Um Blueprint Neuromórfico de HPC: Uma Estrutura Algorítmica para Computação Inspirada no Cérebro

Seção 1: Princípios Fundamentais da Computação Neural de Alto Desempenho

A busca por uma computação que transcenda as limitações da arquitetura de von Neumann nos leva inevitavelmente ao mais sofisticado e eficiente processador de informações conhecido: o cérebro humano. Este documento apresenta um blueprint técnico para um sistema de computação de alto desempenho (HPC) que não apenas se inspira, mas se baseia rigorosamente nos princípios fundamentais da computação neural. Antes de mergulhar nos algoritmos e estruturas de dados, é imperativo estabelecer os axiomas neurocientíficos que governam nosso design. Estes princípios não são meras características desejáveis; são as restrições e os objetivos que definem a própria natureza de uma arquitetura verdadeiramente neuromórfica, distinguindo-a fundamentalmente dos sistemas computacionais convencionais.

1.1 Paralelismo Massivo e Processamento Assíncrono Orientado a Eventos

O princípio mais elementar da computação cerebral é sua escala e modo de operação. O cérebro humano contém aproximadamente 86 a 100 bilhões de neurônios, cada um funcionando como uma unidade de processamento individual.[1] Estes neurônios operam em paralelo, permitindo o processamento simultâneo de vastas quantidades de informação. No entanto, este paralelismo é radicalmente diferente daquele encontrado em supercomputadores tradicionais.

Os circuitos digitais convencionais são síncronos, governados por um relógio global que dita o ritmo de todas as operações. A cada "tique" do relógio, cada componente do sistema executa uma instrução ou permanece ocioso, consumindo energia

independentemente de estar realizando um trabalho útil. Em contraste, a computação neural é fundamentalmente assíncrona e orientada a eventos.[4] Não existe um relógio central. A computação e a comunicação ocorrem apenas quando um evento significativo acontece: a emissão de um potencial de ação, ou "spike". Um spike é um pulso elétrico discreto, um evento "tudo ou nada", que um neurônio dispara quando seu potencial de membrana interno ultrapassa um limiar.[7]

Esta abordagem orientada a eventos é a base da eficiência do cérebro. A energia só é consumida para computar e transmitir informações quando há novas informações a serem processadas. O silêncio é a norma, e a atividade é a exceção. Este paradigma impõe uma restrição de design crucial para nosso blueprint: o sistema deve ser construído sobre uma base de passagem de mensagens e manipuladores de eventos, não sobre um ciclo de relógio global. A lógica computacional não é executada em passos de tempo fixos, mas é acionada pela chegada de spikes de outros neurônios.[8] A comunicação é esparsa e irregular, refletindo o fluxo de informações relevantes, em vez de um fluxo constante e forçado de dados. Esta arquitetura não só economiza energia, mas também codifica informações no tempo preciso dos eventos, uma característica que exploraremos mais adiante.

1.2 Eficiência Energética Extrema através de Esparsidade e Computação na Memória

A disparidade de eficiência energética entre o cérebro e os supercomputadores é astronômica. O cérebro humano realiza um volume de operações estimado em um exaflop (1018 operações por segundo) com um orçamento de energia de apenas 20 watts, o equivalente a uma lâmpada fraca. Um supercomputador convencional que executa a mesma carga de trabalho requer dezenas de megawatts, uma quantidade de energia suficiente para abastecer uma pequena cidade.[11] Esta diferença de mais de seis ordens de magnitude não é um acaso da biologia, mas o resultado de dois princípios de design interligados que são centrais para o nosso blueprint.[14]

O primeiro princípio é a **esparsidade**. A vasta maioria dos neurônios do cérebro está silenciosa na maior parte do tempo. As taxas médias de disparo no córtex são notavelmente baixas, estimadas em cerca de 0.16 Hz, ou aproximadamente um spike a cada seis segundos.[15] A representação da informação é esparsa, o que significa que qualquer estímulo ou conceito é codificado pela atividade de um pequeno subconjunto de neurônios de uma população muito maior.[17] Esta esparsidade tem

duas consequências vitais: primeiro, minimiza drasticamente o consumo de energia, pois apenas uma pequena fração do sistema está ativa a qualquer momento; segundo, melhora a relação sinal-ruído, tornando as representações mais robustas e distintas.

O segundo princípio é a **computação na memória** (in-memory computing). Na arquitetura de von Neumann, que domina a computação moderna, a unidade de processamento (CPU) e a unidade de memória (RAM) são entidades físicas separadas. Uma quantidade significativa de tempo e energia é gasta no transporte de dados entre essas duas unidades através de um barramento, um fenômeno conhecido como o "gargalo de von Neumann". O cérebro anula este gargalo ao co-localizar fisicamente a memória e o processamento.[6] As sinapses, as conexões entre os neurônios, atuam como a memória do sistema, armazenando a "força" ou o peso da conexão. O corpo celular do neurônio (soma) atua como a unidade de processamento, integrando os sinais recebidos através dessas sinapses. A memória está, portanto, distribuída por toda a rede e é intrinsecamente ligada à computação. Nosso blueprint deve adotar esta arquitetura, onde cada nó de processamento possui sua própria memória local, eliminando a necessidade de um pool de memória centralizado e o dispendioso transporte de dados.

1.3 Computação Híbrida Analógica-Digital e Imprecisa

Os sistemas computacionais modernos são construídos sobre a precisão da lógica digital. Cada operação é exata, e os erros são intoleráveis. O cérebro, no entanto, opera com base em um modelo computacional fundamentalmente diferente, que é um híbrido de processos analógicos e digitais e que abraça a imprecisão.

A computação dentro de um único neurônio é em grande parte analógica. O potencial de membrana de um neurônio flutua continuamente à medida que integra as correntes sinápticas de entrada, que são elas mesmas graduadas em força.[7] Este processo de integração é uma soma contínua e ponderada no tempo. No entanto, a saída desta computação analógica é um evento digital: o spike. Quando o potencial de membrana atinge o limiar, um spike "tudo ou nada" é gerado. O sistema, portanto, combina a riqueza da computação analógica com a robustez da sinalização digital.

Além disso, os componentes neurais são inerentemente "desleixados" e probabilísticos quando comparados com os transistores de alta precisão.[8] As

sinapses podem falhar em liberar neurotransmissores, e a quantidade liberada pode variar. O cérebro não depende da precisão de uma única computação, mas da robustez estatística que emerge da computação coletiva de bilhões de unidades imprecisas. Esta tolerância a falhas e a capacidade de extrair um sinal confiável do ruído são características essenciais.

Para o nosso blueprint, isso implica que não devemos superinvestir em aritmética de alta precisão de ponto flutuante. A força do sistema não virá da precisão de unidades individuais, mas da computação paralela massiva de unidades simples e eficientes. Hardware neuromórfico como o Loihi 2 da Intel já explora este princípio ao suportar aritmética de baixa precisão.[21] As regras de atualização em nosso sistema serão, portanto, projetadas para serem computacionalmente baratas, usando operações inteiras ou de ponto fixo de baixa precisão, refletindo a natureza da computação biológica.

1.4 O Cérebro como um Sistema Adaptativo e Auto-Organizado

Talvez a distinção mais profunda entre a computação cerebral e a convencional seja que o cérebro não é programado, mas treinado. Sua estrutura e função não são fixas, mas se auto-organizam continuamente em resposta à experiência sensorial. Este processo, conhecido como plasticidade neural, ocorre em múltiplas escalas de tempo e é o mecanismo fundamental por trás da aprendizagem e da memória.[22]

As conexões sinápticas se fortalecem ou enfraquecem com base nos padrões de atividade (plasticidade sináptica). A excitabilidade intrínseca de um neurônio pode mudar para manter a estabilidade da rede (plasticidade homeostática). Novas conexões podem ser formadas e as existentes podem ser eliminadas (plasticidade estrutural). Este conjunto de mecanismos de plasticidade permite que o cérebro adapte sua própria "fiação" e "código" para modelar a estrutura do mundo e resolver problemas de forma eficiente.

A consequência para o nosso blueprint é monumental: não podemos projetar um algoritmo estático. Devemos projetar um **sistema de aprendizagem** que possa modificar seus próprios parâmetros e, em última análise, sua própria estrutura para se adaptar a novos dados e tarefas. O blueprint deve, portanto, especificar não apenas a lógica de processamento, mas também a hierarquia de regras de

aprendizagem que governam a evolução do sistema ao longo do tempo.

A interconexão desses princípios revela uma verdade mais profunda. O paralelismo massivo só é energeticamente viável por causa da esparsidade. A computação na memória é a arquitetura física que torna o paralelismo esparso eficiente. A plasticidade adaptativa é o que esculpe a rede para gerar códigos esparsos em primeiro lugar. E a imprecisão dos componentes individuais é tolerável devido à robustez estatística do coletivo e à natureza autocorretiva das regras de aprendizagem. Portanto, não estamos projetando um único algoritmo, mas um sistema de laços de feedback aninhados e interativos que operam em diferentes escalas de tempo: laços rápidos para a computação baseada em spikes, laços de médio prazo para a aprendizagem sináptica e laços lentos para a estabilidade e estrutura da rede. O restante deste documento detalhará a implementação algorítmica deste sistema complexo e dinâmico.

Seção 2: A Malha Computacional: Topologia de Rede de Mundo Pequeno e Livre de Escala

A base sobre a qual todos os processos dinâmicos de nossa arquitetura neuromórfica irão operar é sua "fiação" estática, a topologia da rede. Esta estrutura não é aleatória nem uniforme; é uma arquitetura altamente otimizada, moldada por pressões evolutivas para facilitar o fluxo de informações de maneira eficiente. A compreensão e a replicação desta topologia são o primeiro passo para construir um sistema computacional que espelhe a capacidade do cérebro.

2.1 Definindo o Grafo: Neurônios como Nós, Sinapses como Arestas

Na sua forma mais abstrata, o cérebro é uma rede complexa, ou um grafo, no sentido matemático.[8] Neste grafo, os neurônios são os nós (vértices) e as sinapses são as arestas direcionais e ponderadas que os conectam. Com uma escala de aproximadamente

10^{11} nós e 10^{14} a 10^{15} arestas no cérebro humano, a magnitude desta rede é imensa.[1]

Esta perspectiva baseada em grafos é fundamental, pois nos permite aplicar as ferramentas rigorosas da teoria dos grafos para analisar e projetar a arquitetura do nosso sistema. O nosso blueprint começa, portanto, com a definição de um grafo direcionado $G=(V,E)$, onde V é o conjunto de unidades de processamento (neurônios) e E é o conjunto de conexões (sinapses). Cada sinapse $e \in E$ terá propriedades associadas, como um peso (força sináptica) e um atraso de propagação, que serão cruciais para a dinâmica da rede.

