GTR: GATED TOKEN RECURRENCE FOR EFFICIENT DENSE PREDICTION

Anonymous authors
Paper under double-blind review

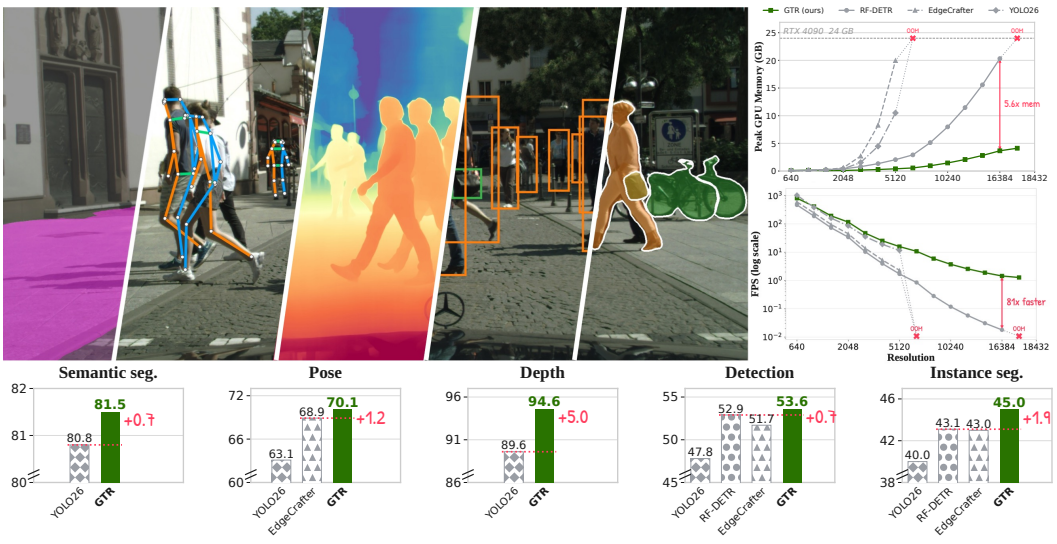

Figure 1: **GTR: a shared backbone design for efficient dense prediction.** *Left:* Task-specific models built on the same GTR backbone design produce predictions across semantic segmentation, human pose estimation, monocular depth estimation, object detection, and instance segmentation. *Right:* Peak GPU memory and inference throughput on RTX 4090 as input resolution increases, comparing GTR-S with RF-DETR-S (Robinson et al., 2026), EdgeCrafter-S (Liu et al., 2026), and YOLO26-S (Jocher et al., 2026). GTR-S shows slower peak-memory growth and higher throughput at increasing resolutions.

ABSTRACT

Self-attention-based vision backbones perform well on dense prediction, but the quadratic computational cost of global softmax attention limits their efficiency as image resolution increases. We introduce Gated Token Recurrence (GTR), a softmax-free recurrent vision backbone that combines gated linear attention, alternating spatial scan directions, and spatially enhanced SwiGLU blocks. GTR is distilled from a detection-specialized DINOv3 teacher using only final-layer patch-token alignment through a linear projection and squared ℓ_2 loss, without masked-token prediction or intermediate-layer supervision. With Objects365 detector pretraining, GTR-L achieves 58.9 box AP on COCO `val2017` with 1.908 ms median batch-one latency under compiled FP16 execution on an RTX 4090. The same backbone also transfers to instance segmentation, pose estimation, oriented detection, semantic segmentation, and monocular depth estimation. In an isolated kernel benchmark, our specialized chunkwise CUDA operator is $4.0\times$ faster than FLA v0.5.0 at 1.6K tokens on RTX 4090. TensorRT deployment on DRIVE AGX Thor achieves 2.282–8.769 ms median batch-one latency across the evaluated models. These results show that recurrent token mixing can provide an efficient alternative to global softmax attention for high-resolution dense prediction and edge deployment. Code and pretrained models will be released upon publication.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Self-attention-based vision backbones have shown strong performance for dense prediction (Liu et al., 2026). However, the quadratic complexity of softmax attention with respect to the number of image tokens makes high-resolution processing increasingly expensive, particularly on resource-constrained edge devices. Linear attention offers a promising alternative: EfficientViT (Cai et al., 2023) demonstrates efficient on-device semantic segmentation, while ViG (Liao et al., 2025) explores gated linear recurrence for detection and segmentation. Despite this progress, developing a general linear-attention backbone that performs consistently across diverse dense prediction tasks while meeting practical constraints on model size, computation, memory, and latency remains challenging. This motivates a central question: can linear attention support a general-purpose vision backbone that remains accurate across diverse dense prediction tasks while translating its theoretical efficiency into practical gains on edge hardware?

