TR-Hash Vision: Deterministic Spatial-Token Routing for a Compact Object Detector

Boris Peyriguere
Complexity-ML, Independent Research
boris@complexity-ai.fr

August 2026

Status: early research report, not a product announcement. Results are from a single random-initialization run; no hyperparameter search was performed.

Abstract

We present a 2.53M-parameter anchor-free object detector trained from random initialization on COCO 2017, whose vision backbone routes each spatial token through 2 of 8 narrow experts using a deterministic, precompiled hash of the token's grid position rather than a learned gating network. The routing table is fixed at construction time, guarantees exact per-expert load balance by construction, and compiles to a compact per-token dispatch code that a fused CUDA kernel consumes directly — removing the runtime gating computation, auxiliary load-balancing loss, and token-dropping/capacity mechanisms that typical soft-routed MoE vision models require. At 50 epochs on COCO, the pretrain checkpoint reaches 16.59 AP50-95 / 27.81 AP50 (one-to-many branch, with NMS). A subsequent full-parameter, clean-image supervised fine-tuning (SFT) stage — Mosaic-free, multi-resolution 512–640 px, stopped at epoch 15/30 on a validation plateau — raises this to **20.05 AP50-95 / 32.50 AP50**. We report both branches' NMS-free (one-to-one) counterpart, which localizes correctly but remains markedly under-calibrated in confidence relative to the O2M branch at both training budgets, and discuss why. We are looking for feedback on the routing design, the CUDA dispatch path, and the O2M/NMS-free gap.

1 Motivation

Sparse Mixture-of-Experts (MoE) layers are usually gated by a small learned network that scores each token against every expert and routes it to the top-k highest-scoring experts. This is effective but comes with recurring costs: the gating network itself adds parameters and compute; load across experts is not guaranteed and typically needs an auxiliary balancing loss; and at low batch sizes or small spatial resolutions (as in a vision backbone operating on a few hundred tokens per image, not tens of thousands as in language models) empirical load can be highly uneven, causing some experts to be under-trained.

TR-Hash removes the learned gate entirely. For a vision backbone, each spatial position in a feature map is assigned a fixed "vocabulary" ID (its position in the stage's grid), and a deterministic hash of that ID selects a fixed top-k set of experts for every token at that position, for the entire training run. Because the assignment is content-independent and known at model-construction time, we can:

- guarantee exact combinatorial balance (every expert is used exactly once per rank-block of `num_experts` positions, for the `balanced_hash` strategy used here);
- precompile the route table into a compact per-position dispatch code (a 5-bit pair index plus one orientation bit for top-2 routing over 8 experts) that a fused CUDA kernel reads directly, with no runtime top-k or softmax computation;
- drop the auxiliary load-balancing loss term entirely, since balance is structural rather than learned.

The tradeoff is that routing cannot adapt to token content or image semantics — only to spatial position within a stage's grid. Whether this constraint costs meaningful accuracy relative to content-aware gating is one of the open questions we would like feedback on.

2 Architecture

2.1 Vision tower

A 7-layer hierarchical tower over stages of depth $(2, 2, 3)$, hidden size 128, 4 attention heads. Each block is:

1. **Windowed self-attention** (window size 8, Swin-style) with a learned relative position bias, over the block's local window.
2. **TR-Hash MoE FFN**: a SwiGLU feed-forward split into a dense shared branch (width 216) applied to every token, and 8 narrow experts (width 27 each),

of which every token activates its assigned top-2, selected by the deterministic hash of its spatial-grid position for that stage.

The shared branch dominates parameter count and is always active; the expert branches contribute a comparatively small residual specialization. Absolute row/column position embeddings (interpolated per-stage for variable input resolution) are added to the feature map before each stage, in addition to the attention block's relative position bias.

A P2-equivalent (stride-8) fine detection level is enabled, giving the detection head access to higher-resolution features for small objects.