2.2 Propriedades de Mundo Pequeno: A Arquitetura do "Melhor de Dois Mundos"

Estudos empíricos de redes cerebrais, tanto estruturais (conexões anatômicas) quanto funcionais (correlações de atividade), revelaram consistentemente uma propriedade topológica notável: elas são redes de "mundo pequeno" (small-world).[26] Uma rede de mundo pequeno é um intermediário entre uma rede regular (como uma treliça) e uma rede aleatória, possuindo o melhor de ambos os mundos.

- **Alto Coeficiente de Agrupamento (High Clustering):** Assim como em uma rede regular, os neurônios em uma rede de mundo pequeno tendem a formar cliques ou grupos densamente interconectados. Se o neurônio A está conectado a B e C, há uma alta probabilidade de que B e C também estejam conectados entre si. Isso reflete a organização do cérebro em módulos funcionais especializados (por exemplo, colunas corticais na visão), onde o processamento local é intenso e eficiente.
- **Curto Comprimento Médio do Caminho (Short Average Path Length):** Assim como em uma rede aleatória, quaisquer dois neurônios na rede, mesmo que em módulos muito distantes, podem ser alcançados através de um número surpreendentemente pequeno de conexões intermediárias.

Esta topologia é extraordinariamente eficiente para o processamento de informações. O alto agrupamento permite o processamento **segregado** e especializado de informações dentro de módulos locais, enquanto o curto comprimento do caminho permite a **integração** rápida de informações entre esses módulos. É uma arquitetura que otimiza simultaneamente a especialização e a comunicação global, um pré-requisito para funções cognitivas complexas que exigem a ligação de informações de diferentes modalidades sensoriais e áreas cerebrais.

2.3 Características Livres de Escala e Hubs de Rede

Além das propriedades de mundo pequeno, as redes cerebrais frequentemente exibem características de redes "livres de escala" (scale-free).[26] A distribuição de conectividade (o número de conexões por nó) em tais redes segue uma lei de potência, muitas vezes com um truncamento exponencial.[28] Em termos práticos, isso significa que, embora a maioria dos neurônios tenha um número relativamente pequeno de conexões, existe um pequeno número de neurônios "hub" que são excepcionalmente bem conectados.

Esses hubs atuam como pontos centrais de trânsito na rede, desempenhando um papel crítico na integração global de informações e na manutenção do curto comprimento do caminho da rede de mundo pequeno. Eles são as pontes que conectam os diversos módulos especializados. A existência de hubs implica que nem todos os neurônios são funcionalmente equivalentes; alguns têm uma importância desproporcional na arquitetura da rede. Isso tem profundas implicações para o roteamento de informações, a resiliência da rede a danos (a remoção de um hub é muito mais prejudicial do que a de um nó comum) e a eficiência da comunicação.

2.4 Blueprint: Algoritmo de Geração de Rede Inicial

Para construir a malha computacional do nosso sistema, precisamos de um algoritmo que possa gerar um grafo com essas propriedades topológicas desejadas. É importante notar que este não é um modelo de desenvolvimento biológico (que será abordado mais tarde com a plasticidade), mas sim um método de engenharia para criar uma rede inicial com a estrutura correta. O algoritmo de Watts-Strogatz é um método canônico para este fim.

O pseudocódigo a seguir descreve o processo:

Code snippet

```
FUNÇÃO GerarRedeMundoPequeno(num_nos, vizinhos_k, prob_religacao_p):
```

```
  // Passo 1: Inicializar uma rede de treliça em anel regular
  grafo = CriarTreliçaAnel(num_nos, vizinhos_k)

  // Passo 2: Iterar sobre cada aresta e religar com probabilidade 'p'
  PARA CADA no_i DE 0 ATÉ num_nos-1:
    PARA CADA no_j NOS vizinhos_k DE no_i:
      // Considerar cada aresta apenas uma vez
      SE no_i < no_j ENTÃO
        SE Random() < prob_religacao_p ENTÃO
          // Encontrar um novo nó para conectar que não seja o próprio nó
          // e que não crie uma aresta duplicada
          no_k = no_i
          ENQUANTO no_k == no_i OU grafo.TemAresta(no_i, no_k):
            no_k = EscolherNoAleatorio(num_nos)

          // Religar a aresta
          RemoverAresta(grafo, no_i, no_j)
          AdicionarAresta(grafo, no_i, no_k)
        FIM SE
      FIM SE
    FIM PARA
  FIM PARA

  RETORNAR grafo
FIM FUNÇÃO
```

Este procedimento cria um grafo que pode ser ajustado:

- Se p=0, o resultado é uma treliça regular com alto agrupamento e longo comprimento de caminho.
- Se p=1, o resultado é um grafo aleatório com baixo agrupamento e curto comprimento de caminho.
- Para valores intermediários de p (tipicamente pequenos), o grafo exibe as características de mundo pequeno desejadas: alto agrupamento e curto comprimento de caminho.

O processo pode ser visualizado da seguinte forma:

Code snippet

```
graph TD
    A -- Alto Agrupamento, Alto Comprimento de Caminho --> B(Religar Arestas com Probabilidade 'p');
    B -- 'p' baixo --> C{Rede de Mundo Pequeno};
    C -- Alto Agrupamento, Baixo Comprimento de Caminho --> D;
```

A topologia da rede não é um substrato passivo; é uma forma de pré-computação. A estrutura de mundo pequeno e livre de escala é uma solução evoluída para o problema de processar um mundo complexo com recursos limitados. Ela incorpora um conjunto de suposições sobre a estrutura estatística dos estímulos naturais e fornece uma configuração de hardware otimizada para processá-los. O curto comprimento do caminho, por exemplo, é um pré-requisito físico para funções cognitivas rápidas que precisam ligar informações de domínios distantes. Portanto, a geração da topologia da rede é uma etapa de design crítica. A escolha dos parâmetros, como o número de vizinhos k e a probabilidade de religação p, não é arbitrária; é uma decisão fundamental que moldará profundamente as capacidades computacionais de todo o sistema.

Seção 3: A Unidade de Processamento Central: O Modelo de Neurônio com Spikes de Izhikevich

Após definir a arquitetura global da nossa rede, a próxima etapa é detalhar o comportamento de seus componentes individuais: os nós de processamento ou neurônios. A escolha do modelo de neurônio é um compromisso entre a plausibilidade biológica e a eficiência computacional. Para este blueprint, adotamos o modelo de Izhikevich, uma escolha que oferece um equilíbrio notável entre esses dois fatores. Ele é capaz de reproduzir uma vasta gama de comportamentos neuronais observados biologicamente, mantendo-se computacionalmente leve o suficiente para simulações em larga escala.

3.1 As Equações e Parâmetros do Modelo

O modelo de Izhikevich é elegantemente simples, descrito por um sistema de duas equações diferenciais ordinárias acopladas, complementadas por uma regra de reinicialização discreta após um spike.[30]

As equações que governam a dinâmica do neurônio são:

1. Potencial de Membrana (v):
   $\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I$
2. Variável de Recuperação da Membrana (u):
   $\frac{du}{dt} = a(bv - u)$

Estas equações são acompanhadas por uma condição de reinicialização após o spike:

- Condição de Spike e Reinicialização:
  Se $v \geq 30$ mV, então o neurônio dispara um spike e suas variáveis são reinicializadas para:
  $v \leftarrow c$
  $u \leftarrow u + d$

Vamos detalhar cada componente:

- **v (Potencial de Membrana):** É a variável de estado primária do neurônio, análoga à voltagem através da membrana celular. Sua dinâmica é não linear (devido ao termo v^2), o que permite a geração de spikes.
- **u (Variável de Recuperação):** Representa de forma abstrata os efeitos combinados da inativação dos canais de sódio (Na^+) e da ativação dos canais de potássio (K^+). Ela fornece um feedback negativo para v, contribuindo para a repolarização da membrana após um spike e para fenômenos de adaptação.
- **I (Corrente de Entrada):** Representa a soma de todas as correntes sinápticas que chegam de outros neurônios. É o principal motor da atividade do neurônio.
- **a,b,c,d (Parâmetros de Controle):** Estes quatro parâmetros adimensionais são a chave para a versatilidade do modelo. Ao ajustar seus valores, é possível replicar uma ampla variedade de comportamentos de disparo de neurônios corticais reais sem alterar as equações fundamentais.[32]

3.2 Um Rico Repertório de Padrões de Disparo

A principal força do modelo de Izhikevich reside em sua capacidade de, com um único

conjunto de equações, gerar uma diversidade de padrões de disparo biologicamente realistas simplesmente ajustando os quatro parâmetros de controle.[32] Isso permite que nosso blueprint suporte uma rede heterogênea, composta por diferentes tipos de neurônios (por exemplo, neurônios piramidais excitatórios e interneurônios inibitórios), o que é essencial para a criação de dinâmicas de rede complexas.

A tabela a seguir, adaptada de Izhikevich (2004), fornece um "livro de receitas" prático para instanciar diferentes populações de neurônios na rede, traduzindo a teoria em configurações concretas e utilizáveis.

Tipo de Neurônio / Padrão de Disparo	Parâmetro 'a'	Parâmetro 'b'	Parâmetro 'c' (mV)	Parâmetro 'd'	Comportamento Característico
Regular Spiking (RS)	0.02	0.2	-65	8	Padrão de disparo tônico com adaptação de frequência. Típico de neurônios piramidais excitatórios.
Intrinsically Bursting (IB)	0.02	0.2	-55	4	Dispara rajadas de spikes (bursts) no início de um estímulo constante.
Chattering (CH)	0.02	0.2	-50	2	Dispara bursts de alta frequência de forma rítmica.
Fast Spiking (FS)	0.1	0.2	-65	2	Dispara spikes de alta frequência

					com pouca ou nenhuma adaptação. Típico de interneurônios inibitórios.
Low-Threshold Spiking (LTS)	0.02	0.25	-65	2	Exibe disparos de baixa frequência com adaptação significativa.
Resonator (RZ)	0.1	0.26	-65	2	Não dispara para um estímulo constante, mas ressoa e dispara em resposta a entradas em uma frequência preferida.
Thalamo-Cortical (TC)	0.02	0.25	-65	0.05	Exibe comportamento de disparo pós-inibitório (rebound bursting).