We introduce *Gated Token Recurrence (GTR)*, a linear-complexity (in the number of image tokens) vision backbone designed to address this challenge. GTR builds on gated linear attention (GLA) (Yang et al., 2024b) with a softmax-free, single-scale recurrent architecture. It maintains a fixed patch grid across 12 blocks and performs one directional scan per block, cycling through four directions across depth to progressively aggregate global context. To complement this directional recurrence with explicit local spatial modeling, we introduce *Spatial SwiGLU*, which incorporates depthwise convolution into the value branch of SwiGLU (Shazeer, 2020) to efficiently mix neighboring patches. Together, directional recurrence and Spatial SwiGLU provide complementary global and local spatial modeling while preserving linear complexity.

Importantly, GTR uses a single distillation objective that supervises only the final patch tokens. Unlike ViT-Linearizer (Wei & Chellappa, 2025), which combines intermediate activation matching with masked prediction, or ViT-AdaLA (Li et al., 2026b), which aligns attention modules before the final features, our approach distills only the final patch tokens. Specifically, GTR is aligned with a frozen, detection-specialized DINOv3 teacher (Siméoni et al., 2025) using the same full-image input. A linear projection and a single squared ℓ_2 loss constitute the entire distillation objective, requiring neither masking nor intermediate supervision.

GTR scales across four model sizes (S, M, L, and X) and transfers to six dense prediction tasks: object detection, instance segmentation, human pose estimation, oriented object detection, semantic segmentation, and monocular depth estimation. With Objects365 detector pre-training (Shao et al., 2019), GTR-L achieves 58.9 box AP on COCO (Lin et al., 2014) `val2017` at a median forward-pass latency of only 1.908 ms under compiled FP16, batch-one inference on an RTX 4090. Beyond detection, GTR achieves competitive results in representative comparisons across the other five tasks using the same backbone design with task-specific heads and training recipes (Section 3.2; Appendix D). As summarized in Figure 1, GTR-S also exhibits slower growth in peak GPU memory and higher inference throughput than the softmax-attention-based EdgeCrafter-S (Liu et al., 2026) as input resolution increases, highlighting the practical advantage of linear-complexity attention for high-resolution dense prediction.

Finally, we bridge architectural efficiency and real-world deployment with dedicated system optimizations. We develop a specialized chunkwise CUDA operator for GLA (Yang et al., 2024b) that computes chunk summaries in parallel while propagating recurrent context through a boundary-state scan. The operator achieves a $4.0\times$ speedup over FLA v0.5.0 (Yang & Zhang, 2024) at 1.6K tokens under FP16 and CUDA Graph execution on an RTX 4090. For end-to-end deployment on DRIVE AGX Thor, we further integrate fused kernels and graph transformations into TensorRT (NVIDIA, 2026). Across six tasks and four model scales, GTR achieves median FP16 batch-one latencies of 2.282–8.769 ms, demonstrating its practicality for real-time dense prediction on edge hardware.

