Dendritic Optimization Hackathon Report

TravelPlanner Experiments + W&B Hyperparameter Sweeps

Author: Nguyen Nguyen, Tue Tran
Platform: Google Colab
Tracking: Weights & Biases (W&B)
Sweep ID: 46sgkuvo (sweep link)
Primary metric: PR-AUC / Average Precision (AP)
Task: Imbalanced binary classification derived from TravelPlanner
Dataset: osunlp/TravelPlanner [1]

Executive summary (numbers-first)

- We compared **Baseline** vs **Dendrites** on TravelPlanner using PR-AUC (AP) and thresholded F1.
- A representative baseline-vs-dendrites evaluation (threshold chosen on validation, evaluated on test) gave: **Baseline AP=0.187 @ thr=0.869** vs **Dendrites AP=0.168 @ thr=0.479**.
- We ran a W&B sweep (ID **46sgkuvo**) over **learning rate** and **weight decay**, producing many runs with **best_val_pr_auc up to 0.41344** and **test PR-AUC up to 0.44663**.
- Key takeaway: dendrites strongly affected score calibration (optimal thresholds ranging from **0.00145** to **0.99975**), so threshold strategy + PR-AUC tracking were essential.

Figure-free report: all plots are summarized as numbers and conclusions.

Contents

1 Project goal

This project evaluated whether dendritic optimization can improve model quality and/or decision behavior on a real, noisy, imbalanced benchmark scenario derived from *TravelPlanner*. The deliverables were:

- a reproducible training/evaluation pipeline in Google Colab,
- clear baseline vs dendrites comparison using PR-AUC and thresholded metrics,
- a W&B sweep showing sensitivity to key hyperparameters,
- an engineering recap of what was built/debugged (including framework-adjacent exploration).

2 Dataset: TravelPlanner

Dataset reference

TravelPlanner is a benchmark for real-world planning with language agents [1].
Hugging Face: https://huggingface.co/datasets/osunlp/TravelPlanner

2.1 How we used it here

In this project, we converted the problem into an **imbalanced binary classification** setup and evaluated using:

- **PR-AUC / Average Precision (AP)** as the main metric (robust for rare positives),
- **F1@0.5** for a fixed operating point,
- **F1 at validation-chosen threshold** (`best_thr`) for a "deployable" decision rule.

Accuracy is misleading under class imbalance. PR-AUC is more informative when false positives and false negatives are asymmetric and positives are rare.

3 Method

3.1 Training + evaluation loop

We implemented a standard PyTorch loop:

- **Train:** `train_one_epoch(...)` logs `train/loss`
- **Validate/Test:** `evaluate(...)` logs `val/pr_auc`, `val/f1@0.5`, `test/pr_auc`, `test/f1@0.5`
- **Threshold selection:** pick `best_thr` on validation to maximize a validation criterion, then evaluate test at that threshold: `final_test_f1@best_thr`, precision, recall.

3.2 Baseline vs Dendrites

We compared:

- **Baseline**: standard modules

- **Dendrites** : dendritic optimization enabled

We observed dendrites can shift score distributions substantially, changing which threshold is optimal. Therefore we always logged `best_thr` and thresholded test metrics in addition to PR-AUC.

4 Hyperparameter sweep report (W&B-style narrative)

This section is intentionally formatted like a W&B report: what we swept, what we learned, and what it means.

4.1 Sweep overview

Sweep metadata

Sweep ID: 46sgkuvo **Project:** vtpy/dendrites-hackathon
We swept over **learning rate** and **weight decay** while tracking PR-AUC and thresholded F1 on validation and test.

4.2 Hyperparameters explored

- **Learning rate** (`lr`): sampled across the 10^{-4}–10^{-3} range (see Appendix table for exact values per run).
- **Weight decay** (`weight_decay`): sampled from near-zero ($\sim 10^{-6}$) up to a few 10^{-3} and higher.
- **Threshold selection behavior** (`best_thr`): not a tuned hyperparameter, but an *observed* quantity that varied widely (from **0.00145** to **0.99975**) and strongly affected downstream F1.