3.3 O Período Refratário e o Atraso Sináptico

Dois mecanismos de controle temporal são essenciais para a dinâmica da rede e são implicitamente ou explicitamente modelados em nosso sistema.

- **Período Refratário:** Após disparar um spike, um neurônio entra em um **período refratário absoluto**, durante o qual é impossível disparar outro spike,

independentemente da intensidade da entrada. Isso ocorre biologicamente devido à inativação dos canais de Na+.[35] Segue-se um **período refratário relativo**, onde um estímulo mais forte que o normal é necessário para provocar um spike. No modelo de Izhikevich, este comportamento é capturado pela reinicialização abrupta de v para um valor baixo (c) e pelo aumento da variável de recuperação u (pelo valor d), que efetivamente hiperpolariza o neurônio e aumenta o limiar para o próximo spike. Este mecanismo é crucial, pois limita a taxa máxima de disparo de um neurônio e garante a propagação unidirecional dos sinais ao longo de uma cadeia neural.

- **Atraso Sináptico:** Existe um atraso mensurável, tipicamente entre 0.5 e 4.0 milissegundos, entre a chegada de um spike no terminal pré-sináptico e o início da resposta elétrica no neurônio pós-sináptico.[37] Este atraso é devido ao tempo necessário para a liberação do neurotransmissor, sua difusão através da fenda sináptica e sua ligação aos receptores pós-sinápticos.[39] Este não é um detalhe menor; os atrasos sinápticos são fundamentais para a criação das dinâmicas temporais precisas necessárias para regras de aprendizagem como a Plasticidade Dependente do Tempo do Spike (STDP) e para a geração de oscilações de rede complexas. Em nosso blueprint, o atraso será modelado como uma propriedade da conexão sináptica (a aresta no grafo), não do neurônio.

3.4 Pseudocódigo: Passo de Atualização de um Único Neurônio

A implementação da dinâmica do neurônio pode ser realizada usando um método numérico simples, como o método de Euler, que é suficiente dada a natureza computacionalmente eficiente do modelo.

Code snippet

```
PROCEDIMENTO AtualizarEstadoNeuronio(neuronio, corrente_entrada, dt):
  // Usar o método de Euler para integração numérica
  v_antigo = neuronio.v

  // Atualizar o potencial de membrana 'v'
  dv = (0.04 * v_antigo^2 + 5 * v_antigo + 140 - neuronio.u + corrente_entrada)
```

```
    neuronio.v = v_antigo + dt * dv

    // Atualizar a variável de recuperação 'u'
    du = neuronio.parametros.a * (neuronio.parametros.b * v_antigo - neuronio.u)
    neuronio.u = neuronio.u + dt * du

    // Verificar a condição de spike
    SE neuronio.v >= 30 ENTÃO
        // Reinicializar as variáveis após o spike
        neuronio.v = neuronio.parametros.c
        neuronio.u = neuronio.u + neuronio.parametros.d
        RETORNAR VERDADEIRO // Spike ocorreu
    SENÃO
        RETORNAR FALSO
    FIM SE
FIM PROCEDIMENTO
```

A escolha do modelo de Izhikevich reflete uma compreensão mais profunda da computação neural. Ao contrário dos primeiros modelos de IA que tratavam o neurônio como uma porta lógica com limiar, o modelo de Izhikevich o define como um sistema dinâmico não linear. Sistemas dinâmicos podem exibir uma rica gama de comportamentos complexos, como ressonância, oscilações sub-limiar, adaptação e bi-estabilidade.[33] Essas propriedades não são meras curiosidades, mas primitivas computacionais em si. Um neurônio ressonador pode atuar como um filtro de frequência, respondendo preferencialmente a entradas que chegam a uma taxa específica. Um neurônio adaptativo pode sinalizar novidade, disparando fortemente para um novo estímulo, mas silenciando se ele persistir. Ao parametrizar os neurônios de forma diferente, estamos, de fato, incorporando diferentes capacidades computacionais diretamente nos nós da nossa rede.

Seção 4: A Linguagem da Rede: Codificação Temporal Esparsa

Para que a nossa rede neuromórfica processe informações do mundo real, primeiro precisamos traduzir dados contínuos ou analógicos para a linguagem que a rede entende: a linguagem dos spikes. Este processo de codificação não é uma mera formalidade técnica; é uma decisão de design fundamental que determina a

velocidade, a eficiência e a própria natureza da computação que o sistema pode realizar. A estratégia de codificação do cérebro é guiada pelo princípio da esparsidade, que otimiza tanto a eficiência metabólica quanto a capacidade de extração de características.

4.1 O Princípio da Codificação Esparsa: Eficiência e Extração de Características

A hipótese da codificação esparsa postula que a informação sensorial é representada pela forte ativação de um pequeno número de neurônios de uma população muito maior.[17] Esta estratégia é vantajosa por várias razões. Primeiro, é metabolicamente eficiente, pois minimiza o número de spikes — eventos que consomem energia — necessários para representar um estímulo.[17] Segundo, melhora a relação sinal-ruído, tornando as representações mais robustas e fáceis de discriminar.

Matematicamente, a codificação esparsa pode ser formulada como um problema de otimização. Dado um sinal de entrada x (por exemplo, um trecho de uma imagem), o objetivo é encontrar uma representação que o aproxime como uma combinação linear de um conjunto de vetores de base, ou "átomos de dicionário", Φ. A aproximação tem a forma $x \approx \Phi a$, onde o vetor de coeficientes a representa a atividade neural e é forçado a ser esparso, ou seja, ter muito poucas entradas diferentes de zero.[18]

Esta é uma teoria poderosa que unifica vários princípios cerebrais. Ela fornece uma razão funcional para a observação de que a atividade neural é esparsa. Mais importante, ela define um objetivo claro para a aprendizagem: a rede deve aprender, através da experiência, um dicionário Φ de características (como as bordas e contornos aprendidos pelos neurônios no córtex visual primário, V1) que possa representar eficientemente os estímulos naturais.[42]

4.2 Modalidades de Codificação: Traduzindo Dados em Spikes

Existem várias estratégias principais para converter um valor analógico em um trem de spikes, cada uma com diferentes compromissos entre velocidade, robustez e complexidade.[44]

- **Codificação por Taxa (Rate Coding):** Esta é a abordagem mais clássica, onde a

intensidade de um estímulo é codificada na frequência de disparo (spikes por segundo) de um neurônio. Um valor de entrada mais alto corresponde a uma taxa de disparo mais alta. Uma maneira comum de implementar isso é usar um processo de Poisson, onde a probabilidade de um neurônio disparar em um pequeno intervalo de tempo é proporcional ao valor da entrada.[45] Embora robusta ao ruído (pois se baseia em uma média ao longo do tempo), a codificação por taxa é inerentemente lenta, pois requer uma janela de tempo para estimar a taxa com precisão.

- **Codificação por Latência (Latency Coding / Time-to-First-Spike):** Esta é uma forma de codificação temporal muito mais rápida. Aqui, a intensidade do estímulo é codificada inversamente no tempo de disparo do primeiro spike. Estímulos mais fortes provocam spikes mais cedo, enquanto estímulos mais fracos resultam em spikes mais tardios ou nenhum spike.[45] A informação está contida no momento preciso de um único evento, tornando este código extremamente eficiente em termos de tempo e energia.
- **Codificação por Ordem de Classificação (Rank-Order Coding):** Esta é uma extensão da codificação por latência para o nível de uma população de neurônios. A informação não está contida nos tempos absolutos dos spikes, mas na *ordem* em que os neurônios de uma população disparam.[46] O neurônio que dispara primeiro sinaliza a característica mais saliente, o segundo a segunda mais saliente, e assim por diante. Isso cria um código robusto e rápido que transmite uma grande quantidade de informação em uma única onda de atividade.

A escolha do esquema de codificação tem implicações profundas. A codificação por taxa pode ser adequada para estados estáveis ou para a integração de evidências ao longo do tempo, enquanto a codificação por latência é ideal para o processamento rápido de estímulos transitórios. Nosso blueprint deve ser flexível para suportar diferentes esquemas, talvez até usando-os em diferentes partes do sistema.

4.3 Pseudocódigo: Algoritmos de Codificação de Entrada

A seguir, apresentamos pseudocódigos para as principais modalidades de codificação.

Code snippet

```
// Codificação por Taxa usando um processo de Poisson
FUNÇÃO CodificacaoPorTaxa(valor, taxa_max, duracao, dt):
  // 'valor' é normalizado entre 0 e 1
  taxa = valor * taxa_max
  prob_spike = taxa * dt
  trem_spikes = InicializarVazio(duracao / dt)

  PARA t DE 0 ATÉ duracao COM PASSO dt:
    SE Random() < prob_spike ENTÃO
      trem_spikes[t / dt] = 1
    SENÃO
      trem_spikes[t / dt] = 0
    FIM SE
  FIM PARA

  RETORNAR trem_spikes

// Codificação por Latência
FUNÇÃO CodificacaoPorLatencia(valor, latencia_max, limiar):
  // 'valor' é normalizado entre 0 e 1, valor mais alto = spike mais cedo
  SE valor > limiar ENTÃO
    // Mapeamento inverso: valor alto -> tempo baixo
    tempo_spike = latencia_max * (1.0 - valor)
    RETORNAR tempo_spike
  SENÃO
    RETORNAR infinito // Nenhum spike
  FIM SE
FIM FUNÇÃO
```

4.4 Analogia Computacional: Matching Pursuit

O problema de encontrar a representação mais esparsa de um sinal (codificação esparsa) é computacionalmente intratável (NP-difícil). No entanto, algoritmos gulosos (greedy) como o **Matching Pursuit (MP)** e sua variante, **Orthogonal Matching**

Pursuit (OMP), fornecem soluções aproximadas eficientes.[49]

O algoritmo MP funciona de forma iterativa:

1. Encontra o átomo do dicionário que tem a maior correlação (produto interno) com o sinal atual (ou o resíduo do sinal).
2. Subtrai a projeção do sinal sobre este átomo "mais compatível".
3. Repete o processo no sinal residual até que uma condição de parada seja atingida (por exemplo, o resíduo seja pequeno o suficiente ou um número desejado de átomos tenha sido selecionado).

Esta abordagem oferece uma analogia computacional poderosa para o que um circuito neural local pode estar realizando. O processo de competição através da inibição lateral e do mecanismo "winner-take-all" (que será detalhado na próxima seção) pode ser visto como uma implementação biológica, paralela e distribuída de uma busca gulosa como o Matching Pursuit. Cada neurônio representa um átomo do dicionário, e a competição seleciona o neurônio (átomo) que melhor "explica" o sinal de entrada. Isso nos ajuda a entender o objetivo computacional por trás da dinâmica neural observada.