In summary, our main contributions are:

- We introduce *GTR*, a single-scale, linear-attention vision backbone for efficient dense prediction. GTR combines gated token recurrence with four-directional scanning across depth and Spatial SwiGLU for complementary global and local spatial modeling.
- We demonstrate the *generality and scalability* of GTR across six dense prediction tasks and four model scales. The same backbone design transfers effectively across object-level and pixel-level tasks using simple final-token distillation and task-specific adaptation.

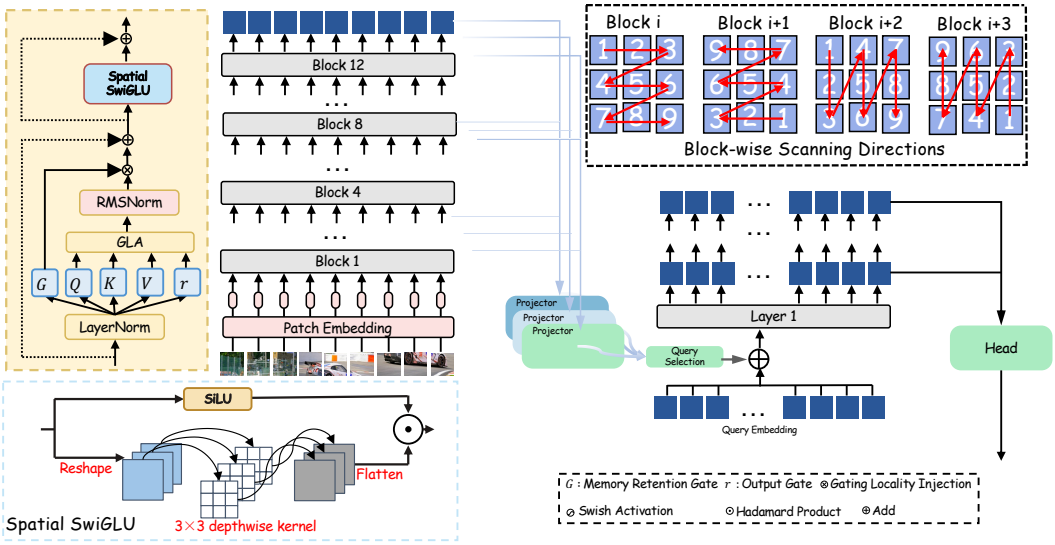

Figure 2: **Overview of GTR.** Each backbone block combines GLA (Yang et al., 2024b) with Spatial SwiGLU, whose 3×3 depthwise convolution mixes neighboring patch tokens in the value branch. The 12-block backbone alternates two-dimensional scan directions and exposes intermediate features to a lightweight three-scale projector. A query-based head decodes the resulting feature pyramid.

- We enable *practical edge deployment* through a specialized chunkwise CUDA operator and TensorRT integration. Across six tasks and four model scales, GTR achieves low-latency FP16 inference on DRIVE AGX Thor, demonstrating low-latency deployment on automotive edge hardware.

Code and pretrained models will be released upon publication.

2 METHOD

Figure 2 provides an overview of GTR and its object detection pipeline. We first describe the GTR backbone, followed by its distillation strategy and specialized chunkwise inference. Given an image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$, GTR produces a stride-16 grid of $L = HW/16^2$ patch tokens and maintains the same resolution throughout 12 blocks. Each block combines gated linear attention (GLA) (Yang et al., 2024b) for long-range context with Spatial SwiGLU for local spatial mixing. The scan direction cycles through four directions across successive blocks to aggregate context from different orientations. For object detection, features from blocks 4, 8, and 12 are used to build a lightweight feature pyramid at strides 8, 16, and 32, followed by a DETR-style decoder (Carion et al., 2020; Peng et al., 2025). The GTR backbone itself remains single-scale, while multi-scale features are constructed in the detection neck. Details of the neck and decoder are provided in Appendix B, and the four GTR model configurations in Appendix D.1.1.

