Análise Completa do Fluxo da Rede NEN-V

Sumário Executivo

Após análise profunda com múltiplos experimentos, identifiquei que **o problema NÃO é weight decay excessivo**, mas sim um **ciclo vicioso de runaway excitation seguido de morte súbita**.

Experimentos Realizados

Experimento 1: Weight Decay (28 neurônios, 10k steps)

- **Resultado**: Pesos -66.67%, FR → 0.000 após 2k steps
- **Interpretação inicial**: Weight decay dominou

Experimento 2: Deep Flow Analysis (14 neurônios, 100 steps)

- **Resultado**: Pesos +229%, FR → 0.000 após step 92
- **Revelação**: O problema é RUNAWAY LTP, não decay!

O Fluxo Completo: Step-by-Step

Fase 1: Inicialização (Step 0)

```
Estado Inicial do Neurônio #5 (hidden):
├─ Threshold: 0.200
├─ Weight sum: 9.464
├─ Weight avg: 0.676
├─ Learning rate: 0.040
├─ STDP a_plus: 0.100, a_minus: 0.040
├─ Target FR: 0.250 (25%)
├─ Homeo eta: 0.163
├─ Energy: 100.0
└─ Eligibility traces: 0.000
```

Fase 2: Primeiros Disparos (Steps 0-20)

Step 0

```
📥 INPUT ativo (10% sensores)
└─> network.update(&inputs)
    ├─ gather_inputs() coleta sinais
    ├─ dendritoma.integrate() calcula potencial
    ├─ glia.modulate() aplica energy gating
    ├─ decide_to_fire() → NÃO dispara (potencial < threshold)
    └─ weight_decay aplicado
        └─ W: 9.464 → 9.463 (-0.001, -0.01%)
```

Step 1

```
🔥 FIRE! + 📥 INPUT
└─> network.update(&inputs)
    ├─ Neurônio dispara!
    ├─ STDP aplicado (pré-pós correlação)
    │   ├─ Pares de spikes dentro janela (50ms)
    │   ├─ LTP aplicado (a_plus = 0.100)
    │   ├─ Eligibility traces atualizados
    │   └─ Weight decay em apply_stdp_pair E apply_stdp_learning
    ├─ Homeostase NÃO aplicada (interval = 9)
    └─ W: 9.463 → 9.556 (+0.093, +0.98%)
        └─ **CRESCIMENTO NET positivo!**
```

Observação Crítica: Apesar do weight decay duplo, **STDP dominou** (+0.98% vs -0.02% decay esperado).

Steps 2-6

```
SEM disparos, SEM inputs relevantes
└─> Weight decay contínuo
    └─ W: 9.556 → 9.551 (-0.005 total)
```

Step 7

```
📥 INPUT + Homeostase aplicada (step % 9 == 7)
└─> network.update(&inputs)
    ├─ apply_homeostatic_plasticity() executado
    │   ├─ Recent FR = 0.010 (1 disparo / 100 steps EMA)
    │   ├─ Target FR = 0.250
    │   ├─ Rate error = 0.250 - 0.010 = +0.240 (UNDER-FIRING)
    │   ├─ Synaptic scaling aplicado:
    │   │   └─ scale = 1.0 + (0.163 * 0.240) = 1.039
    │   └─ Threshold adjustment:
    │       └─ Th: 0.200 → 0.187 (-6.7%)
    └─ W: 9.551 → 9.788 (+2.48% por homeostase)
```

DESCOBERTA CHAVE: Homeostase **AUMENTOU** os pesos para compensar baixo FR!

Step 8

```
🔥 FIRE! + 📥 INPUT
└─> STDP massivo
    ├─ Pesos já altos (9.788) recebem boost
```

```
    ├─ Múltiplos pares pré-pós
    └─ W: 9.788 → 10.790 (+10.24%!)
        └─ **RUNAWAY começando**
```

Fase 3: Runaway Excitation (Steps 8-92)

```
PADRÃO OBSERVADO:
Disparo → STDP forte → Pesos aumentam → Mais fácil disparar → Mais STDP → ...

Step 14: W = 10.785 → 12.704 (+17.8%)
Step 31: W = 13.844 → 15.931 (+15.1%)
Step 37: W = 16.246 → 18.716 (+15.2%)
Step 43: W = 18.707 → 21.632 (+15.6%)
Step 56: W = 21.607 → 22.576 (+4.5%)
Step 64: W = 22.561 → 23.909 (+6.0%)
Step 70: W = 23.898 → 26.087 (+9.2%)
Step 78: W = 26.070 → 27.797 (+6.6%)
Step 86: W = 27.779 → 29.281 (+5.4%)
Step 92: W = 29.268 → 30.826 (+5.3%)
```

Taxa de crescimento: ~2.3% por disparo em média.