É crucial entender que a codificação não é um passo de pré-processamento isolado, como ocorre em redes neurais artificiais tradicionais, onde os dados são convertidos em um vetor estático. Em SNNs, a codificação é um processo dinâmico que se desenrola no tempo. A dinâmica da codificação (por exemplo, a estocasticidade na codificação de Poisson ou as constantes de tempo na codificação de latência) interage diretamente com a dinâmica dos neurônios na primeira camada da rede. Além disso, o feedback da própria rede pode modular o processo de codificação, por exemplo, através de mecanismos de atenção que alteram o ganho dos neurônios sensoriais. Portanto, a codificação de entrada não é uma utilidade externa, mas a primeira etapa da computação temporal e dinâmica da rede. O nosso blueprint deve tratar a camada "codificadora" como uma parte integrante e dinâmica da própria rede.

Seção 5: Computação Local: Dinâmicas Competitivas e Seleção de Características

Dentro de qualquer módulo funcional do cérebro, a computação não ocorre de forma

isolada em cada neurônio. Em vez disso, os neurônios em uma população local interagem constantemente, e o motivo computacional mais fundamental que emerge dessas interações é a **competição**. Este mecanismo é essencial para refinar representações, tomar decisões e implementar eficientemente o princípio da codificação esparsa discutido anteriormente. A competição garante que apenas as informações mais relevantes sejam propagadas, suprimindo o ruído e a redundância.

5.1 Inibição Lateral: O Mecanismo para Realce de Contraste

A base neurobiológica da competição é a **inibição lateral**. Este é um motivo de circuito onipresente no sistema nervoso, onde um neurônio excitado, ao disparar, não apenas envia sinais excitatórios para a frente, mas também ativa interneurônios inibitórios locais que, por sua vez, suprimem a atividade dos neurônios vizinhos.[53] O efeito líquido é que a atividade em uma área tende a suprimir a atividade em suas imediações.

Este mecanismo tem um efeito poderoso de **realce de contraste**, tanto no domínio espacial quanto no temporal. O exemplo clássico é a ilusão visual das bandas de Mach, onde as bordas entre tons de cinza adjacentes parecem mais nítidas do que realmente são, porque a inibição lateral escurece artificialmente o lado escuro da borda e clareia o lado claro.[53] Do ponto de vista computacional, a inibição lateral é um princípio fundamental para a aprendizagem de características e a detecção de saliência.[54] Ela ajuda a rede a focar nas diferenças e mudanças, que são frequentemente as portadoras de mais informação.

Em nosso blueprint, a inibição lateral será implementada estruturalmente. Uma população de neurônios excitatórios (por exemplo, modelados como neurônios Regular Spiking) terá conexões não apenas para a próxima camada, mas também para um pool local de interneurônios inibitórios (modelados como neurônios Fast Spiking). Esses interneurônios inibitórios projetam-se de volta para a população excitatória de forma difusa, criando um laço de feedback negativo que implementa a competição.

5.2 Winner-Take-All (WTA): O Resultado Decisivo da Competição

Quando a inibição lateral é suficientemente forte, a competição "suave" que ela medeia pode se tornar um processo decisivo conhecido como **Winner-Take-All (WTA)**. Nesta dinâmica, apenas o neurônio (ou um pequeno conjunto de neurônios, no caso de k-WTA) que recebe a entrada mais forte e dispara primeiro consegue se manter ativo. Seu disparo aumenta a inibição geral na população, silenciando rapidamente todos os seus concorrentes antes que eles tenham a chance de atingir o limiar.[56]

O WTA é a realização algorítmica da ideia central da codificação esparsa. Se cada neurônio em uma população representa uma característica diferente (um átomo do dicionário), o WTA garante que apenas o neurônio cuja característica melhor "corresponde" à entrada atual dispare. Isso impõe a esparsidade na representação e, ao mesmo tempo, executa uma função de seleção ou classificação. O "vencedor" da competição sinaliza a presença da sua característica preferida na entrada. Este é um bloco de construção fundamental para a tomada de decisão em todos os níveis do sistema.

5.3 Diagrama Mermaid: Sequência de um Circuito WTA

A sequência de eventos em um circuito de inibição lateral que implementa o WTA pode ser visualizada com o seguinte diagrama:

Code snippet

```
sequenceDiagram
    participant Entrada
    participant População_Excitatória
    participant Interneurônio_Inibitório

    Input->>População_Excitatória: Trem de Spikes de Entrada Chega
    activate População_Excitatória
    População_Excitatória->>População_Excitatória: Neurônios começam a integrar a
```

```
entrada
    Note right of População_Excitatória: O potencial do Neurônio A sobe mais rápido

    População_Excitatória-->>Interneurônio_Inibitório: Neurônios ativos excitam o interneurônio
    activate Interneurônio_Inibitório

    Note right of População_Excitatória: Neurônio A atinge o limiar e dispara ANTES de ser fortemente inibido.
    População_Excitatória-->>Saída: Neurônio A ("Vencedor") envia spike

    Interneurônio_Inibitório-->>População_Excitatória: Interneurônio dispara, enviando sinal inibitório de volta
    deactivate Interneurônio_Inibitório

    Note left of População_Excitatória: Outros neurônios (B, C, etc.) são suprimidos pela inibição e seus potenciais são reinicializados.
    deactivate População_Excitatória
```

5.4 Pseudocódigo: Atualização de População com Inibição Lateral e WTA

A implementação desta dinâmica competitiva requer um algoritmo que opere no nível da população, não em neurônios individuais de forma isolada. O pseudocódigo a seguir descreve uma abordagem simplificada.

Code snippet

```
PROCEDIMENTO AtualizarPopulacaoLocal(populacao, entradas, dt):
    // Passo 1: Calcular a corrente excitatória para todos os neurônios
    correntes_excitatorias = {}
    PARA CADA neuronio EM populacao:
        correntes_excitatorias[neuronio] = CalcularCorrenteDeEntrada(neuronio, entradas)
```

```
    // Passo 2: Calcular a inibição total da população (feedback)
    // A inibição pode ser baseada na atividade de disparo recente da população
    atividade_total_recente = ObterAtividadeRecente(populacao)
    corrente_inibitoria = forca_inibicao * atividade_total_recente

    // Passo 3: Atualizar o estado de cada neurônio com a corrente total
    vencedores =
    PARA CADA neuronio EM populacao:
        corrente_total = correntes_excitatorias[neuronio] - corrente_inibitoria

        // Atualiza o estado interno do neurônio (v, u)
        ocorreu_spike = AtualizarEstadoNeuronio(neuronio, corrente_total, dt)

        SE ocorreu_spike ENTÃO
            Adicionar(vencedores, neuronio)
        FIM SE
    FIM PARA

    // Passo 4 (Opcional, para um WTA estrito): Resetar não-vencedores
    // Em uma implementação mais biológica, a própria inibição faz isso.
    // Em uma implementação de WTA explícita, podemos forçar o reset.
    SE Tamanho(vencedores) > 0 ENTÃO
        // Implementação simplificada: se alguém disparou, aumenta a inibição para os
outros
        // ou, em uma versão mais abstrata, reseta os outros.
        PARA CADA neuronio EM populacao:
            SE neuronio NAO ESTÁ EM vencedores ENTÃO
                ResetarPotencial(neuronio) // Suprime não-vencedores
            FIM SE
        FIM PARA
    FIM SE

    // Propagar os spikes dos vencedores
    PARA CADA vencedor EM vencedores:
        PropagarSpike(vencedor)
    FIM PARA
FIM PROCEDIMENTO
```

É fundamental reconhecer que a competição é mais do que um simples mecanismo

de seleção. Ela funciona como um poderoso mecanismo de **normalização da atividade**. A inibição lateral, ao reduzir a excitação de todos os neurônios em um pool de forma proporcional à atividade total desse pool, efetivamente reescala a atividade da camada. Isso impede que a atividade da rede sature (todos os neurônios disparando) ou morra (nenhum neurônio disparando). Funcionalmente, é análogo a muitos esquemas de normalização usados em deep learning (como a normalização em lote ou a função softmax), mas implementado de forma distribuída e biologicamente plausível. A competição é, portanto, um mecanismo homeostático de ação rápida, operando na escala de tempo de milissegundos para manter a atividade da rede dentro de uma faixa dinâmica saudável e computacionalmente útil.

Seção 6: A Hierarquia de Aprendizagem e Plasticidade

Um sistema computacional que apenas processa informações com uma arquitetura fixa, por mais otimizada que seja, não é verdadeiramente inteligente. A marca registrada da computação cerebral é sua capacidade de aprender e se adaptar. Essa adaptação não é um processo monolítico, mas uma sinfonia de múltiplos mecanismos de plasticidade que operam em diferentes escalas de tempo, desde milissegundos a dias, cada um com uma função computacional distinta. Este blueprint organiza esses mecanismos em uma hierarquia, desde a formação de associações locais até a otimização global da arquitetura da rede.

6.1 Fundamento da Aprendizagem: A Regra de Hebb

Na base da nossa hierarquia de aprendizagem está o postulado de Donald Hebb, elegantemente resumido como: "neurônios que disparam juntos, conectam-se".[23] A regra de Hebb é o princípio fundamental da aprendizagem associativa. Ela afirma que a força de uma sinapse entre dois neurônios aumenta se ambos os neurônios estiverem ativos simultaneamente ou em estreita sucessão temporal.

Matematicamente, a forma mais simples da regra de Hebb pode ser expressa como uma mudança no peso sináptico wij (da pré-sináptica j para a pós-sináptica i) que é proporcional ao produto de suas atividades (xj e yi):

$$\Delta w_{ij} = \eta \cdot y_i \cdot x_j$$
onde η é uma pequena taxa de aprendizagem.[60] Esta regra simples permite que a rede aprenda correlações em seus dados de entrada. Se a entrada do neurônio

j consistentemente contribui para o disparo do neurônio i, a conexão entre eles será fortalecida, formando uma associação.

No entanto, a regra de Hebb pura tem uma falha crítica: é instável. Como o fortalecimento do peso leva a uma maior ativação pós-sináptica, que por sua vez leva a um maior fortalecimento do peso, os pesos tendem a crescer sem limites em um ciclo de feedback positivo.[60] Portanto, ela deve ser vista não como uma regra de aprendizagem completa, mas como o princípio fundamental sobre o qual mecanismos mais sofisticados e estáveis são construídos.