2.1 GTR BACKBONE

GTR combines three components: gated recurrence for long-range context, directional scanning for two-dimensional context, and Spatial SwiGLU for local spatial mixing.

Key-only gated recurrence. We adopt the key-only variant of GLA (Yang et al., 2024b). For one attention head, let $\mathbf{X} \in \mathbb{R}^{L\times d}$ denote the input tokens. We first compute $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \mathbf{K} = \mathbf{X}\mathbf{W}_K, \mathbf{V} = \mathbf{X}\mathbf{W}_V$, where $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d\times d_k}$ and $\mathbf{W}_V \in \mathbb{R}^{d\times d_v}$. Let $\boldsymbol{x}_t \in \mathbb{R}^{1\times d}$ be the input row at token t, and let $\boldsymbol{q}_t, \boldsymbol{k}_t$, and $\boldsymbol{v}_t$ denote the corresponding projected row vectors.

GLA maintains a recurrent state $\mathbf{S}_t \in \mathbb{R}^{d_k\times d_v}$ that summarizes the tokens encountered along the current scan, with $\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\gamma}_t)\mathbf{S}_{t-1} + \boldsymbol{k}_t^\top \boldsymbol{v}_t$, $\boldsymbol{o}_t = d_k^{-1/2}\boldsymbol{q}_t\mathbf{S}_t$, and $\mathbf{S}_0 = \mathbf{0}$. Here, $\boldsymbol{\gamma}_t =$

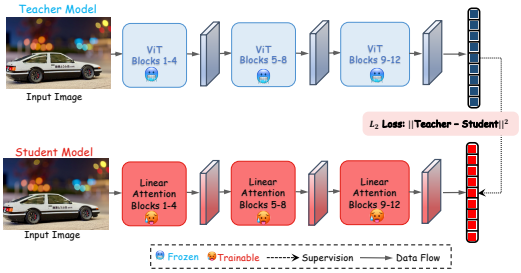

Figure 3: **Final-output representation alignment.** The frozen detection-specialized DINOv3 teacher (Siméoni et al., 2025) and GTR student process the same full image. We align their final patch features using a squared ℓ_2 loss after a learned linear projection of the student features (Equation 2). The projection is omitted from the figure for clarity.

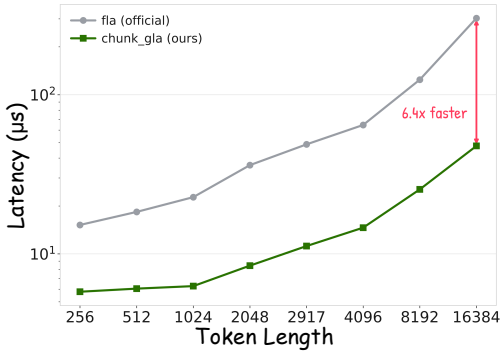

Figure 4: **GLA operator latency on RTX 4090.** We compare our inference-only operator with FLA v0.5.0 (Yang & Zhang, 2024) using FP16 inputs and CUDA Graph execution. A 640×640 input corresponds to 1.6K stride-16 patch tokens. Details are provided in Appendix E.3.

$\sigma(\boldsymbol{x}_t \mathbf{W}_\alpha^{(1)} \mathbf{W}_\alpha^{(2)} + \boldsymbol{b}_\alpha)^{1/\tau}$ is a learned key-wise retention gate with $\boldsymbol{\gamma}_t \in (0,1)^{1\times d_k}$. Following GLA, the gate uses a low-rank projection with $\mathbf{W}_\alpha^{(1)} \in \mathbb{R}^{d\times r}$, $\mathbf{W}_\alpha^{(2)} \in \mathbb{R}^{r\times d_k}$, $\boldsymbol{b}_\alpha \in \mathbb{R}^{1\times d_k}$, rank $r = 16$, and temperature $\tau = 16$. The sigmoid σ and the power are applied elementwise. Each gate entry scales one row of $\mathbf{S}_{t-1}$ across all d_v value channels.