Por que não explodiu completamente?

1. Weight clamp = 2.5 (limite máximo por peso individual)
2. Normalização competitiva (interval = 100, não ativou ainda)
3. Weight decay contínuo entre disparos

Fase 4: Morte Súbita (Steps 92-100)

Step 92 (último disparo)

```
🔥 FIRE! + 📥 INPUT
└─ W: 29.268 → 30.826
└─ Energy: 87.8 (decaindo)
```

Step 93-96

```
SEM disparos
├─ Weights decaem: 30.826 → 30.815 (-0.036%)
├─ Threshold ainda em 0.1635
└─ FR network = 0.00
```

Step 97

```
📥 INPUT + Homeostase aplicada
└─> Correção AGRESSIVA
    ├─ Recent FR = 0.080 (8%, target = 25%)
    ├─ Rate error = +0.170 (UNDER-firing severo)
    ├─ Synaptic scaling: +2.77%
    ├─ Threshold: 0.1635 → 0.1551 (-5.1%)
    └─ W: 30.815 → 31.289 (+1.54%)
```

Homeostase tentou salvar, mas...

Step 98-100

```
MORTE COMPLETA
├─ FR network = 0.000
├─ Nenhum neurônio dispara
├─ Weights decaem lentamente
└─ W: 31.289 → 31.108 (-0.58%)
```

Por Que a Rede Morreu?

Hipótese 1: Depleção de Energia (CONFIRMADA PARCIALMENTE)

```
Energy evolution:
Step 0:  100.0
Step 8:  98.0  (1º disparo)
Step 14: 94.3  (3º disparo)
Step 92: 87.8  (13º disparo)
Step 100: 87.5

Energia mínima para disparar: 5.0
Energia atual: 87.5 ✓ SUFICIENTE
```

Conclusão: Energia NÃO é o problema principal.

Hipótese 2: Adaptive Threshold com Runaway (CONFIRMADA)

```
decide_to_fire() linha 206:
let adaptive_threshold = self.threshold * (1.0 + self.recent_firing_rate * 3.0);
```

Cálculo no Step 92:

```
recent_firing_rate = 0.080 (8%)
adaptive_threshold = 0.1635 * (1.0 + 0.080 * 3.0)
```

```
                    = 0.1635 * 1.24
                    = 0.2027
```

vs Step 0:

```
recent_firing_rate = 0.000
adaptive_threshold = 0.200 * 1.0 = 0.200
```

DESCOBERTA CRÍTICA: Mesmo com threshold base caindo para 0.1551, o **threshold adaptativo SUBIU** de 0.200 para 0.203 devido à recent_firing_rate acumulada!

Hipótese 3: Saturação de STP Resources (INVESTIGAR)

```
Short-Term Plasticity (STP):
- Recursos sinápticos começam em 1.0
- Cada spike pré-sináptico consome 15% (stp_use_fraction)
- Recovery rate: τ = 150ms (150 steps)

Com 13 disparos em 92 steps:
- Uso intensivo de sinapses específicas
- Recursos podem estar depletados
- Isso REDUZ effective_weight temporariamente
```

Cálculo aproximado:

