SNNプロジェクト: 学習・推論コマンドガイド

このドキュメントでは、モデルの学習（Training）と推論・デモ（Inference/Demo）を実行するための主要なコマンドについて解説します。 プロジェクトディレクトリのルートで実行してください。

1. 学習 (Training)

モデルを一から学習、あるいは継続学習させるためのコマンドです。

汎用トレーナー

CLIまたはスクリプト経由で、設定ファイル (`configs/`) を指定して学習を開始します。

```
# SNN CLIを使用する場合（推奨）
snn-cli gradient-train --model_config configs/models/stable_small_snn.yaml --data

# 直接スクリプトを実行する場合
python scripts/training/train.py --config configs/experiments/brain_v14_config.ya
```

タスク特化型学習スクリプト

特定のタスクやデータセットに特化した学習スクリプトです。

- **MNIST SNN学習**:

  ```
  python scripts/training/train_mnist_snn.py
  ```

- **CIFAR-10 Bio-PC (Predictive Coding) 学習**:

  ```
  python scripts/training/train_bio_pc_cifar10.py
  ```

- **Spiking VLM (Vision-Language Model) 学習**:

  ```
  python scripts/training/train_spiking_vlm.py
  ```

- **Planner (推論エンジン) 学習**:

  ```
  python scripts/training/train_planner.py
  ```

2. 推論・デモ (Inference & Demo)

学習済みモデルを使って推論を行ったり、対話デモを動かしたりします。

CLIによる推論

```
# 単一テキストの推論
snn-cli predict --text "Hello SNN" --model_path models/checkpoints/best_model.pt

# チャットモード
snn-cli chat --model_config configs/models/brain_v4_synesthesia.yaml
```

Webアプリ/APIサーバー

```
# FastAPIサーバー起動
python app/main.py
```

統合デモ

視覚・言語・運動野を統合したデモを実行します。

```
python app/unified_perception_demo.py
```

3. 高度な学習パラダイム (Advanced Paradigms)

通常の勾配学習（Backpropagation）以外の、生物学的・効率的な学習手法です。

⚡ STDP (Spike-Timing Dependent Plasticity)

教師なし学習の一種で、スパイクのタイミングに基づいてシナプス結合を強化・減衰させます。

```
python scripts/experiments/learning/run_stdp_learning.py
```

🧠 SCAL (Statistical Centroid Alignment Learning)

勾配計算を行わず、統計的な重心アライメントによって高速に学習する独自手法です。

```
python scripts/training/run_improved_scal_training.py \
    --config configs/templates/base_config.yaml \
    --model_config configs/models/small.yaml \
    --data_path data/smoke_test_data.jsonl \
    --override_config "training.epochs=10" \
    --override_config "training.batch_size=4" \
    --override_config "training.gradient_based.type=standard"

# 自動調整 (Auto-tune)
python scripts/optimization/auto_tune_efficiency.py \
    --model-config configs/models/small.yaml \
    --n-trials 20
```

💧 蒸留ワークフロー (Distillation Workflow)

データ生成から蒸留学習までの完全なフローです。

```
# 1. 古いデータを削除（クリーンな状態で再作成）
rm -rf precomputed_data/smoke_distill

# 2. 蒸留データの再生成
python scripts/data/prepare_distillation_data.py  \
    --input_file data/smoke_test_data.jsonl \
    --output_dir precomputed_data/smoke_distill \
    --teacher_model gpt2

# 3. 蒸留学習の実行
python scripts/training/train.py \
    --model_config configs/models/bit_rwkv_micro.yaml \
    --data_path precomputed_data/smoke_distill/distillation_data.jsonl \
    --paradigm gradient_based \
    --override_config "training.gradient_based.type=distillation" \
    --override_config "training.gradient_based.distillation.teacher_model=gpt2"
```

4. 進化と自己改善 (Evolution & Self-Improvement)

Phase 6以降のシステムでは、単なる「学習」を超え、**「経験」と「進化」**によってモデルが自律的に更新されます。

A. 自己修正による適応 (On-Chip Self-Correction)

バックプロパゲーション（勾配法）を使わず、稼働中にリアルタイムでシナプス荷重を調整します。

- **実行コマンド**:

  ```
  python scripts/experiments/systems/run_phase6_agi_prototype.py
  ```

- **メカニズム**:
  - **R-STDP (Reward-modulated STDP)**: 報酬信号に基づいて、局所的なヘブ則学習を変調します。
  - 外部からの教師データは不要で、環境からのフィードバックのみで適応します。

B. 社会的学習と文化継承 (Social Learning)

個体単独の学習ではなく、集団での合意形成を通じて知識を獲得します。

- **実行コマンド**:

  ```
  python scripts/experiments/systems/run_phase7_civilization.py
  ```

- **メカニズム**:

- **Meme Propagation**: 有用と判断された概念ベクトル（Meme）は `CultureRepository` に保存されます。
- **Knowledge Retrieval**: 新しく生まれたエージェントは、このリポジトリから知識をロードした状態で開始できます。

C. 再帰的自己改善 (Recursive Self-Improvement)

遺伝的アルゴリズムとメタ学習を組み合わせ、システム自体が次世代のシステムを設計・生成します。

- **実行コマンド**:

```
python scripts/experiments/systems/run_phase8_singularity.py
```

- **メカニズム**:
  - **Mutation**: パラメータ空間およびハイパーパラメータにランダムな変異を加えます。
  - **Selection**: 仮想環境でのタスク実行スコアに基づき、最も適応度の高い個体を選択します。
  - **Hot-Swap**: 稼働中のOSカーネル上で、脳モデルを即座に最新版へ差し替えます。