Code snippet

```
// Pseudocódigo conceitual para a Regra de Hebb
PROCEDIMENTO AprendizagemHebbiana(sinapse, atividade_pre, atividade_pos, taxa_aprendizagem):
    // Calcula a mudança no peso
    delta_peso = taxa_aprendizagem * atividade_pos * atividade_pre

    // Atualiza o peso da sinapse
    sinapse.peso = sinapse.peso + delta_peso
FIM PROCEDIMENTO
```

6.2 Aprendizagem Causal: Plasticidade Dependente do Tempo do Spike (STDP)

A Plasticidade Dependente do Tempo do Spike (Spike-Timing-Dependent Plasticity - STDP) é um refinamento temporalmente preciso da regra de Hebb.[61] Ela não depende apenas de os neurônios dispararem "juntos", mas da ordem e do intervalo preciso entre seus spikes, tipicamente em uma escala de dezenas de milissegundos. A STDP é um mecanismo chave para a aprendizagem de sequências e relações causais na rede.

A regra canônica da STDP é a seguinte:

- **Potenciação de Longo Prazo (LTP):** Se um neurônio pré-sináptico dispara *pouco antes* (geralmente < 20 ms) de um neurônio pós-sináptico, a sinapse entre eles é fortalecida. Isso reforça as conexões que são preditivas ou causais.
- **Depressão de Longo Prazo (LTD):** Se o neurônio pré-sináptico dispara *pouco depois* do neurônio pós-sináptico, a sinapse é enfraquecida. Isso penaliza conexões que não são causalmente relacionadas.

Para implementar a STDP computacionalmente, cada sinapse precisa manter um registro da atividade recente dos neurônios pré e pós-sinápticos. Isso é frequentemente feito usando "traços sinápticos", que são variáveis que decaem exponencialmente e são incrementadas a cada spike.[62]

Code snippet

```
// Estrutura de dados para uma sinapse com STDP
ESTRUTURA SinapseSTDP:
    peso
    traco_pre_sinaptico // Traço da atividade pré-sináptica
    traco_pos_sinaptico // Traço da atividade pós-sináptica
    parametros_stdp // (A_plus, A_minus, tau_plus, tau_minus)

// Pseudocódigo para a atualização da STDP
PROCEDIMENTO AtualizarSinapseSTDP(sinapse, ocorreu_spike_pre,
ocorreu_spike_pos, dt):
    // Atualizar os traços sinápticos (decaimento exponencial)
    sinapse.traco_pre_sinaptico *= exp(-dt / sinapse.parametros_stdp.tau_plus)
    sinapse.traco_pos_sinaptico *= exp(-dt / sinapse.parametros_stdp.tau_minus)

    // Se o neurônio pré-sináptico disparou
    SE ocorreu_spike_pre ENTÃO
        // O peso diminui com base no traço pós-sináptico (LTD)
        sinapse.peso -= sinapse.parametros_stdp.A_minus * sinapse.traco_pos_sinaptico
        // Incrementa o traço pré-sináptico
        sinapse.traco_pre_sinaptico += 1.0
    FIM SE
```

```
    // Se o neurônio pós-sináptico disparou
    SE ocorreu_spike_pos ENTÃO
        // O peso aumenta com base no traço pré-sináptico (LTP)
        sinapse.peso += sinapse.parametros_stdp.A_plus * sinapse.traco_pre_sinaptico
        // Incrementa o traço pós-sináptico
        sinapse.traco_pos_sinaptico += 1.0
    FIM SE

    // Manter os pesos dentro de limites razoáveis
    sinapse.peso = Limitar(sinapse.peso, peso_min, peso_max)
FIM PROCEDIMENTO
```

6.3 Aprendizagem Guiada: Plasticidade Modulada por Dopamina

A STDP é uma forma de aprendizagem não supervisionada; ela fortalece as conexões com base em correlações locais, sem qualquer noção de se a ação resultante foi "boa" ou "ruim" para o sistema como um todo. Para uma aprendizagem orientada a objetivos, o cérebro emprega um "terceiro fator": os neuromoduladores, como a dopamina.[61]

A dopamina é fortemente associada ao sistema de recompensa do cérebro. A liberação de dopamina sinaliza que um resultado inesperadamente bom ocorreu. Este sinal de dopamina pode interagir com a STDP, transformando-a em uma forma de aprendizagem por reforço.[64] A regra de três fatores funciona da seguinte forma:

1. As sinapses mantêm um "traço de elegibilidade", que é uma memória de curto prazo de sua atividade recente de STDP (ou seja, se elas foram recentemente candidatas a LTP ou LTD).
2. Se um sinal de recompensa global (dopamina) chega enquanto este traço de elegibilidade está ativo, a mudança de peso pendente é confirmada e consolidada.
3. Se nenhuma recompensa chegar, o traço de elegibilidade decai e nenhuma mudança de longo prazo ocorre, ou a mudança pode ser revertida.

Isso permite que a rede associe ações (padrões de disparo) com resultados recompensadores, mesmo que a recompensa seja atrasada. É o mecanismo que

permite à rede aprender quais padrões de atividade levam a resultados desejáveis.

Code snippet

```
// Modificação do procedimento STDP para incluir a modulação por dopamina
PROCEDIMENTO AtualizarSinapseComDopamina(sinapse, ocorreu_spike_pre,
ocorreu_spike_pos, nivel_dopamina, dt):
    //... (cálculo dos traços pré e pós-sinápticos como antes)...

    // Calcular a mudança de peso potencial (traço de elegibilidade)
    mudanca_potencial = 0
    SE ocorreu_spike_pre ENTÃO
        mudanca_potencial -= sinapse.parametros_stdp.A_minus *
sinapse.traco_pos_sinaptico
    FIM SE
    SE ocorreu_spike_pos ENTÃO
        mudanca_potencial += sinapse.parametros_stdp.A_plus *
sinapse.traco_pre_sinaptico
    FIM SE

    // Atualizar o traço de elegibilidade da sinapse
    sinapse.traco_elegibilidade *= exp(-dt / tau_elegibilidade)
    sinapse.traco_elegibilidade += mudanca_potencial

    // Aplicar a mudança de peso real modulada pela dopamina
    // A dopamina atua como um sinal de "confirmação"
    mudanca_real_peso = taxa_aprendizagem_reforco * nivel_dopamina *
sinapse.traco_elegibilidade
    sinapse.peso += mudanca_real_peso

    //... (manter os pesos dentro dos limites)...
FIM PROCEDIMENTO
```

6.4 Estabilidade da Rede: Plasticidade Homeostática

Enquanto a STDP e a aprendizagem hebbiana promovem a instabilidade ao fortalecer seletivamente as sinapses, a **plasticidade homeostática** atua como uma força contrária e estabilizadora, operando em escalas de tempo mais lentas (horas a dias).[66] Seu objetivo é manter a atividade geral de um neurônio ou de um circuito dentro de uma faixa de operação estável e saudável, evitando a hiperexcitabilidade (que pode levar a convulsões) ou o silêncio prolongado.

O principal mecanismo homeostático é o **escalonamento sináptico (synaptic scaling)**. Ele funciona da seguinte forma [68]:

1. Cada neurônio monitora sua própria taxa de disparo média ao longo do tempo.
2. Ele compara essa taxa média com uma "taxa de disparo alvo" interna.
3. Se a taxa de disparo média estiver muito alta, o neurônio multiplica o peso de *todas* as suas sinapses de entrada por um fator menor que 1, tornando-as mais fracas e reduzindo sua excitabilidade.
4. Se a taxa de disparo média estiver muito baixa, ele multiplica os pesos por um fator maior que 1, fortalecendo-os e aumentando sua excitabilidade.

Crucialmente, este escalonamento é multiplicativo, o que significa que ele preserva as *diferenças relativas* de força entre as sinapses que foram estabelecidas pela STDP. A STDP lida com a aprendizagem de padrões específicos, enquanto o escalonamento sináptico garante que a atividade geral do neurônio permaneça estável.[70]

Code snippet

```
// Pseudocódigo para o escalonamento sináptico
PROCEDIMENTO EscalonamentoSinapticoHomeostatico(neuronio, dt):
  // Atualizar a taxa de disparo média do neurônio (filtro passa-baixa lento)
  taxa_instantanea = neuronio.spikes_recentes / tempo_janela
  neuronio.taxa_media += (dt / tau_homeostatico) * (taxa_instantanea - neuronio.taxa_media)

  // Calcular o fator de escalonamento
  erro = neuronio.taxa_alvo - neuronio.taxa_media
  fator_escalonamento = 1.0 + taxa_aprendizagem_homeostatica * erro
```

```
  // Aplicar o fator de escalonamento a todas as sinapses de entrada do neurônio
  PARA CADA sinapse_entrada EM neuronio.sinapses_de_entrada:
      sinapse_entrada.peso *= fator_escalonamento
  FIM PARA
FIM PROCEDIMENTO
```

6.5 Otimização da Arquitetura: Plasticidade Estrutural

A forma mais lenta e talvez mais profunda de plasticidade é a **plasticidade estrutural**, que envolve a criação física de novas sinapses (**sinaptogênese**) e a eliminação de sinapses existentes (**poda sináptica** ou synaptic pruning).[71] Durante o desenvolvimento, o cérebro produz uma superabundância de conexões, muitas das quais são posteriormente podadas com base na atividade e na experiência.[73] Este processo não para na infância, mas continua em menor grau ao longo da vida, otimizando a própria fiação da rede.

A poda sináptica não é aleatória. Sinapses que são fracas ou raramente usadas são marcadas para eliminação, enquanto as fortes e frequentemente ativas são estabilizadas e mantidas.[74] Isso permite que a rede refine sua topologia, removendo conexões redundantes e ineficientes e liberando recursos metabólicos e espaciais.[72]

Computacionalmente, isso pode ser modelado como um processo que opera na escala de tempo mais lenta de todas:

1. Periodicamente, o sistema avalia a "utilidade" de cada sinapse. A utilidade pode ser simplesmente seu peso absoluto ou uma medida mais complexa de sua contribuição para a informação da rede.
2. Sinapses com utilidade abaixo de um certo limiar por um período prolongado são marcadas para poda e eventualmente removidas do grafo da rede.
3. Simultaneamente, novos "brotes" sinápticos podem ser formados aleatoriamente entre neurônios próximos, criando novas conexões candidatas que serão testadas pela STDP e outros mecanismos.