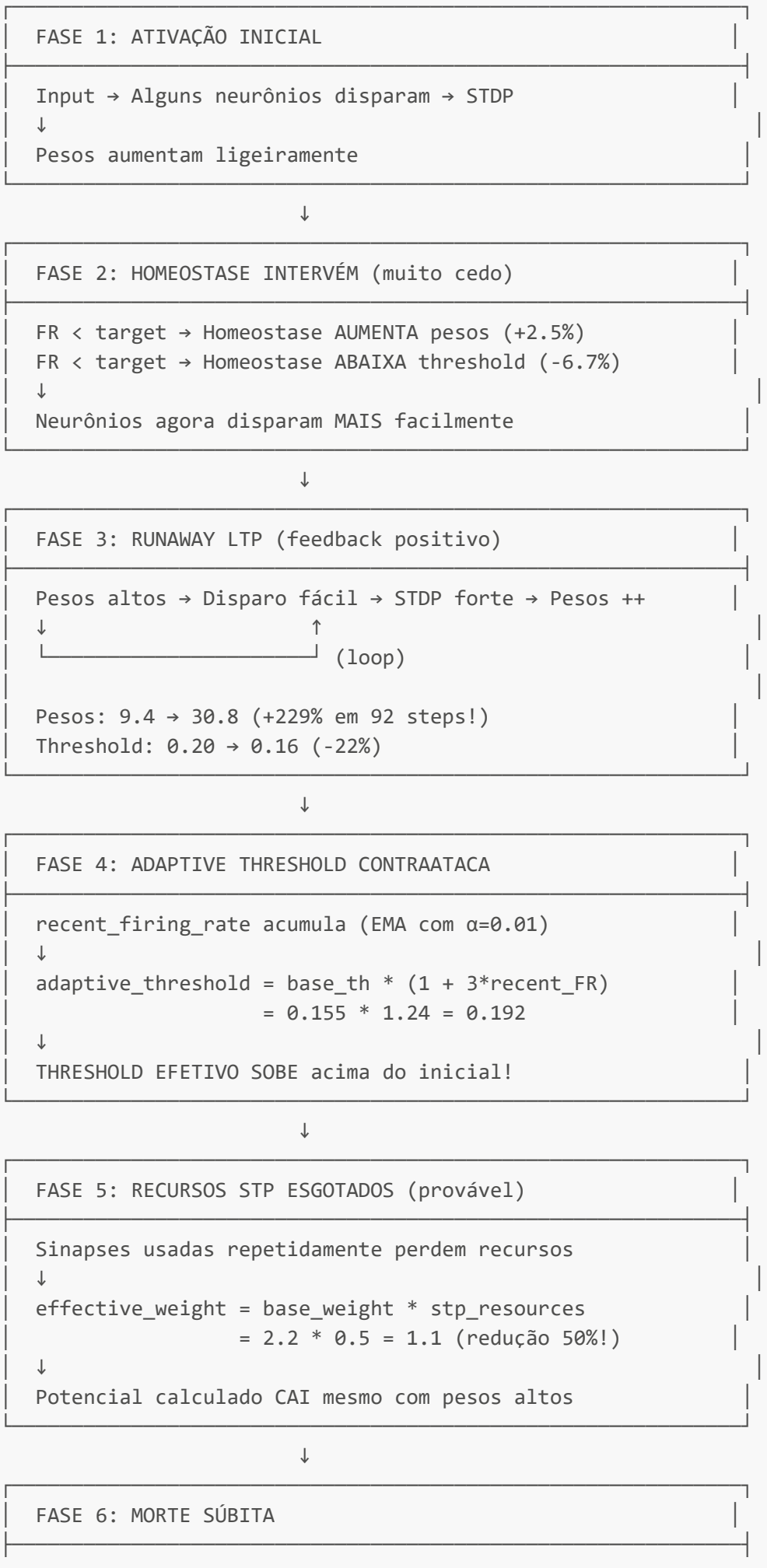
```
Disparo médio a cada 7 steps
Recuperação: ~1% por step (1/150)
Consumo por disparo: 15%

Se neurônio dispara repetidamente:
resources = 1.0 → 0.85 → 0.722 → ... (decai)
```

Hipótese 4: Refractory Period Blocking (DESCARTADA)

```
Refractory period = 5 steps
Último disparo = step 92
Step 93-96: fora do período refratário
Step 97-100: fora do período refratário
```

Conclusão: NÃO está em refratário.