GTR uses only this key-side state decay, with the value-side decay fixed to one during both training and inference. A separate data-dependent output gate is applied to the RMS-normalized output (Zhang & Sennrich, 2019), where $\boldsymbol{r}_t = \mathrm{SiLU}(\boldsymbol{x}_t \mathbf{W}_r)$ and $\boldsymbol{u}_t = \boldsymbol{r}_t \odot \mathrm{RMSNorm}(\boldsymbol{o}_t)$. For multi-head GLA, the outputs of all heads are concatenated and projected back to the model dimension. Note that the output gate $\boldsymbol{r}_t$ is distinct from the state-retention gate $\boldsymbol{\gamma}_t$.

Directional scanning. Since the recurrence is causal along its scan order, a single scan captures context from only one direction. To model two-dimensional context, GTR cycles through four scan directions across successive blocks (see Figure 2): left-to-right, right-to-left, top-to-bottom, and bottom-to-top. Each block performs only one directional scan and restores the original spatial order afterward. Across depth, the alternating directions allow tokens to progressively receive complementary spatial context.

Spatial SwiGLU. Directional recurrence provides long-range context, but local interactions remain dependent on the scan order. We therefore introduce Spatial SwiGLU to provide explicit local spatial mixing:

$$\mathbf{Y} = \left[\mathrm{SiLU}(\mathbf{X}\mathbf{W}_g) \odot \mathrm{DWConv}_{3\times 3}(\mathbf{X}\mathbf{W}_v)\right]\mathbf{W}_o. \tag{1}$$

The value branch is reshaped to the two-dimensional patch grid, processed by a 3×3 depthwise convolution, and then restored to sequence order. This provides non-causal local interaction among neighboring patches, while the gating branch remains tokenwise. **Note that GTR uses no explicit positional embeddings.**

Complexity discussion. For fixed model width and head count, GTR scales linearly with the number of tokens L. The recurrent update costs $\mathcal{O}(Ld_kd_v)$ per head, while projections, Spatial SwiGLU, and other channel operations are also linear in L. The recurrent state itself has constant size with respect to L, while memory for token activations and intermediate workspaces grows linearly with sequence length.

2.2 Distillation

Figure 3 illustrates our simple final-feature distillation pipeline. We use a DINOv3 backbone (Siméoni et al., 2025), adapted for detection following EdgeCrafter (Liu et al., 2026), as the frozen teacher. The teacher and GTR receive the same full image, yielding spatially corre-

sponding patch tokens. We supervise only the final patch-token representations, without masking or intermediate-layer matching.

Let $\mathbf{Y}_{\text{tea}} \in \mathbb{R}^{L\times d_t}$ and $\mathbf{Y}_{\text{stu}} \in \mathbb{R}^{L\times d_s}$ denote the final teacher and student representations at the L patch positions. Since their feature dimensions may differ, we apply a learned linear projection $\phi : \mathbb{R}^{d_s} \to \mathbb{R}^{d_t}$ to the student features. The distillation objective for one image is

$$\mathcal{L}_{\text{align}} = \frac{1}{L}\sum_{i=1}^{L} \left\|\phi(\mathbf{Y}^i_{\text{stu}}) - \mathbf{Y}^i_{\text{tea}}\right\|_2^2. \tag{2}$$

The loss is averaged over patch positions and excludes non-patch tokens. Only GTR and the linear projection are optimized. This simple objective encourages the projected student features to match the teacher's final representations while allowing GTR to learn a different internal computation. Unlike methods that use masking, intermediate supervision, or attention alignment, our distillation relies solely on final patch-token regression. Appendix D.1.2 provides the teacher assignments, training images, and optimization details. The resulting GTR weights serve as backbone initialization for subsequent detector pre-training and task-specific training.

