high_level_overview

Where each piece lives

Layer	Lives inside	Role
BICEP	“Physics + logic” sandbox	• Generates Brownian path ensemble• Feeds those paths through the **ENN cell** → collapses to the push-out symbol **p_t**
Fusion Alpha	Master signal head	• Consumes **p_t** (from BICEP/ENN)• Consumes **z_t** (graph embedding)• Consumes FinBERT sentiment & technicals• Emits directional score + size
Risk dial (Ising or limit/colimit functor)	Parallel overlay	• Reads live micro-/macro-risk gauges• Scales or vetoes the position coming out of Fusion Alpha

> So the ENN stays inside BICEP, upstream of Fusion Alpha — not inside Fusion Alpha itself.

End-to-end inference loop (every tick)

```
┌── 1. Update contradiction graph ──────────────────────┐
│   PyG MessagePassing → z_t                    │
└──────────────────────────────────────────────────────────
──────────┘
         │
         │
┌── 2. Brownian paths (Triton SDE) ─┐
│   → ENN cell w/ entangled states  │
│   → state collapse → p_t          │
└──────────────────────────────────────┘
         │
         ▼
```

```
3. Feature stack  $x_t$  =  [ $z_t$ ‖ $p_t$ ‖ FinBERT ‖ TA ]
        |
        ▼
4. Fusion Alpha forward pass  →  (dir, raw_size)
        |
        ▼
5. Risk dial  (Ising or limit/colimit)  →  leverage_mult
        |
        ▼
6. Position sizing  size = raw_size × leverage_mult
        |
        ▼
7. Live router / execution
```

Key points

- **BICEP = "generate context"** – ENN sits here, translating Brownian exploration into the minimal symbol $\mathbf{p_t}$ your theory demands.
- **Fusion Alpha = "decide direction"** – it *needs* both $\mathbf{p_t}$ and $\mathbf{z_t}$ to judge whether the market's outward story and inward crowd-state disagree.
- **Risk dial = "keep you alive"** – whether it's Ising criticality or the category-limit blend of vol / corr / liquidity, it gates the final size.

Training pipeline (offline)

1. **Pre-compute contradiction graphs** for each training window.
2. **Sample Brownian paths → ENN collapse** to generate $\mathbf{p_t}$ sequences.
3. Build training tuples (z_t , p_t , FinBERT, TA) → future returns.
4. **Train Fusion Alpha** (with dropout, attention, etc.).
5. Separately fit / tune the **risk dial** on out-of-sample periods.
6. Optionally fine-tune ENN + graph encoder jointly once Fusion Alpha stabilises.

Why keep ENN inside BICEP?

- **Separation of concerns** – BICEP handles stochastic information discovery; Fusion Alpha focuses on exploiting contradictions for alpha.
- **Latency neutrality** – ENN's 0.7 ms forward pass is amortised *before* Fusion Alpha, so the head model stays lightweight.
- **Modularity –** you can iterate on BICEP (e.g., new path kernels) without touching the trained Fusion Alpha weights.

If you ever want to experiment with an *all-in-one* super-net, you can splice an ENN layer directly into Fusion Alpha, but the current architecture keeps debugging, risk controls, and theory alignment far cleaner.

TL;DR – Ordering is:

1️⃣ Contradiction-graph encoder → 2️⃣ BICEP + ENN → 3️⃣ Fusion Alpha → 4️⃣ Risk dial → 5️⃣ Execution.