O Ciclo Vicioso Completo


FASE 1: ATIVAÇÃO INICIAL
Input → Alguns neurônios disparam → STDP
↓
Pesos aumentam ligeiramente
↓
FASE 2: HOMEOSTASE INTERVÉM (muito cedo)
FR < target → Homeostase AUMENTA pesos (+2.5%)
FR < target → Homeostase ABAIXA threshold (-6.7%)
↓
Neurônios agora disparam MAIS facilmente
↓
FASE 3: RUNAWAY LTP (feedback positivo)
Pesos altos → Disparo fácil → STDP forte → Pesos ++
↓ ↑
└──────────────────────┘ (loop)
Pesos: 9.4 → 30.8 (+229% em 92 steps!)
Threshold: 0.20 → 0.16 (-22%)
↓
FASE 4: ADAPTIVE THRESHOLD CONTRAATACA
recent_firing_rate acumula (EMA com α=0.01)
↓
adaptive_threshold = base_th * (1 + 3*recent_FR)
= 0.155 * 1.24 = 0.192
↓
THRESHOLD EFETIVO SOBE acima do inicial!
↓
FASE 5: RECURSOS STP ESGOTADOS (provável)
Sinapses usadas repetidamente perdem recursos
↓
effective_weight = base_weight * stp_resources
= 2.2 * 0.5 = 1.1 (redução 50%!)
↓
Potencial calculado CAI mesmo com pesos altos
↓
FASE 6: MORTE SÚBITA

```
│  Potencial < adaptive_threshold                                  │
│  ↓                                                               │
│  Neurônios param de disparar                                     │
│  ↓                                                               │
│  Sem disparos → Sem STDP → Só weight decay                       │
│  ↓                                                               │
│  FR network → 0.000                                              │
│  ↓                                                               │
│  □ REDE MORTA                                                    │
└──────────────────────────────────────────────────────────────────┘
```

Mecanismos em Ação

1. STDP (Spike-Timing-Dependent Plasticity)

Código: dendritoma.rs:406-477

```rust
pub fn apply_stdp_pair(&mut self, pre_neuron_id: usize, delta_t: i64, reward: f64)
-> bool {
    let weight_change = if delta_t > 0 {
        // LTP (potenciação)
        final_a_plus * self.plasticity[pre_neuron_id] *
            (-delta_t as f64 / self.stdp_tau_plus).exp() *
            reward_modulation
    } else {
        // LTD (depressão)
        -final_a_minus * self.plasticity[pre_neuron_id] *
            (delta_t.abs() as f64 / self.stdp_tau_minus).exp()
    };

    self.weights[pre_neuron_id] += weight_change;

    // Decay proporcional
    let proportional_decay = self.weights[pre_neuron_id] * 0.0001;
    self.weights[pre_neuron_id] -= proportional_decay;

    // ...
}
```

Efeito observado: +0.98% a +17.8% por disparo (varia com correlação).

2. Homeostatic Plasticity

Código: nenv.rs:apply_homeostatic_plasticity (não lido completamente, inferido)

```rust
// Aproximado baseado em comportamento observado
fn apply_homeostatic_plasticity(&mut self, current_time: i64, has_input: bool) {
    if (current_time - self.last_homeo_update) < self.homeo_interval {
        return; // Só aplica a cada N steps
```

```
    }

    let rate_error = self.target_firing_rate - self.recent_firing_rate;

    // Synaptic scaling
    let scale = 1.0 + self.homeo_eta * rate_error * self.homeo_weight_ratio;
    self.dendritoma.apply_synaptic_scaling(rate_error, self.homeo_eta * 0.7);

    // Threshold adjustment
    let threshold_delta = -self.homeo_eta * rate_error *
self.homeo_threshold_ratio;
    self.threshold += threshold_delta;
    self.threshold = self.threshold.clamp(0.001, self.base_threshold * 2.0);

    self.last_homeo_update = current_time;
}
```

Efeito observado:

- Pesos: +2.48% quando FR baixo
- Threshold: -6.7% quando FR baixo

3. Adaptive Threshold (Sparse Coding)

Código: nenv.rs:206

```
let adaptive_threshold = self.threshold * (1.0 + self.recent_firing_rate * 3.0);
```

Efeito observado:

- Step 0: 0.200 * 1.0 = 0.200
- Step 92: 0.155 * 1.24 = 0.192 (SUBIU!)

PROBLEMA: Threshold **efetivo** sobe quando neurônio dispara muito, MESMO que threshold base caia.

4. Short-Term Plasticity (STP)

Código: dendritoma.rs:221-250

```
pub fn integrate(&mut self, inputs: &[f64]) -> f64 {
    let mut potential = 0.0;

    for i in 0..self.weights.len() {
        let base_weight = self.weights[i] + self.weights_ltm[i];
        let stp_modulation = self.synaptic_resources[i] *
self.stp_facilitation[i];
        let effective_weight = base_weight * stp_modulation;

        potential += inputs[i] * effective_weight;
```

```
        // Consome recursos se input ativo
        if inputs[i].abs() > 0.1 {
            self.synaptic_resources[i] *= 1.0 - self.stp_use_fraction; // 0.85
            self.stp_facilitation[i] += 0.1; // Facilitação temporária
        }
    }

    potential
}
```

Efeito: Recursos depletam com uso repetido, **reduzindo effective_weight**.