Code snippet

```
// Pseudocódigo conceitual para a plasticidade estrutural
PROCEDIMENTO PlasticidadeEstrutural(rede, limiar_poda, prob_germinacao):
    // Passo 1: Poda Sináptica
    PARA CADA sinapse EM rede.sinapses:
        SE Abs(sinapse.peso) < limiar_poda ENTÃO
            // Marcar para remoção ou remover diretamente
            RemoverSinapse(rede, sinapse)
        FIM SE
    FIM PARA

    // Passo 2: Sinaptogênese
    PARA CADA neuronio_pre EM rede.neuronios:
        SE Random() < prob_germinacao ENTÃO
            // Tentar formar uma nova conexão com um neurônio pós-sináptico próximo
            neuronio_pos = EncontrarVizinhoProximo(neuronio_pre, rede)
            SE NAO rede.TemSinapse(neuronio_pre, neuronio_pos) ENTÃO
                CriarNovaSinapse(rede, neuronio_pre, neuronio_pos, peso_inicial_pequeno)
            FIM SE
        FIM SE
    FIM PARA
FIM PROCEDIMENTO
```

Esses mecanismos de plasticidade não são independentes, mas formam um sistema de controle aninhado e cooperativo. A STDP, operando em milissegundos, aprende rapidamente as correlações temporais. A modulação por dopamina, em segundos, guia essa aprendizagem em direção a objetivos recompensadores. A plasticidade homeostática, em horas ou dias, garante que a rede permaneça estável e funcional. E a plasticidade estrutural, ao longo de dias ou semanas, otimiza a própria arquitetura de hardware da rede. Juntos, eles permitem que o sistema se auto-organize e se adapte de forma robusta e eficiente a um mundo complexo e em constante mudança.

Seção 7: A Arquitetura do Sistema: Mapeamento para Hardware Neuromórfico

A tradução do blueprint algorítmico, descrito nas seções anteriores, para uma

implementação física eficiente é o desafio final. Embora uma simulação em software em CPUs ou GPUs convencionais seja possível, ela não captura a eficiência energética e o paralelismo inerentes ao modelo. A verdadeira promessa de um sistema computacional inspirado no cérebro reside no hardware neuromórfico — silício projetado especificamente para emular os princípios da computação neural. Esta seção descreve como nossa arquitetura conceitual se alinha com os paradigmas de hardware neuromórfico existentes e aborda o problema prático de mapear uma rede neural com spikes (SNN) para um substrato físico.

7.1 O Paradigma Neuromórfico: IBM TrueNorth e Intel Loihi 2

Nas últimas décadas, surgiram várias plataformas de hardware neuromórfico, com o TrueNorth da IBM e a série Loihi da Intel sendo exemplos proeminentes. Embora com abordagens de design diferentes, eles compartilham princípios fundamentais que se alinham diretamente com nosso blueprint [6]:

- **Computação Orientada a Eventos:** Ambos os chips são fundamentalmente assíncronos. A computação e a comunicação são acionadas por eventos de spike, eliminando a necessidade de um relógio global e reduzindo drasticamente o consumo de energia em estado ocioso.[6] O TrueNorth usa uma abordagem GALS (Globally Asynchronous, Locally Synchronous), enquanto o Loihi 2 avança ainda mais na direção da assincronia.
- **Paralelismo Massivo e Arquitetura Distribuída:** Ambos são processadores many-core. O TrueNorth possui 4096 "núcleos neurosinápticos", cada um com seus próprios neurônios e memória sináptica.[76] O Loihi 2 possui 128 núcleos de neurônios (NCs).[77] Esta arquitetura distribuída implementa o princípio da computação na memória, co-localizando processamento e armazenamento para minimizar o movimento de dados.
- **Escalabilidade:** As arquiteturas são projetadas para serem escaláveis. Múltiplos chips podem ser interligados para formar sistemas maiores, com o TrueNorth usando uma malha 2D e o Loihi 2 suportando topologias de malha 3D e comunicação inter-chip de alta velocidade.[76]
- **Flexibilidade e Plausibilidade Biológica:** O Loihi 2, em particular, oferece uma programabilidade significativa. Ele permite a implementação de modelos de neurônios personalizados via microcódigo, suporta spikes graduados (que podem carregar valores inteiros, não apenas binários) e possui suporte de hardware para regras de aprendizagem de três fatores, como a plasticidade modulada por

dopamina.[77] Isso o torna uma plataforma ideal para implementar os mecanismos de aprendizagem mais complexos do nosso blueprint.

Esses chips demonstram que os princípios delineados na Seção 1 não são apenas teóricos, mas podem ser realizados em silício, oferecendo ganhos de ordens de magnitude em eficiência energética para cargas de trabalho adequadas, como as baseadas em SNNs.

7.2 Mapeamento da Rede: O Desafio da Partição e do Posicionamento

Ter o hardware certo é apenas metade da batalha. O desafio prático é como mapear eficientemente uma SNN em larga escala, definida por nosso grafo G=(V,E), para os recursos finitos de um chip neuromórfico. Um chip como o Loihi 2 tem um número limitado de núcleos, e cada núcleo tem uma capacidade limitada de neurônios e sinapses que pode armazenar.[79]

O problema de mapeamento pode ser dividido em duas etapas principais:

1. **Particionamento (Clustering):** A SNN deve ser dividida em múltiplos clusters ou partições. O objetivo é que cada cluster possa ser contido dentro dos recursos de um único núcleo de hardware. A forma como essa partição é feita é crítica. Uma partição ruim pode resultar em um número excessivo de conexões *entre* os clusters. Essas conexões "globais" devem ser roteadas através da Rede-em-Chip (NoC) que interliga os núcleos, incorrendo em maior latência e consumo de energia em comparação com as conexões "locais" dentro de um mesmo núcleo.[80] Portanto, o objetivo do particionamento é **minimizar a comunicação inter-cluster**, mantendo os neurônios que se comunicam fortemente entre si no mesmo cluster.
2. **Posicionamento (Placement):** Uma vez que a rede é particionada, cada cluster deve ser atribuído a um núcleo físico específico no chip. O objetivo do posicionamento é minimizar a distância total de comunicação na NoC. Clusters que se comunicam frequentemente devem ser colocados em núcleos fisicamente próximos no chip para reduzir a latência de roteamento e a energia.

Este é um problema de otimização combinatória complexo, análogo aos problemas de particionamento de grafos e posicionamento de circuitos em design de VLSI.

7.3 Pseudocódigo: Algoritmo de Mapeamento de Rede

Dado que o problema de mapeamento ótimo é NP-difícil, abordagens heurísticas e gulosas são necessárias. O pseudocódigo a seguir descreve uma estratégia de mapeamento de alto nível, inspirada em abordagens como SpiNeMap e NeuMap.[81]

Code snippet

```
// Estruturas de dados
// GrafoSNN: representa a rede neural com neurônios e sinapses ponderadas
// HardwareSpec: descreve os recursos do chip (num_nucleos, neuronios_por_nucleo, etc.)
// Mapeamento: um dicionário que atribui cada neurônio a um núcleo específico

FUNÇÃO MapearSNNParaHardware(grafo_snn, hardware_spec):
    // Passo 1: Particionamento da Rede (Clustering)
    // O objetivo é minimizar os spikes que cruzam as fronteiras dos clusters.
    // Algoritmos de particionamento de grafos (ex: METIS) são adequados aqui.

    clusters = ParticionarGrafo(
        grafo_snn,
        num_particoes = hardware_spec.num_nucleos,
        restricao_tamanho = hardware_spec.neuronios_por_nucleo,
        objetivo = MINIMIZAR_CORTE_ARESTAS_PONDERADO_POR_TAXA_DE_SPIKE
    )

    // Passo 2: Posicionamento dos Clusters (Placement)
    // O objetivo é mapear clusters que se comunicam muito para núcleos próximos.
    // Isso pode ser resolvido com algoritmos como a quadratura ou recozimento simulado.

    // Calcular a matriz de comunicação entre clusters
    matriz_comunicacao_cluster = CalcularComunicacaoInterCluster(clusters, grafo_snn)
```

```
    // Atribuir cada cluster a um núcleo físico
    mapeamento_final = PosicionarClusters(
        clusters,
        matriz_comunicacao_cluster,
        hardware_spec.topologia_noc,
        objetivo = MINIMIZAR_CUSTO_TOTAL_COMUNICACAO
    )

    RETORNAR mapeamento_final
FIM FUNÇÃO

// Função auxiliar para o objetivo de particionamento
FUNÇÃO OBJETIVO_PARTICIONAMENTO(aresta):
    // Pondera o corte da aresta pela sua atividade esperada
    RETORNAR aresta.peso * aresta.neuronio_pre.taxa_media_disparo
```

A intuição por trás deste algoritmo é clara: para criar um mapeamento eficiente, não basta olhar para a estrutura estática da rede. É preciso considerar sua dinâmica. Ao ponderar os cortes de arestas pela taxa de disparo esperada dos neurônios pré-sinápticos, o algoritmo prioriza manter as vias de comunicação mais ativas dentro dos núcleos locais, reduzindo assim o tráfego na NoC, o que, por sua vez, diminui a latência e o consumo de energia. Este processo de mapeamento consciente da atividade é crucial para extrair o máximo desempenho do hardware neuromórfico.

Seção 8: Conclusão e Perspectivas Futuras

Este blueprint delineou uma estrutura algorítmica para um sistema de computação de alto desempenho que se baseia nos princípios fundamentais da computação neural. Afastando-se da arquitetura de von Neumann, propusemos um modelo que é massivamente paralelo, assíncrono, orientado a eventos e notavelmente eficiente em termos de energia. A essência deste sistema não reside em um único algoritmo, mas em uma complexa interação de componentes e processos que operam em múltiplas escalas de tempo e organização.

Recapitulando os pilares do nosso design:

- **A Malha Computacional:** A rede é construída sobre uma topologia de mundo pequeno e livre de escala, uma estrutura otimizada para o equilíbrio entre processamento local especializado (segregação) e comunicação global rápida (integração). Esta topologia não é um substrato passivo, mas uma forma de pré-computação que molda o fluxo de informação.
- **A Unidade de Processamento:** O neurônio com spikes, modelado pelas equações eficientes de Izhikevich, serve como uma unidade de processamento dinâmica e não linear. Sua capacidade de replicar uma vasta gama de comportamentos biológicos permite a criação de redes heterogêneas com capacidades computacionais ricas, que vão muito além da simples soma e limiar.
- **A Linguagem da Rede:** A informação é codificada em padrões temporais esparsos de spikes. Estratégias como a codificação por taxa, latência e ordem de classificação oferecem diferentes compromissos para traduzir dados do mundo real para o domínio dos spikes, com a esparsidade sendo o princípio orientador para a eficiência.
- **A Hierarquia de Aprendizagem:** A verdadeira inteligência do sistema emerge de uma hierarquia de mecanismos de plasticidade. A STDP aprende correlações causais locais, a plasticidade modulada por dopamina guia a aprendizagem em direção a objetivos recompensadores, a plasticidade homeostática garante a estabilidade da rede a longo prazo, e a plasticidade estrutural otimiza a própria arquitetura da rede ao longo do tempo.