2.2 Neck and heads

A PAN-style neck (2 repeats, normalized per-channel fusion weights for adjacent-scale fusion) feeds two prediction heads sharing the same backbone features:

- **One-to-many (O2M) head**: the primary head, trained with a top-k=8 dynamic label assignment per ground-truth object (TAL/SimOTA-style), decoded at inference with confidence thresholding + NMS.
- **One-to-one (O2O) head**: a lightweight auxiliary head (57K parameters, 2.3% of the model) trained with Hungarian one-to-one matching, intended to produce NMS-free predictions directly (in the spirit of YOLOv10/DETR dual-assignment training). Its loss weight ramps linearly from 0.25 to 1.0 over training, its parameters get a 1.5× LR multiplier relative to the rest of the head, and only 25% of its gradient is propagated back into the shared backbone features (`one_to_one_shared_gradient_scale=0.25`), so as not to let the auxiliary objective distort the features the primary head depends on.

Total model size: **2,533,369 parameters** (2,475,941 without the one-to-one head).

Both heads use distribution focal loss (DFL) for box regression, a quality/IoU-aware classification loss, and an independently learned positive logit scale per pyramid level for the regression distribution.

3 Training recipe

3.1 Pretraining (from random initialization)

- **Dataset**: COCO 2017 `train2017` (118,287 images), validated on `val2017`, 80 classes, from random initialization (no ImageNet or COCO-pretrained backbone).
- **Resolution**: 640×640, patch size 8.
- **Optimizer**: MuSGD (momentum 0.947, muon weight 0.528, sgd weight 0.674), backbone LR multiplier 1.0, expert LR multiplier 1.5, warmup 3 epochs, base LR 5.4e-3 decayed to a 0.04952 floor ratio, weight decay 6.4e-4.
- **Batch**: 16 images/GPU on a single GPU, gradient-accumulated to a nominal batch of 64.
- **Augmentation**: Mosaic (16 source images per canvas, composited at 2× resolution then trained at the model's native 640px — augmentation strength increases at no extra compute cost, since the canvas is downsampled back down), plus MixUp and multi-scale training. Exact coefficients are in the public training script.[1]
- **Schedule**: 50 epochs, Mosaic disabled for the final 10 epochs (`close_mosaic_epochs`) to give the model a clean, full-resolution fine-tuning phase at low LR before completion — this produced the single largest jump in validation AP of the entire run (§4).
- **Step accounting**: `packed_epochs=2` halves the number of optimizer steps per epoch relative to a full pass over the dataset, independent of Mosaic. This is a deliberate divisor on total training compute, not an artifact of augmentation — we explicitly do not credit a downscaled Mosaic-composited tile as equivalent to a full-resolution un-packed step (§6).

3.2 Supervised fine-tuning (from the pretrain checkpoint)

A second stage takes the pretrain checkpoint and refines all parameters under clean-image conditions:

- **Initialization**: full-parameter transfer from the pretrain checkpoint (§4).
- **Augmentation**: Mosaic and MixUp disabled; light augmentation only, multi-resolution training in $[512, 640]$px.
- **Optimizer**: MuSGD, same family as pretraining, restarted LR schedule at a lower peak.
- **EMA**: 0.9999.
- **Checkpoint selection**: official COCO mAP50-95, selected independently for the O2M and NMS-free branches (their best epochs differ by one).
- **Schedule**: configured for 30 epochs; stopped during epoch 15 after the validation curve plateaued (last logged step 27,020).

[1] https://github.com/Complexity-ML/complexity-framework

The purpose of this stage is to let the detector consolidate on real, undistorted images after a pretraining run whose augmentation policy (Mosaic-composited canvases, MixUp) intentionally trades realism for effective dataset coverage under a from-scratch, no-pretrained-backbone budget.

4 Results

4.1 Pretraining

COCO `val2017`, 50 epochs, single random-init run:

Branch	AP50-95	AP50	AR100
O2M + NMS	16.59	27.81	34.35
One-to-one (NMS-free)	8.06	12.08	—

O2M size breakdown: AP75 17.10, AP-S 9.47, AP-M 17.96, AP-L 24.29. One-to-one: AP75 8.80, AP-S 5.77, AP-M 10.93, AP-L 12.14.