5. Weight Decay

Aplicado em 3 lugares:

1. **apply_stdp_pair** (linha 471): `decay = weight * 0.0001`
2. **apply_stdp_learning** (linha 503): `weight *= 0.9999`
3. **apply_weight_maintenance** (linha 365): `weight *= 1.0 - effective_decay`

Efeito total: ~-0.02% por step quando não há STDP.

Root Cause: O Problema Real

O problema **NÃO É**:

- ❌ Weight decay excessivo
- ❌ STDP muito fraco
- ❌ Falta de energia
- ❌ Período refratário

O problema **É**:

- ✅ **Adaptive threshold + recent_firing_rate** cria um **ceiling dinâmico**
- ✅ **Homeostase intervém muito cedo** (interval = 9), causando runaway
- ✅ **STP resources esgotam** com atividade repetida
- ✅ **Falta de mecanismo anti-runaway** em STDP

Soluções Propostas

Solução 1: Ajustar Adaptive Threshold (CRÍTICO)

Problema: Multiplicador de 3.0 é muito agressivo.

```
// ANTES (nenv.rs:206)
let adaptive_threshold = self.threshold * (1.0 + self.recent_firing_rate * 3.0);

// DEPOIS
let adaptive_threshold = self.threshold * (1.0 + self.recent_firing_rate * 1.0);
```

Efeito esperado: Threshold adaptativo sobe menos com firing rate.

Solução 2: Retardar Homeostase

Problema: Homeostase aplica muito cedo (interval = 9).

```
// ANTES (nenv.rs:165)
homeo_interval: 9,

// DEPOIS
homeo_interval: 50, // Só aplica após 50 steps
```

Efeito esperado: Rede tem tempo para estabilizar naturalmente antes de homeostase intervir.

Solução 3: Reduzir Amplitude Homeostática

Problema: Homeostase muda pesos +2.48% de uma vez.

```
// ANTES (nenv.rs:164)
homeo_eta: 0.1627,

// DEPOIS
homeo_eta: 0.05, // Correção mais suave
```

Solução 4: Soft Cap em STDP LTP

Problema: STDP pode aumentar pesos sem limite (até weight_clamp).

```
// Em apply_stdp_pair, após calcular weight_change:
if weight_change > 0.0 { // LTP
    // Soft saturation: quanto maior o peso, menor o ganho
    let saturation_factor = 1.0 - (self.weights[pre_neuron_id] /
self.weight_clamp);
    weight_change *= saturation_factor.max(0.1); // Mínimo 10% do ganho
}

self.weights[pre_neuron_id] += weight_change;
```

Solução 5: Recuperação Mais Rápida de STP

Problema: STP recovery é muito lento (τ = 150).

```
// ANTES (dendritoma.rs:139)
stp_recovery_tau: 150.0,
```

```
// DEPOIS
stp_recovery_tau: 50.0, // Recuperação 3x mais rápida
```

Experimento de Validação

Teste Proposto

1. Aplicar **Solução 1 + Solução 2** (adaptive threshold + homeo interval)
2. Rodar `deep_flow_analysis` por 1000 steps
3. Verificar:
   - ✅ Pesos estáveis (±20% da inicial)
   - ✅ FR network mantém > 0.10
   - ✅ Sem runaway (max weight < 15.0)
   - ✅ Sem morte súbita (FR em steps finais > 0.05)

Critérios de Sucesso

```
Métrica              | Antes    | Alvo      | Crítico
---------------------|----------|-----------|----------
Weight change        | +229%    | ±30%      | < 100%
FR final             | 0.000    | > 0.10    | > 0.05
Último disparo       | step 92  | > step 950| > step 900
Adaptive threshold   | 0.203    | < 0.220   | < 0.250
```

Arquivos para modificar:

- `src/nenv.rs`: Linha 206 (adaptive threshold multiplier)
- `src/nenv.rs`: Linha 165 (homeo_interval) OU
- `src/autoconfig/params.rs`: Ajustar parâmetros default

Comando de teste:

```
cargo run --release --example deep_flow_analysis
```