4.3 Headline sweep findings (numbers)

Best validation PR-AUC observed: **0.41344** (run tdif3p2t, `best_thr=0.75876`)
Another high validation PR-AUC: **0.40934** (run hpmq62fe, `best_thr=0.69435`)
Highest test PR-AUC observed: **0.44663** (run 3ei24bcq) and **0.44634** (run hpmq62fe)
Largest calibration extremes: `best_thr=0.00145` (gp84dgsz) to `best_thr=0.99975` (y8krklgw)

4.4 Why this matters (business framing)

Travel planning / itinerary reasoning pipelines often sit inside larger agent systems where:

- **False positives** can trigger wasted downstream tool calls (search, booking checks, constraint solvers),
- **False negatives** can miss valid itinerary options and reduce user trust.

Our results show dendrites can meaningfully change decision behavior (threshold calibration) even when headline AP changes are small. That means dendrites may be valuable in practice if tuned to a target operating point (e.g., “precision-first” vs “recall-first”).

5 Results

5.1 Baseline vs Dendrites on TravelPlanner (VAL-chosen threshold, TEST eval)

This is the clean baseline-vs-dendrites comparison you summarized from your evaluation artifact:

Setting	Test AP (PR-AUC)	Val-chosen thr	TP / FP	TN / FN
Baseline	0.187	0.869	TP=4, FP=25	TN=39, FN=4
Dendrites	0.168	0.479	TP=1, FP=8	TN=56, FN=7

At the validation-chosen operating point, dendrites reduced false positives (FP 25 → 8) but also reduced true positives (TP 4 → 1), which lowered AP in this specific head-to-head. This is consistent with dendrites shifting score distributions and emphasizing the need for calibrated threshold selection.

5.2 Additional baseline snapshot (single run)

A baseline run you logged earlier ("baseline_adamw") reported:

- **Best validation PR-AUC:** 0.15237 at epoch 8 with **best threshold:** 0.87788
- **Test PR-AUC:** 0.24924 **Test F1@0.5:** 0.20
- **Test at best threshold (strict):** precision = 0, recall = 0, F1 = 0

5.3 Sweep results (selected top-performing runs)

From the sweep logs you provided, some notable runs included:

- **Run hpmq62fe:** best_val_pr_auc=0.40934, test/pr_auc=0.44634, best_thr=0.69435, final_test_f1@best_thr=0.42857 (precision=0.50, recall=0.375).
- **Run 3ei24bcq:** test/pr_auc=0.44663, best_val_pr_auc=0.29750, best_thr=0.55035.
- **Run tdif3p2t:** best_val_pr_auc=0.41344, test/pr_auc=0.25857, best_thr=0.75876.
- **Run jeg08kfr:** test/pr_auc=0.39453 and strong test/f1@0.5=0.46154 with best_thr=0.39074.

6 Engineering recap (start-to-end)

6.1 What we built

- A Colab-friendly training pipeline with explicit **train/val/test splits** and W&B logging.
- A consistent evaluation suite reporting **PR-AUC**, **F1@0.5**, and **validation-chosen thresholds**.
- A W&B sweep configuration (ID **46sgkuvo**) exploring `lr` and `weight_decay`.

6.2 Major debugging milestones encountered

- Resolved a **regex patching** issue in Colab (`re.sub` bad escape) by avoiding template escape pitfalls.
- Diagnosed PerforatedAI package **export/import structure** in Colab:
  - `perforatedai` imported successfully but exposed *no top-level names*.
  - Located internal modules containing expected symbols (e.g., `tracker_perforatedai.py`, `modules_perforatedai.py`).
  - Observed that `initialize_pai` was present as a string hit but not importable as a public function, while classes like `PAINeuronModule` / `PAIDendriteModule` were available.

6.3 Framework-adjacent exploration: NanoGPT (attempted)

We explored integrating dendrites into NanoGPT (single-file-ish pure PyTorch) for bonus points. In Colab, the attempt reached key blockers:

- NanoGPT patching reached an **ImportError** attempting to import a `DendriticLinear`-style entry point.
- NanoGPT config override then raised **Unknown config key:** `dend_activation` when the training script did not define that key.

Given time constraints, we prioritized completing the TravelPlanner baseline+dendrites experiments and sweep analysis.

7 Limitations and next steps

7.1 Limitations

- **Threshold instability:** some runs picked extreme thresholds (near 0 or near 1), which can inflate/deflate F1 even when AP is stable.
- **Small positive counts:** confusion-matrix deltas can swing with a few examples; multiple seeds would improve confidence.
- **Matched sweep needed:** to attribute effects to dendrites vs hyperparameters, baseline and dendrites should be swept under identical search spaces and seeds.