Progression over the last three eval points (O2M+NMS AP50-95): 13.67 (epoch 44) → 16.59 (epoch 49/50), i.e. still improving at a faster rate than earlier in training once Mosaic was disabled for the final 10 epochs, suggesting the run was not close to convergence at 50 epochs.

4.2 Supervised fine-tuning

The SFT stage confirms that prediction directly: refining the same 2.53M-parameter model on clean, augmentation-light images pushes both branches meaningfully higher, with the O2M branch gaining faster than the NMS-free branch.

Branch	Epoch	AP50-95	AP50	AR100
O2M + NMS	12	**20.05**	**32.50**	37.85
NMS-free	13	9.62	14.01	38.79

Relative to the pretrain checkpoint, O2M+NMS AP50-95 improves 16.59 → 20.05 (+3.46, a 20.9% relative gain) and NMS-free AP50-95 improves 8.06 → 9.62 (+1.56, a 19.4% relative gain) — comparable relative gains on both branches, so SFT narrows neither closes nor widens the O2M/NMS-free gap in relative terms, even though the absolute gap widens slightly (8.53 → 10.43 AP50-95 points).

SFT size breakdown: O2M AP-S 10.96, AP-M 20.74, AP-L 30.30. NMS-free: AP-S 7.05, AP-M 11.97, AP-L 15.28. Both branches' best-F1 operating points shift to a higher decision confidence than at pretraining (O2M: 0.199; NMS-free: 0.141), consistent with a better-calibrated model overall, though the NMS-free branch's best confidence remains well below the O2M branch's, as discussed in §5.

Training was stopped early (epoch 15 of a configured 30) once the validation curve plateaued; the best checkpoints for each branch were selected independently by official COCO mAP50-95 (O2M: epoch 12; NMS-free: epoch 13), one epoch apart, indicating the two branches converge on a similar timescale under SFT despite their different absolute performance.

For reference, Ultralytics YOLOv8n (3.2M params, comparable scale) reports 37.3 AP50-95 on the same benchmark — but trained for 500 epochs, roughly 10× our combined pretrain+SFT compute budget, and (per Ultralytics' own YOLO26 training-recipe documentation) their more recent releases are initialized from intermediate pretrained weights rather than a single from-scratch run, which we do not have visibility into for a fully controlled comparison. We view our number as a compute-matched checkpoint on a trajectory, not a final result — training compute, not architecture, is the current bottleneck to closing this gap, based on the epoch-over-epoch trend at both stages.

5 The O2M / NMS-free gap

The one-to-one branch localizes correctly — qualitatively, its predicted boxes on held-out images coincide closely with the O2M branch's boxes on the same objects — but its confidence scores are systematically low, at both the pretrain and SFT checkpoints. On a representative street scene (pretrain checkpoint), the O2M branch's top detections had confidences of 0.25–0.78; the one-to-one branch's best detections on the *same* objects topped out at 0.20, below the default 0.25 decision threshold, i.e. it would report zero detections without manually lowering the threshold. Lowering the threshold recovers matching boxes plus a small number of extra false positives not present in the O2M output. SFT raises the NMS-free branch's best-F1 confidence to 0.141 and its AP50-95 to 9.62, a real gain, but the gap to O2M widens in absolute terms (§4.2) rather than closing — weak evidence against the underexposure explanation dominating on its own, since the same additional full-parameter training that helps the O2M branch does not proportionally narrow the gap.

We see two plausible, non-exclusive explanations, and would appreciate feedback on which is more likely to dominate, or whether we are missing something:

1. **Underexposure**: the one-to-one loss weight ramps from 0.25 to 1.0 over the pretraining run and only a quarter of its gradient reaches the shared backbone — by design, to protect the primary head — but this may leave the auxiliary head with too few effective full-weight gradient steps to reach confidence calibration comparable to the primary head, even with its higher (1.5×) LR multiplier and the additional SFT epochs.

