EvoSpikeNet - 分散型進化ニューロモーフィックフレームワーク

著者: Masahiro Aoki
ドキュメントID；MT2025-AI-03-002
ORCID ID：0009-0007-9222-4181
所属: Moonlight Technologies 株式会社

文書バージョン	作成日	作成者	概要
Ver 2.0	2025年9月25日	Masahiro Aoki	実装状況とプロジェクトの方向転換を反映した全面改訂版
Ver 1.0	2025年7月15日	Masahiro Aoki	初版

改訂にあたっての注記

本書は、EvoSpikeNetプロジェクトの初期設計（Ver 1.0）を、実際の開発成果と現在の技術的ロードマップに基づき全面的に改訂したものです。初期構想段階のいくつかの仮説は、開発の過程でより現実的かつ高性能なアプローチへと刷新されました。

1. **アーキテクチャの刷新**: 当初検討されたneat-pythonのような集団ベースの進化的アルゴリズム（EA）は採用されていません。プロジェクトは、単一ネットワーク内での学習（可塑性ルールや代理勾配法）を中核とする、より現代的なアーキテクチャへと方針転換し実装されています。
2. **SNNコアの性能**: SNNコアは「純粋なPython」実装ではなく、業界標準の高性能計算ライブラリである**PyTorch**を基盤としています。これにより、GPUアクセラレーションに完全対応し、スパース行列（CSR形式）の活用によるメモリ効率の最適化も実現しています。
3. **プロジェクトの進捗**: プロジェクトは活発に開発が進められています。当初のスコープを超え、「スパイキングTransformer言語モデル」の実装、Web UIによる高度な可視化・分析ツールの提供、LLMを活用したデータ蒸留といった先進的な機能が既に実装・テスト済みです。

健全に進行しているプロジェクトの現状と、次のマイルストーンであるマルチモーダル機能の実用化に向けた計画を定義するものです。

1. 序論

1.1. 本書の目的

本書は、次世代AIシミュレーション・推論フレームワーク「EvoSpikeNet」の**現在の実装状況を反映した**基本設計とアーキテクチャを定義するものです。開発チーム、プロジェクトマネージャー、および技術的ステークホルダーが、システムの全体像、主要コンポーネント、技術的要件を正確に共有し、今後の開発（特にマルチモーダル機能）を円滑に推進することを目的とします。

1.2. プロジェクトビジョン

エネルギー効率と自己進化する適応性がAIの性能を定義する未来を創造し、地球規模の課題解決に貢献する持続可能な知性を構築する。

1.3. スコープ

本設計書が対象とするスコープは以下の通りです。

- **実装済みコア機能**: PyTorchベースのSNNシミュレーションコア、動的グラフ進化エンジン（STDP等）、インタラクティブなWeb UIを備えた可視化エンジン。
- **実装済み拡張機能**: エネルギー駆動型コンピューティング（AEG）、量子インスパイアード・ニューロン（一部除外）、snnTorchを基盤とする高機能な**SpikingEvoSpikeNetLM**（スパイキング言語モデル）、LLMデータ蒸留モジュール。
- **実装が中止された機能**: 安定性確保のため、**分散ニューロモーフィックグリッド**の実装は中断されました。
- **将来のスコープ**: 現在開発中の**マルチモーダル機能**の基本設計。

2. システムアーキテクチャ

2.1. 設計思想

EvoSpikeNetは、以下の3つの設計思想に基づいています。

- **イベント駆動**: 脳のように、情報が変化したとき（スパイクが発生したとき）にのみ計算を行うことで、不要な電力消費を劇的に削減します [5]。
- **自己組織化**: ネットワーク構造が固定的ではなく、経験を通じて接続（シナプス）を動的に生成・削除することで、環境に継続的に適応します [9]。

- **スケーラビリティ**: 単一ノード内でのマルチGPU活用を前提とし、強力な計算資源上で大規模なシミュレーションを実行可能なアーキテクチャを目指します。

2.2. 高レベルアーキテクチャ

EvoSpikeNetは、以下の主要コンポーネントから構成される階層的アーキテクチャを採用しています。

レイヤー	コンポーネント	役割
アプリケーション層	EvoSpikeNet Insight Engine (Web UI)	SNNの内部状態、進化プロセス、言語モデルの注意機構等を**ブラウザからインタラクティブに可視化・分析**する。
フレームワーク層	動的グラフ進化エンジン SpikingEvoSpikeNetLM LLMデータ蒸留モジュール	学習則に基づきネットワーク構造を更新。 Transformerベースのスパイク駆動言語モデル。 高品質な合成データを生成。
コアエンジン層	SNNシミュレーションコア	**PyTorchベース。**整数演算とスパース接続行列を用いて、スパイク伝播をGPU/CPU上で高速に計算する。
インフラ層	**[削除]** 分散ニューロモーフィックグリッド	**[実装中断]** 開発初期に検討されたが、技術的困難（デッドロック）により、プロジェクトの安定性を優先して削除。