7.2 Next steps (actionable)

1. Run **paired sweeps** (baseline vs dendrites) with identical ranges and **3–5 random seeds**; report mean±std of AP.
2. Choose a **business operating point** (e.g., "precision ≥ 0.5") and compare models at that constraint, not only at max-F1 thresholds.
3. Add **efficiency metrics** (params, wall-clock/step, memory) to quantify any compute/size wins from dendrites.

Appendix A: Full sweep run summaries (from logs shared)

How to read this table

Each row is a run whose config and summary metrics were explicitly printed in your Colab logs. All values below are copied from those run summaries: `lr`, `weight_decay`, `best_val_pr_auc`, `best_thr`, `test/pr_auc`, `test/f1@0.5`, and final test metrics at `best_thr`.

Run ID	Run name	lr	wd	best_val_AP	best_thr	test_AP	F1@0.5	F1@thr	P/R@th
tdif3p2t	pretty-sweep-1	0.0005784	0.0019845	0.41344	0.75876	0.25857	0.20000	0.31579	0.27273 / 0.37
hpmq62fe	curious-sweep-2	0.0010658	0.0002224	0.40934	0.69435	0.44634	0.15385	0.42857	0.50000 / 0.37
3ei24bcq	lemon-sweep-3	0.0001149	0.00002155	0.29750	0.55035	0.44663	0.27273	0.35294	0.33333 / 0.37
jeg08kfr	classic-sweep-4	0.0019701	0.000002321	0.29750	0.39074	0.39453	0.46154	0.46154	0.60000 / 0.37
zjy5t9dq	tough-sweep-5	0.0014207	0.0056958	0.27225	0.78951	0.30060	0.33333	0.18182	0.33333 / 0.12
48tpv2qo	frosty-sweep-6	0.0011561	0.00002006	0.26626	0.55600	0.34423	0.18182	0.18182	0.33333 / 0.12
gekumtcm	lucky-sweep-7	0.0002432	0.00007782	0.26575	0.97494	0.31560	0.33333	0.00000	0.00000 / 0.00
ppuk8dyw	stilted-sweep-8	0.0015590	0.000003121	0.26575	0.99683	0.23132	0.12500	0.22222	1.00000 / 0.12
mvju26qr	rural-sweep-9	0.0010187	0.00007526	0.26863	0.99782	0.27915	0.26667	0.18182	0.33333 / 0.12
1ukkyivo	logical-sweep-10	0.0023387	0.0001423	0.27580	0.00432	0.15659	0.33333	0.23729	0.13725 / 0.87
6rnmju46	soft-sweep-11	0.0009815	0.0019128	0.27798	0.00247	0.17696	0.23529	0.25455	0.14894 / 0.87
y8krklgw	dandy-sweep-12	0.0003044	0.000001158	0.28125	0.99975	0.18707	0.23529	0.18182	0.33333 / 0.12
gp84dgsz	splendid-sweep-13	0.0028425	0.0016237	0.28697	0.00145	0.16911	0.11111	0.20000	0.11290 / 0.87
lhmuf9y8	lively-sweep-14	0.0014925	0.0002486	0.28292	0.00557	0.17502	0.20000	0.22222	0.13043 / 0.75
nrvhpglz	different-sweep-15	0.0015612	0.0008791	0.28585	0.00163	0.17849	0.12500	0.21875	0.12500 / 0.87
u2vd34cr	genial-sweep-16	0.0003452	0.0024057	0.16135	0.00181	0.19422	0.28571	0.20690	0.12000 / 0.75
8kvnmpro	dark-sweep-17	0.0025003	0.00008332	0.19283	0.00629	0.17540	0.18750	0.17143	0.09677 / 0.75
0c280l2j	unique-sweep-18	0.0001529	0.000003187	0.19851	0.01089	0.21114	0.28571	0.27907	0.17143 / 0.75
4b183pr9	deep-sweep-19	0.0001940	0.000004501	0.19373	0.00884	0.21242	0.28571	0.32432	0.20690 / 0.75
c56jbayo	effortless-sweep-20	0.0002972	0.00004052	0.17151	0.00429	0.20766	0.28571	0.33333	0.21429 / 0.75

References

[1] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, Yu Su. *TravelPlanner: A Benchmark for Real-World Planning with Language Agents*. arXiv preprint arXiv:2402.01622, 2024.