens, é importante destacar que é essencial a utilização de um *Hardware* puramente neuromórfico, como o Chip *Loihi* da intel, para extrair todo o potencial que a combinação destas duas tecnologias oferece, pois este potencial acaba sendo desperdiçado em *Hardware* convencional. Assim, recomenda-se a utilização de Simuladores para aprofundar o estudo sobre a aplicação destas tecnologias, porém até a realização deste projeto, tanto a acessibilidade dos Chips neuromórficos, assim como estes simuladores ainda se encontram nas suas etapas de desenvolvimento inicial.

Além disso, uma área de pesquisa promissora para explorar em trabalhos futuros está relacionada ao impacto das lacunas de resolução espacial entre os pixels DVS (*Dynamic Vision Sensor*) e APS (*Active Pixel Sensor*) nas redes neurais e abordagens de processamento de imagem. Embora a integração de pixels DVS e APS em uma matriz de pixels híbridos seja uma tendência crescente na indústria, a disparidade na resolução espacial entre esses pixels representa um desafio significativo. Isso se deve ao fato de que a resolução espacial dos pixels DVS é consideravelmente menor do que a dos pixels APS, resultando em informações sensoriais discrepantes. Portanto, analisar como as redes

neurais podem lidar com essa diferença de resolução e como isso afeta o desempenho de tarefas de visão computacional é uma área crucial a ser investigada.

Adicionalmente, é de suma importância considerar o uso de câmeras baseadas em eventos em experimentos, especialmente sem a necessidade de depender exclusivamente de conjuntos de dados pré-processados. Essas câmeras têm se destacado pela sua capacidade de capturar eventos em tempo real, fornecendo uma abordagem mais dinâmica e eficiente para a visão computacional. No entanto, é fundamental também investigar os ruídos gerados por essas câmeras de eventos, pois os ruídos gerados pelas câmeras de evento podem ser distintos e têm o potencial de influenciar negativamente a qualidade das informações capturadas. Portanto, é essencial entender como esses ruídos se comportam e como eles podem ser tratados para melhorar o desempenho das redes neurais em tarefas de visão computacional. Essa análise crítica dos ruídos, juntamente com a consideração das disparidades de resolução espacial entre os pixels DVS e APS, contribuirá para o desenvolvimento de abordagens mais robustas e eficazes na integração de câmeras baseadas em eventos em aplicações práticas.

Destaca-se também a quantidade reduzida disponível de Datasets puramente neuromórficos orientados à classificação de imagens, pois existe um número limitado de bancos de dados o que desacelera as pesquisas e analises implementadas na área.

Referências Bibliográficas

AMIR, A.; TABA, B.; BERG, D.; MELANO, T.; MCKINSTRY, J.; NOLFO, C. D.; NAYAK, T.; ANDREOPOULOS, A.; GARREAU, G.; MENDOZA, M.; KUSNITZ, J.; DEBOLE, M.; ESSER, S.; DELBRUCK, T.; FLICKNER, M.; MODHA, D. A low power, fully event-based gesture recognition system. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2017. p. 7388–7397.

CORDONE, L. *Performance of spiking neural networks on event data for embedded automotive applications*. Tese (Theses) — Université Côte d'Azur, dez. 2022. Disponível em: <https://theses.hal.science/tel-04026653>.

CORDONE, L.; MIRAMOND, B.; FERRANTE, S. *Learning from Event Cameras with Sparse Spiking Convolutional Neural Networks*. 2021.

CORDONE, L.; MIRAMOND, B.; THIERION, P. *Object Detection with Spiking Neural Networks on Automotive Event Data*. 2022.

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

DENG, L.; WU, Y.; HU, X.; LIANG, L.; DING, Y.; LI, G.; ZHAO, G.; LI, P.; XIE, Y. Rethinking the performance comparison between snns and anns. *Neural Networks*, v. 121, p. 294–307, 2020. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608019302667>.

FANG, W.; CHEN, Y.; DING, J.; CHEN, D.; YU, Z.; ZHOU, H.; MASQUELIER, T.; TIAN, Y.; CONTRIBUTORS other. *SpikingJelly*. 2020. <https://github.com/fangwei123456/spikingjelly>. Accessed: 2023.

FANG, W.; YU, Z.; CHEN, Y.; MASQUELIER, T.; HUANG, T.; TIAN, Y. *Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks*. 2021.

GALLEGO, G.; DELBRUCK, T.; ORCHARD, G.; BARTOLOZZI, C.; TABA, B.; CENSI, A.; LEUTENEGGER, S.; DAVISON, A. J.; CONRADT, J.; DANIILIDIS, K.; SCARAMUZZA, D. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Institute of Electrical and Electronics Engineers (IEEE), v. 44, n. 1, p. 154–180, jan 2022. Disponível em: <https://doi.org/10.1109%2Ftpami.2020.3008413>.