A convergência desses princípios aponta para um novo paradigma computacional. Em vez de impor uma ordem síncrona e centralizada, este sistema permite que a ordem emerja da auto-organização de interações locais e assíncronas. A competição através da inibição lateral atua como um mecanismo de normalização em tempo real, enquanto a hierarquia de regras de aprendizagem ajusta continuamente a rede para modelar melhor a estrutura do seu ambiente.

Desafios e Perspectivas Futuras

Apesar do imenso potencial, a realização prática de SNNs em larga escala e de alto desempenho enfrenta desafios significativos. O treinamento de redes com spikes profundas permanece uma área de pesquisa ativa.[83] A natureza não diferenciável do evento de spike impede a aplicação direta do backpropagation, o algoritmo que impulsionou a revolução do deep learning. Embora métodos como o gradiente substituto (surrogate gradient) tenham mostrado sucesso, eles vêm com seus próprios custos computacionais e de memória.[85]

O futuro da computação neuromórfica dependerá da co-evolução de algoritmos,

software e hardware.

1. **Algoritmos de Aprendizagem:** O desenvolvimento de regras de aprendizagem mais eficientes e biologicamente plausíveis que possam treinar redes profundas sem as desvantagens do backpropagation através do tempo (BPTT) é crucial. A exploração de regras de aprendizagem locais e de três fatores, como as descritas neste blueprint, é uma via promissora.
2. **Frameworks de Software:** Ferramentas como Lava [87] e Brian2 [89] estão se tornando essenciais para abstrair a complexidade do hardware neuromórfico e permitir que os pesquisadores desenvolvam e testem algoritmos de forma mais produtiva. A criação de compiladores e mapeadores mais inteligentes, que possam otimizar automaticamente a partição e o posicionamento de SNNs em hardware (conforme discutido na Seção 7), será fundamental.
3. **Hardware Neuromórfico:** Chips como o Loihi 2 da Intel representam um passo significativo, mas ainda estão em fase de pesquisa.[91] As futuras gerações de hardware provavelmente verão uma maior densidade de neurônios e sinapses, maior programabilidade e suporte ainda mais integrado para mecanismos de plasticidade on-chip. A superação das limitações práticas, como a variabilidade entre dispositivos em implementações analógicas e a complexidade da programação, será essencial para a adoção em larga escala.[92]

Em conclusão, o blueprint apresentado aqui não é um destino final, mas um roteiro. Ele estabelece uma estrutura baseada em décadas de pesquisa em neurociência computacional, oferecendo um caminho para a construção de sistemas que não apenas calculam, mas aprendem, se adaptam e operam com uma eficiência que a natureza aperfeiçoou ao longo de milhões de anos. A jornada para replicar a HPC do cérebro está apenas começando, mas os princípios orientadores são claros, e o potencial para revolucionar a inteligência artificial e a computação é profundo.

Works cited

1. medicine.yale.edu, accessed July 3, 2025, https://medicine.yale.edu/lab/colon-ramos/overview/#:~:text=The%20human%20brain%20consists%20of,and%20assemble%20into%20functional%20circuits.
2. A New Field of Neuroscience Aims to Map Connections in the Brain, accessed July 3, 2025, https://hms.harvard.edu/news/new-field-neuroscience-aims-map-connections-brain
3. Basic Neural Units of the Brain: Neurons, Synapses and Action Potential - arXiv, accessed July 3, 2025, https://arxiv.org/abs/1906.01703
4. A closer look at Neuromorphic Computing | by Mrigeeshashwin | Electronics Club IITK, accessed July 3, 2025,

https://medium.com/electronics-club-iitk/a-closer-look-at-neuromorphic-computing-a16162b00ebb
5. Neural Spiking Dynamics in Asynchronous Digital Circuits - Computer Systems Lab @ Yale, accessed July 3, 2025, https://csl.yale.edu/~rajit/ps/ijcnn2013.pdf
6. TrueNorth: A Deep Dive into IBM's Neuromorphic Chip Design, accessed July 3, 2025, https://open-neuromorphic.org/blog/truenorth-deep-dive-ibm-neuromorphic-chip-design/
7. The computational power of the human brain - Frontiers, accessed July 3, 2025, https://www.frontiersin.org/journals/cellular-neuroscience/articles/10.3389/fncel.2023.1220030/full
8. How Brains Are Built- Principles of Computational Neuroscience-2 - arXiv, accessed July 3, 2025, https://arxiv.org/pdf/1704.03855
9. Parallel processing (psychology) - Wikipedia, accessed July 3, 2025, https://en.wikipedia.org/wiki/Parallel_processing_(psychology)
10. IBM Has Created A Revolutionary New Model For Computing—The Human Brain, accessed July 3, 2025, https://digitaltonto.com/2016/ibm-has-created-a-revolutionary-new-model-for-computing-the-human-brain/
11. www.nist.gov, accessed July 3, 2025, https://www.nist.gov/blogs/taking-measure/brain-inspired-computing-can-help-us-create-faster-more-energy-efficient#:~:text=Even%20though%20modern%20AI%20hardware,consuming%2020%20watts%20of%20power.
12. Brain-Inspired Computing Can Help Us Create Faster, More Energy-Efficient Devices — If We Win the Race | NIST, accessed July 3, 2025, https://www.nist.gov/blogs/taking-measure/brain-inspired-computing-can-help-us-create-faster-more-energy-efficient
13. Physics 414: Brains vs Computers, accessed July 3, 2025, https://webhome.phy.duke.edu/~hsg/414/images/brain-vs-computer.html
14. Researchers propose the next platform for brain-inspired computing | The Current - UCSB, accessed July 3, 2025, https://news.ucsb.edu/2024/021528/researchers-propose-next-platform-brain-inspired-computing
15. Neuron firing rates in humans - AI Impacts, accessed July 3, 2025, https://aiimpacts.org/rate-of-neuron-firing/
16. Metabolic Estimates of Rate of Cortical Firing - AI Impacts, accessed July 3, 2025, https://aiimpacts.org/metabolic-estimates-of-rate-of-cortical-firing/
17. Sparse Coding in Sensory Systems - Number Analytics, accessed July 3, 2025, https://www.numberanalytics.com/blog/sparse-coding-sensory-systems-ultimate-guide
18. Sparse Coding in Neural Basis - Number Analytics, accessed July 3, 2025, https://www.numberanalytics.com/blog/sparse-coding-neural-basis-consciousness
19. How can AI be more energy efficient? UB researchers turn to the ..., accessed July 3, 2025,

https://www.buffalo.edu/news/releases/2025/07/neuromorphic-computing.html
20. TrueNorth Architecture IBM's Neuromorphic Chip - Janathjsk - Medium, accessed July 3, 2025, https://janathjsk.medium.com/truenorth-architecture-ibms-neuromorphic-chip-63cbfec42b98
21. Neuromorphic Principles for Efficient Large Language Models on Intel Loihi 2 - arXiv, accessed July 3, 2025, https://arxiv.org/html/2503.18002v2
22. Computational Brain and Behavior: Bridging Neuroscience and Artificial Intelligence, accessed July 3, 2025, https://neurolaunch.com/computational-brain-and-behavior/
23. Hebbian Learning - The Decision Lab, accessed July 3, 2025, https://thedecisionlab.com/reference-guide/neuroscience/hebbian-learning
24. Harnessing Neuroplasticity in Computational Models - Number Analytics, accessed July 3, 2025, https://www.numberanalytics.com/blog/neuroplasticity-computational-models-cognition
25. Computational Modeling of Neural Plasticity for Self-Organization of Neural Networks, accessed July 3, 2025, https://www.researchgate.net/publication/261920045_Computational_Modeling_of_Neural_Plasticity_for_Self-Organization_of_Neural_Networks
26. A review of structural and functional brain networks: small world and atlas - PMC, accessed July 3, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC4883160/
27. Small-world human brain networks: Perspectives and challenges - Helab@BNU, accessed July 3, 2025, https://helab.bnu.edu.cn/wp-content/uploads/pdf/Liao_NBR2017.pdf
28. Adaptive reconfiguration of fractal small-world human brain functional networks - PNAS, accessed July 3, 2025, https://www.pnas.org/doi/10.1073/pnas.0606005103
29. Small-world and scale-free organization of voxel-based resting-state functional connectivity in the human brain - Dutch Connectome Lab, accessed July 3, 2025, http://www.dutchconnectomelab.nl/wordpress/wp-content/uploads/van_den_Heuvel2008_Small-world_and_scale-free_organization_of_voxel-based_resting-state_functional_connectivity_in_the_human.pdf
30. Izhikevich Neuron Model and its Application in Pattern Recognition - SETI Net, accessed July 3, 2025, https://www.seti.net/Neuron%20Lab/NeuronReferences/Izhikevich%20Model%20and%20backpropagation.pdf
31. Hybrid spiking models - Eugene.Izhikevich, accessed July 3, 2025, https://izhikevich.org/publications/hybrid_spiking_models.pdf
32. The Izhikevich neuron model and different firing patterns of known... - ResearchGate, accessed July 3, 2025, https://www.researchgate.net/figure/The-Izhikevich-neuron-model-and-different-firing-patterns-of-known-types-of-neurons_fig4_229086913
33. Izhikevich Neuron - Simbrain Documentation, accessed July 3, 2025, https://simbrain.net/Documentation/v3/Pages/Network/neuron/Izhikevich.html