2.3 Hardware-Aware Chunkwise Inference

The chunkwise formulation of GLA (Yang et al., 2024b) rewrites the token-wise recurrence as efficient matrix operations within fixed-size chunks. Our implementation follows a simple three-stage procedure: we first compute chunk summaries in parallel, then propagate context across chunk boundaries through a lightweight recurrent scan, and finally compute the outputs of all chunks independently. Our inference-only operator is specialized for the GTR configuration: FP16 inputs, chunk size $C = 64$, per-head dimensions $d_k = 32$ and $d_v = 64$, and a zero initial state. Training continues to use the reference FLA kernels (Yang & Zhang, 2024). The operator takes the same queries, keys, values, and learned log-decays $\boldsymbol{g}_t = \log \boldsymbol{\gamma}_t$ as the original recurrence.

For chunk c with valid length $C_c \le C$, let $\boldsymbol{A}_{c,r} = \sum_{i=0}^{r} \boldsymbol{g}_{c,i}$ denote the cumulative log-decay. We summarize the contribution of each chunk as

$$\begin{aligned} \widehat{\boldsymbol{k}}_{c,r} &= \boldsymbol{k}_{c,r} \odot \exp(\boldsymbol{A}_{c,C_c-1} - \boldsymbol{A}_{c,r}), & \mathbf{U}_c &= \sum_{r=0}^{C_c-1} \widehat{\boldsymbol{k}}_{c,r}^{\top} \boldsymbol{v}_{c,r}, \\ \boldsymbol{d}_c &= \exp(\boldsymbol{A}_{c,C_c-1}), & \mathbf{S}_{[c+1]} &= \mathrm{Diag}(\boldsymbol{d}_c)\mathbf{S}_{[c]} + \mathbf{U}_c. \end{aligned} \tag{3}$$

The expensive chunk summaries $\mathbf{U}_c$ are computed in parallel across chunks and heads. Only the boundary states $\mathbf{S}_{[c]}$ are propagated sequentially, making the remaining recurrent scan lightweight. This changes only the execution schedule, not the underlying GLA recurrence.

Once the boundary states are available, all chunk outputs can again be computed independently. Defining $\widetilde{\boldsymbol{q}}_{c,r} = \boldsymbol{q}_{c,r} \odot \exp(\boldsymbol{A}_{c,r})$ and $\widetilde{\boldsymbol{k}}_{c,r} = \boldsymbol{k}_{c,r} \odot \exp(-\boldsymbol{A}_{c,r})$, the output at position r is $\boldsymbol{o}_{c,r} = d_k^{-1/2}\left(\widetilde{\boldsymbol{q}}_{c,r}\mathbf{S}_{[c]} + \sum_{j=0}^{r} \widetilde{\boldsymbol{q}}_{c,r}\widetilde{\boldsymbol{k}}_{c,j}^{\top}\boldsymbol{v}_{c,j}\right)$. Our output kernel fuses the within-chunk computation with the boundary-state readout. Additional implementation details, including mixed-precision accumulation, length-aware dispatch, and optional output-gate fusion, are provided in Appendix E.3.1.

As shown in Figure 4, our operator is $2.6\times$, $4.0\times$, and $6.4\times$ faster than FLA v0.5.0 (Yang & Zhang, 2024) at sequence lengths of 256, 1.6K, and 16.4K tokens, respectively, under FP16 and CUDA Graph execution on RTX 4090. This isolated benchmark excludes output-epilogue fusion. Full-model latency on RTX 4090 and TensorRT deployment on DRIVE AGX Thor are evaluated separately in Sections 3.1 and 3.3.