GEHRIG, D.; LOQUERCIO, A.; DERPANIS, K. G.; SCARAMUZZA, D. *End-to-End Learning of Representations for Asynchronous Event-Based Data*. 2019.

HE, W.; WU, Y.; DENG, L.; LI, G.; WANG, H.; TIAN, Y.; DING, W.; WANG, W.; XIE, Y. *Comparing SNNs and RNNs on Neuromorphic Vision Datasets: Similarities and Differences*. 2020.

JOHANSSON, O. *Training of Object Detection Spiking Neural Networks for Event-Based Vision*. 2021. 37 p.

KAISER, J.; MOSTAFA, H.; NEFTCI, E. Synaptic plasticity dynamics for deep continuous local learning (DECOLLE). *Frontiers in Neuroscience*, Frontiers Media SA, v. 14, may 2020. Disponível em: <https://doi.org/10.3389%2Ffnins.2020.00424>.

LI, H.; LIU, H.; JI, X.; LI, G.; SHI, L. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, v. 11, 2017. ISSN 1662-453X. Disponível em: <https://www.frontiersin.org/articles/10.3389/fnins.2017.00309>.

LI, L.; JAMIESON, K.; DESALVO, G.; ROSTAMIZADEH, A.; TALWALKAR, A. *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*. 2018.

LICHTSTEINER, P.; POSCH, C.; DELBRUCK, T. A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, v. 43, n. 2, p. 566–576, 2008.

LIU, Q.; RUAN, H.; XING, D.; TANG, H.; PAN, G. *Effective AER Object Classification Using Segmented Probability-Maximization Learning in Spiking Neural Networks*. 2020.

LOSHCHILOV, I.; HUTTER, F. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017.

NEFTCI, E. O.; MOSTAFA, H.; ZENKE, F. *Surrogate Gradient Learning in Spiking Neural Networks*. 2019.

O'SHEA, K.; NASH, R. *An Introduction to Convolutional Neural Networks*. 2015.

PELLEGRINI, T.; ZIMMER, R.; MASQUELIER, T. *Low-activity supervised convolutional spiking neural networks applied to speech commands recognition*. 2020.

SAMADZADEH, A.; FAR, F. S. T.; JAVADI, A.; NICKABADI, A.; CHEHREGHANI, M. H. *Convolutional Spiking Neural Networks for Spatio-Temporal Feature Extraction*. 2021.

SHRESTHA, S. B.; ORCHARD, G. *SLAYER: Spike Layer Error Reassignment in Time*. 2018.

SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015.

SIRONI, A.; BRAMBILLA, M.; BOURDIS, N.; LAGORCE, X.; BENOSMAN, R. *HATS: Histograms of Averaged Time Surfaces for Robust Event-based Object Classification*. 2018.

WU, Y.; DENG, L.; LI, G.; ZHU, J.; SHI, L. *Direct Training for Spiking Neural Networks: Faster, Larger, Better*. 2018.

WU, Y.; DENG, L.; LI, G.; ZHU, J.; SHI, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, Frontiers Media SA, v. 12, may 2018. Disponível em: <https://doi.org/10.3389%2Ffnins.2018.00331>.

XIANGHAN, Z.; QUNLI, Z. An analysis of students' failing in university based on least square method and a new arctan exp logistic regression function. *Mathematical Problems in Engineering*, v. 2022, p. 1–9, 02 2022.

XING, Y.; CATERINA, G. D.; SORAGHAN, J. A new spiking convolutional recurrent neural network (scrnn) with applications to event-based hand gesture recognition. *Frontiers in Neuroscience*, v. 14, 2020. ISSN 1662-453X. Disponível em: <https://www.frontiersin.org/articles/10.3389/fnins.2020.590164>.

YI, T.-H.; LI, H.-N.; ZHAO, X.-Y. Noise smoothing for structural vibration test signals using an improved wavelet thresholding technique. *Sensors (Basel, Switzerland)*, v. 12, p. 11205–20, 12 2012.

ZAKHAROV, N. S.; SAPOZHENKOV, N. O.; TYAN, R. V.; NAZAROV, V. P. Applying the heaviside step function to simulate the changes of temperature in automotive batteries. *Journal of Physics: Conference Series*, IOP Publishing, v. 2061, n. 1, p. 012001, oct 2021. Disponível em: <https://dx.doi.org/10.1088/1742-6596/2061/1/012001>.

ZIMMER, R.; PELLEGRINI, T.; SINGH, S. F.; MASQUELIER, T. *Technical report: supervised training of convolutional spiking neural networks with PyTorch*. 2019.