34. A Nature-Inspired Neural Network Framework Based on an Adaptation of the Izhikevich Model Gage K. R. Hooper Inde - arXiv, accessed July 3, 2025, https://arxiv.org/pdf/2506.04247
35. The Refractory Period - Neuroscience - NCBI Bookshelf, accessed July 3, 2025, https://www.ncbi.nlm.nih.gov/books/NBK11146/
36. Refractory periods: Subphases and roles - Kenhub, accessed July 3, 2025, https://www.kenhub.com/en/library/physiology/refractory-periods
37. Synaptic delay | biochemistry - Britannica, accessed July 3, 2025, https://www.britannica.com/science/synaptic-delay
38. The measurement of synaptic delay, and the time course of acetylcholine release at the neuromuscular junction | Proceedings of the Royal Society of London. Series B. Biological Sciences - Journals, accessed July 3, 2025, https://royalsocietypublishing.org/doi/10.1098/rspb.1965.0016
39. Synapses and Neurotransmitter Receptors – Physiology - UW Pressbooks, accessed July 3, 2025, https://uw.pressbooks.pub/physiology/chapter/synapses-and-neurotransmitter-receptors/
40. Sparse coding - Scholarpedia, accessed July 3, 2025, http://www.scholarpedia.org/article/Sparse_coding
41. Sparse-Coding Variational Autoencoders - MIT Press Direct, accessed July 3, 2025, https://direct.mit.edu/neco/article-pdf/36/12/2571/2479569/neco_a_01715.pdf
42. What is the principle of sparse coding? Explain its relation to other coding schemes such as dense codes or grandmother cells, and give examples of each in the nervous system. Why is sparse coding more common higher in sensory hierarchies? - Charles Frye, accessed July 3, 2025, http://charlesfrye.github.io/FoundationalNeuroscience/48/
43. Lecture 15 Sparse Coding, accessed July 3, 2025, https://bernstein-network.de/wp-content/uploads/2021/03/Lecture-15-Sparse-coding-2020.pdf
44. snntorch.spikegen - Read the Docs, accessed July 3, 2025, https://snntorch.readthedocs.io/en/latest/snntorch.spikegen.html
45. Tutorial 1 - Spike Encoding — snntorch 0.9.4 documentation, accessed July 3, 2025, https://snntorch.readthedocs.io/en/latest/tutorials/tutorial_1.html
46. Supervised Learning With First-to-Spike Decoding in Multilayer Spiking Neural Networks - Frontiers, accessed July 3, 2025, https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2021.617862/full
47. Spike encoding techniques for IoT time-varying signals benchmarked on a neuromorphic classification task - PubMed Central, accessed July 3, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC9811205/
48. On the Future of Training Spiking Neural Networks, accessed July 3, 2025, https://www.dfki.de/fileadmin/user_upload/import/12987_ICPRAM_2023_118_CR.pdf
49. Matching pursuit - Wikipedia, accessed July 3, 2025,

https://en.wikipedia.org/wiki/Matching_pursuit
50. Matching pursuit – Knowledge and References - Taylor & Francis, accessed July 3, 2025, https://taylorandfrancis.com/knowledge/Engineering_and_technology/Engineering_support_and_special_topics/Matching_pursuit/
51. Matching pursuit and greedy algorithms | Advanced Signal Processing Class Notes | Fiveable, accessed July 3, 2025, https://library.fiveable.me/advanced-signal-processing/unit-8/matching-pursuit-greedy-algorithms/study-guide/v70OZZN9hTiUqaGT
52. Mastering Orthogonal Matching Pursuit - Number Analytics, accessed July 3, 2025, https://www.numberanalytics.com/blog/mastering-orthogonal-matching-pursuit
53. Lateral inhibition - Wikipedia, accessed July 3, 2025, https://en.wikipedia.org/wiki/Lateral_inhibition
54. Lateral Inhibition-inspired Convolutional Neural Network for Visual Attention and Saliency Detection - Association for the Advancement of Artificial Intelligence (AAAI), accessed July 3, 2025, https://cdn.aaai.org/ojs/12238/12238-13-15766-1-2-20201228.pdf
55. Suppression helps: Lateral Inhibition-inspired Convolutional Neural Network for Image Classification | OpenReview, accessed July 3, 2025, https://openreview.net/forum?id=N3kGYG3ZcTi
56. Winner-take-all (computing) - Wikipedia, accessed July 3, 2025, https://en.wikipedia.org/wiki/Winner-take-all_(computing)
57. What is Competitive Learning? - DataCamp, accessed July 3, 2025, https://www.datacamp.com/blog/what-is-competitive-learning
58. Dowsing the Winner-take-all neural network - IndiaAI, accessed July 3, 2025, https://indiaai.gov.in/article/dowsing-the-winner-take-all-neural-network
59. Hebbian Learning, accessed July 3, 2025, https://www.cs.jhu.edu/~ayuille/JHUcourses/ProbabilisticModelsOfVisualCognition2020/Lec6/HebbianYuilleKersten.pdf
60. 3.1 simple Hebbian Learning - Rice ECE, accessed July 3, 2025, https://www.ece.rice.edu/~erzsebet/ANNcourse/handouts502/course-cf-3.pdf
61. Spike-timing-dependent plasticity - Wikipedia, accessed July 3, 2025, https://en.wikipedia.org/wiki/Spike-timing-dependent_plasticity
62. NESTML STDP windows tutorial - Read the Docs, accessed July 3, 2025, https://nestml.readthedocs.io/en/latest/tutorials/stdp_windows/stdp_windows.html
63. Dopaminergic Neuromodulation of Spike Timing Dependent Plasticity in Mature Adult Rodent and Human Cortical Neurons, accessed July 3, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC8102156/
64. Modulation of Spike-Timing Dependent Plasticity: Towards the Inclusion of a Third Factor in Computational Models - Frontiers, accessed July 3, 2025, https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2018.00049/full
65. arXiv:2109.05539v5 [cs.NE] 7 Jul 2022, accessed July 3, 2025,

https://arxiv.org/pdf/2109.05539
66. Homeostatic mechanisms regulate distinct aspects of cortical circuit dynamics - PNAS, accessed July 3, 2025, https://www.pnas.org/doi/10.1073/pnas.1918368117
67. Homeostatic Synaptic Plasticity: Local and Global Mechanisms for Stabilizing Neuronal Function - PubMed Central, accessed July 3, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC3249629/
68. Unlocking Homeostatic Plasticity - Number Analytics, accessed July 3, 2025, https://www.numberanalytics.com/blog/homeostatic-plasticity-computational-neuroscience-guide
69. Unlocking Synaptic Scaling Secrets, accessed July 3, 2025, https://www.numberanalytics.com/blog/ultimate-guide-synaptic-scaling-computational-neuroscience
70. The Self-Tuning Neuron: Synaptic Scaling of Excitatory Synapses - PMC - PubMed Central, accessed July 3, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC2834419/
71. Synaptogenesis and synaptic pruning | Intro to Brain and Behavior Class Notes - Fiveable, accessed July 3, 2025, https://library.fiveable.me/introduction-brain-behavior/unit-6/synaptogenesis-synaptic-pruning/study-guide/fmt6bYol8By4DBlr
72. The information theory of developmental pruning: Optimizing global network architectures using local synaptic rules | PLOS Computational Biology, accessed July 3, 2025, https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1009458
73. Core Concept: How synaptic pruning shapes neural wiring during development and, possibly, in disease - PubMed Central, accessed July 3, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC7368197/
74. Dynamically Optimizing Network Structure Based on Synaptic Pruning in the Brain - Frontiers, accessed July 3, 2025, https://www.frontiersin.org/journals/systems-neuroscience/articles/10.3389/fnsys.2021.620558/full
75. Anandtech: "Intel's First 4nm EUV Chip, Ready Today: Loihi 2 for Neuromorphic Computing" : r/hardware - Reddit, accessed July 3, 2025, https://www.reddit.com/r/hardware/comments/pylq4h/anandtech_intels_first_4nm_euv_chip_ready_today/
76. A Look at TrueNorth - IBM - Neuromorphic Chip, accessed July 3, 2025, https://open-neuromorphic.org/neuromorphic-computing/hardware/truenorth-ibm/
77. A Look at Loihi 2 - Intel - Open Neuromorphic, accessed July 3, 2025, https://open-neuromorphic.org/neuromorphic-computing/hardware/loihi-2-intel/
78. Taking Neuromorphic Computing with Loihi 2 to the Next Level Technology Brief - Intel, accessed July 3, 2025, https://download.intel.com/newsroom/2021/new-technologies/neuromorphic-computing-loihi-2-brief.pdf
79. bio-realistic neural network implementation on loihi 2 with izhikevich neurons - arXiv, accessed July 3, 2025, https://arxiv.org/pdf/2307.11844

80. Mapping Spiking Neural Networks to Neuromorphic Hardware - UC Irvine, accessed July 3, 2025, https://sites.socsci.uci.edu/~jkrichma/balaji-mappingsnn-ieeevlsi2020.pdf
81. Optimal Mapping of Spiking Neural Network to Neuromorphic Hardware for Edge-AI - MDPI, accessed July 3, 2025, https://www.mdpi.com/1424-8220/22/19/7248
82. Mapping Spiking Neural Networks to Neuromorphic Hardware | Request PDF - ResearchGate, accessed July 3, 2025, https://www.researchgate.net/publication/337550752_Mapping_Spiking_Neural_Networks_to_Neuromorphic_Hardware
83. Direct learning-based deep spiking neural networks: a review - PMC, accessed July 3, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC10313197/
84. Deep Learning in Spiking Neural Networks - arXiv, accessed July 3, 2025, http://arxiv.org/pdf/1804.08150
85. Advancing Training Efficiency of Deep Spiking Neural Networks through Rate-based Backpropagation - NIPS, accessed July 3, 2025, https://proceedings.neurips.cc/paper_files/paper/2024/file/d1bdc488ec18f64177b2275a03984683-Paper-Conference.pdf
86. Direct Training High-Performance Deep Spiking Neural Networks: A Review of Theories and Methods - arXiv, accessed July 3, 2025, https://arxiv.org/html/2405.04289v2
87. Walk through Lava — Lava documentation - Lava framework, accessed July 3, 2025, https://lava-nc.org/lava/notebooks/end_to_end/tutorial00_tour_through_lava.html
88. lava-nc/lava: A Software Framework for Neuromorphic Computing - GitHub, accessed July 3, 2025, https://github.com/lava-nc/lava
89. Brian 2 documentation — Brian 2 0.0.post128 documentation, accessed July 3, 2025, https://brian2.readthedocs.io/
90. The Brian Simulator | The Brian spiking neural network simulator, accessed July 3, 2025, https://briansimulator.org/
91. Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook - Dynamic field theory, accessed July 3, 2025, https://dynamicfieldtheory.org/upload/file/1631291311_c647b66b9e48f0a9baff/DavisEtAl2021.pdf
92. The Promise and Pitfalls of Neuromorphic Computers - EE Times, accessed July 3, 2025, https://www.eetimes.com/the-promise-and-pitfalls-of-neuromorphic-computers/