3. コアコンポーネント - 詳細設計

3.1. SNNシミュレーションコア

- **ニューロンモデル**: Leaky Integrate-and-Fire (LIF) モデルを基本とします。

$$\tau \frac{dV(t)}{dt} = -V(t) + RI(t)$$

- **数値精度**: シナプス重みとスパイクはINT8、膜電位はINT16で表現し、浮動小数点演算を排除します。
- **接続表現**: 接続行列は、GPUでのスパース行列ベクトル積（SpMV）に最適化されたCSR (Compressed Sparse Row) 形式で保持します。
- **実装**: torch.sparse_csr_tensorを全面的に活用し、PyTorchの性能を最大限に引き出します。

3.2. 動的グラフ進化エンジン

- **可塑性ルール**:
  - **STDP (スパイクタイミング依存可塑性)**: ニューロン間の発火タイミングの相関に基づき、シナプス結合を強化・弱化させます [9]。

$$\Delta w_{ij} = \begin{cases} A_+ \exp\left(-\frac{\Delta t_{ij}}{\tau_+}\right) \text{ if } \Delta t_{ij} > 0 \\ -A\_\exp\left(-\frac{|\Delta t_{ij}|}{\tau\_}\right) \text{ if } \Delta t_{ij} < 0 \end{cases}$$

  - **ホメオスタティック可塑性**: ネットワーク全体の活動が安定するように、ニューロンの平均発火率を一定の範囲（例: 10Hz）に保つ負のフィードバック機構を実装します。
- **更新アルゴリズム**: スパーステンソルの高コストな再構築を避けるため、追加・削除されるエッジ情報を一時バッファに格納し、Nタイムステップごとにバッチ更新するアルゴリズムを採用しています。

3.3. 言語解釈モジュール (SpikingEvoSpikeNetLM)

SNNの時間情報処理能力を自然言語処理（NLP）に応用するため、当初の構想を大幅に発展させた、snnTorchベースのTransformer型言語モデルを実装済みです。

- **基盤**: Transformerアーキテクチャをスパイク領域で効率的に動作させることを目的とします。
- **時間適応スパイクエンコーディング (TAS-Encoding)**: テキストトークンをスパイク列に変換する独自の手法です。トークンの意味的重要度に応じてスパイク密度を動的に調整します。

  スパイク確率

$$P(s_{i,t}) = \sigma\left(\alpha \cdot \frac{|e_i|}{\tau_{\text{window},i}}\right)$$

- **ChronoSpikeAttention**: 標準的なScaled Dot-Product Attention (

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

  の概念をスパイク領域で近似・発展させた、ハイブリッドなアテンション機構です。
- **学習と制御**: メタ学習（MetaSTDP）やエネルギー効率（AEG）を考慮した高度な制御機構を統合しています。また、代理勾配法による訓練も可能です。

3.4. EvoSpikeNet Insight Engine (可視化)

AIの内部動作を理解するための、インタラクティブなWeb分析ツールとして実装済みです。

- **技術スタック**: Plotly/Dashをフロントエンドとし、バックエンドでMatplotlibやPyGraphistryのレンダリング技術を統合しています。
- **主要機能**:
  - **スパイク・ラスタープロット**: どのニューロンが、いつ発火したかを時系列で表示。
  - **動的コネクティビティ・グラフ**: 活性化したシナプスをリアルタイムでハイライト。
  - **言語モデル分析**: テキスト生成時の内部スパイク活動やアテンションの重みをステップごとに可視化し、「思考の連鎖」を追跡。

3.5. [実装中断] 分散ニューロモーフィックグリッド

- **経緯**: 当初、torch.distributedやSpiNNaker[2]を参考に、マルチノードでの大規模シミュレーションが計画されました。Pythonのmultiprocessingを用いたプロトタイプが開発されましたが、解決困難なデッドロック問題に直面しました。
- **決定**: プロジェクト全体の安定性と開発速度を優先するため、この機能の実装は公式に**中止**されました。スケーラビリティは、単一ノード内でのマルチGPU利用によって確保します。

4. 拡張機能

- **メタ可塑性**:
  - **概要**: 学習ルールそのものを進化させるメタ学習機能。STDPのパラメータ（例: τ_+）

をタスクの成功度（Reward）に応じて動的に調整します。実装済みです。

$$\tau_+ \leftarrow \tau_+ + \eta \cdot (\text{Reward} - \text{Reward}_{\text{baseline}})$$