3 Experiments

In this section, we evaluate GTR across diverse dense prediction tasks and analyze its key design choices. We first evaluate GTR on object detection, and then transfer the backbone to five additional tasks: instance segmentation, human pose estimation, oriented object detection, semantic segmentation, and monocular depth estimation. Each task uses a task-specific head, initialization, and

Table 1: **COCO (Lin et al., 2014) detection performance and RTX 4090 latency.** We compare GTR with three recent baselines; full results and COCO metrics are provided in Appendix C.1. GTR uses 640×640 inputs, while RF-DETR uses its native scale-specific resolutions. Latency is measured with compiled FP16 inference at batch size one (Appendix E.2). ‡ indicates models without Objects365 (Shao et al., 2019) pre-training. Per scale group, best and second-best median latency and AP are highlighted.

Model	#Epochs	#Params (M)	GFLOPs	Latency (ms) ↓			COCO AP ↑					
				Min.	Mean	Median	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
RF-DETR-S (Robinson et al., 2026)	–	32	60	1.207	1.285	1.299	52.9	71.9	57.0	32.0	58.3	73.0
YOLO26-S (Jocher et al., 2026)	70	10	21	0.922	0.977	0.962	47.8	64.6	52.1	29.1	52.5	64.3
ECDet-S‡ (Liu et al., 2026)	74	10	26	1.511	1.641	1.652	51.7	69.4	55.8	32.3	56.4	70.5
GTR-S (ours)‡	30	12.1	33.8	1.215	1.225	1.225	50.7	68.5	54.7	31.1	55.5	70.2
GTR-S (ours)	30	12.1	33.8	1.215	1.225	1.225	53.6	71.1	58.3	36.4	58.3	70.2
RF-DETR-M (Robinson et al., 2026)	–	34	79	1.326	1.428	1.431	54.7	73.5	59.2	36.1	59.7	73.8
YOLO26-M (Jocher et al., 2026)	80	20	68	1.459	1.481	1.473	52.5	69.8	57.2	36.2	56.9	68.5
ECDet-M‡ (Liu et al., 2026)	62	19	53	1.901	2.080	2.095	54.3	72.2	58.7	35.9	59.1	72.7
GTR-M (ours)‡	30	22.7	62.6	1.450	1.462	1.462	54.0	72.2	58.5	35.3	59.1	73.1
GTR-M (ours)	30	22.7	62.6	1.450	1.462	1.462	57.3	75.0	62.4	41.1	62.0	74.1
RF-DETR-L (Robinson et al., 2026)	–	34	126	1.805	1.952	1.968	56.5	75.1	61.3	39.0	61.0	73.9
YOLO26-L (Jocher et al., 2026)	60	25	86	1.953	1.969	1.967	54.3	71.5	59.4	37.8	58.6	70.3
ECDet-L‡ (Liu et al., 2026)	50	32	101	2.493	2.713	2.730	57.0	75.1	61.7	38.7	62.5	75.0
GTR-L (ours)‡	30	37.2	106	1.902	1.908	1.908	55.5	73.7	60.3	36.8	61.1	74.8
GTR-L (ours)	30	37.2	106	1.902	1.908	1.908	58.9	76.8	64.3	42.4	64.0	76.0
RF-DETR-X (Robinson et al., 2026)	–	126	300	2.971	3.227	3.243	58.6	77.4	63.8	40.3	63.9	76.2
YOLO26-X (Jocher et al., 2026)	40	55	194	3.069	3.088	3.087	56.9	74.1	62.1	41.3	61.2	72.7
ECDet-X‡ (Liu et al., 2026)	50	49	151	2.950	3.025	3.021	57.9	76.0	62.9	38.7	63.4	76.1
GTR-X (ours)‡	30	46.5	130.2	2.104	2.117	2.115	56.2	74.5	61.1	37.3	61.8	74.9
GTR-X (ours)	30	46.5	130.2	2.104	2.117	2.115	59.4	77.3	64.7	42.1	64.6	76.4

training recipe, as detailed in Appendix D. The main paper focuses on recent competitive methods, with more comprehensive comparisons provided in Appendix C. We conclude with ablations of the main architectural and training choices.