2. **Assignment mismatch under a hash-routed backbone**: Hungarian one-to-one matching assumes the backbone can produce sufficiently discrimina-

tive per-position features to break ties between near-duplicate candidate boxes. If the position-only hash routing (§2.1) caps how much the backbone can specialize spatially beyond what the fixed route table already encodes, one-to-one assignment may be a harder training signal to fit than one-to-many assignment, independent of training budget — consistent with SFT narrowing the gap only in relative, not absolute, terms.

6 A step-accounting pitfall we corrected mid-project

Our first pass at this experiment silently under-trained the model relative to its parameter count. The cause: our packed-epoch schedule computed the number of optimizer steps per epoch by crediting a Mosaic-composited canvas with a step-count "discount" proportional to how many source images it packed in (e.g., a 4-image Mosaic tile counted for a quarter step). This conflates two different things — *informational exposure* (how many source images the model has been shown, even if downscaled) and *optimizer-step budget* (how many real gradient updates it received) — and it silently shrank the total number of optimizer steps well below what the un-augmented dataset size would imply for the requested epoch count. We decoupled the two: `packed_epochs` is now a plain, independent divisor on the natural (un-packed) step count, and Mosaic tile count is purely an augmentation-strength knob with no effect on step budget. `packed_epochs=1` now reproduces the natural, un-packed step count exactly.

We mention this because it is an easy trap for anyone combining Mosaic-style augmentation with an epoch-budget or packed-epoch mechanism, and because it is directly responsible for why the pretraining run in this report trains for meaningfully more optimizer steps than our own earlier (uncorrected) baseline at the same nominal epoch count and Mosaic configuration.

7 Limitations

- **Single run, no seed variance.** All numbers above are from one random-initialization pretraining run and one SFT run from that checkpoint; we have not measured seed-to-seed variance at either stage.
- **Compute-matched, not accuracy-matched, comparison to YOLO baselines.** The combined pretrain+SFT budget here is far short of typical from-scratch YOLO training schedules (300–500 epochs); the gap in §4 should be read as a compute-budget gap, not evidence about the routing design's ceiling.
- **`end-to-end` was not actually toggleable in our launch scripts until the pretrain report.** A shell-script/argparse default mismatch meant the one-to-one head was trained in every run regardless of the script's `END_TO_END` setting (`argparse.BooleanOptionalAction` defaults to `True`, and our launcher only ever explicitly *enabled* the flag, never explicitly disabled it). This does not affect the O2M numbers reported here, and the reported 2.53M parameter count already includes the one-to-one head, but it does mean we have never actually run a clean `end_to_end=False` ablation to isolate whether the auxiliary head's (partial) shared-gradient flow helps or hurts the primary branch. We fixed the script; that ablation is future work.
- **SFT stopped early on a plateau, not a fixed schedule.** The SFT run was configured for 30 epochs and stopped at epoch 15 once validation stopped improving; we have not verified whether a longer, re-warmed schedule or a different LR floor would extend the gains in §4.2.

8 Future work

- A controlled `end_to_end=False` ablation to isolate the one-to-one head's effect on the primary branch.
- Longer combined pretrain+SFT budgets to establish whether the gap to compute-mismatched YOLO baselines closes with additional compute alone, or whether architectural changes are needed.
- A targeted intervention on the NMS-free branch (e.g. a higher shared-gradient scale or an earlier full loss weight) to test the underexposure hypothesis in §5 directly, now that the SFT stage did not close the absolute gap on its own.
- CUDA kernel profiling for the fused dispatch path relative to a standard soft-routed MoE baseline at equivalent parameter count, to quantify the runtime-gating removal's actual wall-clock benefit (not yet measured — we have the mechanism but no head-to-head timing).

Model checkpoints: AETHORIA-AI/TR-HASH-Vision-v8-2M-COCO (pretrain) and AETHORIA-AI/TR-HASH-Vision-v8-2M-COCO-SFT (SFT). Feedback on routing design, CUDA efficiency, and the O2M/NMS-free gap is especially welcome.