- **エネルギー駆動型コンピューティング (AEG)**:
  - **概要**: エネルギー消費を計算プロセスを制御する要因として組み込みます [10]。各ニューロンが仮想「エネルギー」を消費し、省エネな情報処理経路を自律的に見つけ出します。実装済みです。
- **量子インスパイアード・ニューロン**:
  - **概要**: 量子重ね合わせの概念を用いて、情報の表現能力を高めます [8]。
  - **実装状況**: EntangledSynchronyLayer（ニューロン群の同期制御）など一部機能は実装済みです。しかし、GraphAnnealingRuleは低レベルのクラッシュを引き起こしたため、安定性確保のために削除されました。**部分的な実装**となります。

5. 技術仕様

カテゴリ	**仕様**
コアアーキテクチャ	動的スパイクニューラルネットワーク (SNN)
数値精度	INT8 (シナプス重み、スパイク), INT16 (膜電位)
基盤フレームワーク	**PyTorch 2.1.0+, SNNTorch, SpikingJelly, transformers**
GPU技術	CUDA 11.8+, CuPy, CuSparse
分散技術	**[削除]**
ハードウェア要件	開発: NVIDIA RTX 3060 (12GB VRAM以上) 大規模: NVIDIA A100 (40GB VRAM) x100, InfiniBand 10Gbps エッジ: Intel Loihi 2 (Lava)
可視化バックエンド	Matplotlib, Plotly/Dash, PyGraphistry

6. 応用例と性能目標

応用分野	タスク / データセット	性能目標 (比較対象モデル)
自律システム	ドローン障害物回避 / DVS128	衝突回避率99%以上、10ms未満の反応速度
エッジAI	心電図異常検知 / PhysioNet MIT-BIH	ResNet-18比でエネルギー消費を85%以上削減しつつ、F1スコア95%以上を維持
言語解釈	感情分析 / SST-2	**[実装済み]** SpikingEvoSpikeNetLMにより、BERT-base比で電力効率を大幅に改善しつつ、同等以上の精度を目指す。
計算論的神経科学	脳シミュレーション	マウス脳規模（約1億ニューロン）のリアルタイム実行（**ストレッチゴール**）

7. 限界と将来の課題

- **スケーラビリティの現実性**: 分散グリッドを中断したため、単一ノードのメモリと計算能力が大規模シミュレーションの直接的な上限となります。
- **スパース性の限界**: 接続率が極端に低いネットワークでは、GPUのメモリアクセスパターンが非効率になり、演算器の性能を十分に引き出せない可能性があります。
- **実装の未成熟性**: 量子インスパイアード機能など、一部の拡張機能は研究段階にあり、製品レベルの安定性を保証するにはさらなる基礎研究が必要です。
- **競合との差別化**: IntelのLoihi 2[1]のような専用ハードウェアは、エネルギー効率で本フレームワークを上回る可能性があります。EvoSpikeNetの強みは、**汎用GPU上での柔軟性とアクセシビリティ**にあります。
- **次期開発計画**: 現在の最優先課題は、**マルチモーダル機能の実装**です。MultiModalEvoSpikeNetLMのプロトタイプは完成していますが、実データでの学習、推論機能の実装が次のマイルストーンとなります。

8. 参考文献

[1] Davies, M. et al. (2018). Loihi: A Neuromorphic Manycore Processor with On-Chip Learning.

IEEE Micro.
[2] Furber, S. B. et al. (2014). The SpiNNaker project. Proceedings of the IEEE.
[3] Li, J. et al. (2022). Scaling Up Dynamic Graph Representation Learning via Spiking Neural Networks. arXiv:2208.10364.
[4] Li, Y. et al. (2024). Brain-Inspired Spiking Neural Networks for Energy-Efficient Object Detection. CVPR 2025 (to be published).
[5] Eshraghian, J. K. et al. (2023). Training Spiking Neural Networks Using Lessons From Deep Learning. arXiv:2301.09174.
[6] Fang, W. et al. (2023). SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence. Science Advances.
[7] Li, S. et al. (2020). PyTorch Distributed: Experiences on Accelerating Data Parallel Training. Proceedings of the VLDB Endowment.
[8] Zeng, Y. et al. (2020). Quantum Superposition Inspired Spiking Neural Network. arXiv:2010.12197.
[9] Zenke, F., & Gerstner, W. (2017). Hebbian plasticity requires compensatory processes on multiple timescales. Philosophical Transactions of the Royal Society B.
[10] Hawkins et al., "Neuromorphic Computing: From Devices to Systems," Nature Reviews EE, vol. 1, 2023.