Training details and inference latency. After backbone distillation, the main GTR detectors are trained for 36 epochs on Objects365 (Shao et al., 2019), followed by 30 epochs on COCO (Lin et al., 2014) `train2017`. Direct-COCO variants skip the Objects365 stage. We evaluate on COCO `val2017` at 640×640 resolution. For latency, we benchmark all models on an RTX 4090 using FP16, batch size one, `torch.compile`, and CUDA Graphs, and report median CUDA-event latency after warm-up. We measure the network forward pass for DETR-style models and additionally include CUDA NMS for methods that require separate post-processing. Baseline accuracy is taken from the corresponding sources, while latency is measured under our setup. Full training and latency protocols are provided in Appendices D and E.2, respectively.

3.1 Real-Time Object Detection

Table 1 compares GTR with three recent baselines: YOLO26 (Jocher et al., 2026), RF-DETR (Robinson et al., 2026), and ECDet (Liu et al., 2026). With Objects365 pre-training (Shao et al., 2019), GTR-S/M/L/X achieve 53.6/57.3/58.9/59.4 AP, respectively, obtaining the best AP across all four scales. GTR-L reaches 58.9 AP at 1.908 ms median latency, outperforming all L-scale baselines in both AP and latency. GTR-X achieves 59.4 AP at 2.115 ms, compared with 58.6 AP at 3.243 ms for RF-DETR-X. Without Objects365 pre-training, GTR consistently reduces latency relative to ECDet across all four scales. Full results are provided in Appendix C.1.

3.2 Transfer to Other Dense-Prediction Tasks

We train separate task-specific models using the same GTR backbone design. Table 2 summarizes representative comparisons at the S and X scales, while Appendix C provides results across all scales and metrics.

Instance segmentation and human pose estimation. On COCO (Lin et al., 2014), GTR transfers effectively to both instance segmentation and human pose estimation. For instance segmentation, GTR-S/X achieve 45.0/49.8 mask AP without SAM2-generated pseudo masks (Ravi et al., 2025). GTR-X reaches 49.8 mask AP with 25.3% lower latency than RF-DETR-Seg-X (Robinson et al.,

Table 2: **Transfer of the GTR backbone to five additional dense prediction tasks at the S and X scales.** Each task uses a separately trained model with a task-specific head (Appendix D). Params in M, FLOPs in GFLOPs, Lat. is the median RTX 4090 latency (ms) measured as in Table 1; complete comparisons are provided in Appendix C. † denotes Objects365 (Shao et al., 2019) pre-training with SAM2 pseudo masks (Ravi et al., 2025) for instance segmentation and additional Objects365 supervision for pose estimation. Per task and scale, the best latency and accuracy are highlighted.

Task	Model	S				X			
		Params	FLOPs	Lat. ↓	Metric	Params	FLOPs	Lat. ↓	Metric
Instance Seg. COCO mask AP ↑	YOLO26-Seg† (Jocher et al., 2026)	10.4	34.2	1.163	40.0	62.8	313.5	4.139	47.0
	RF-DETR-Seg† (Robinson et al., 2026)	33.7	70.6	1.445	43.1	38.1	260.0	3.311	48.8
	ECInsSeg (Liu et al., 2026)	10.3	33.1	1.755	43.0	49.9	168.1	3.140	48.4
	GTR (ours)	12.6	46.8	1.465	45.0	47.4	151.6	2.472	49.8
Pose COCO keypoint AP ↑	YOLO26-Pose (Jocher et al., 2026)	10.4	23.9	0.813	63.1	57.6	201.7	2.788	71.6
	DETRPose† (Janampa & Pattichis, 2026)	11.5	33.1	1.218	67.0	73.3	239.5	4.145	73.3
	ECPose (Liu et al., 2026)	9.9	30.4	1.519	68.9	50.6	172.2	3.281	74.8
	GTR (ours)	11.9	37.0	1.455	70.1	47.6	142.9	2.590	75.5
Oriented Det. DOTA AP_{50} ↑	YOLO26-obb (Jocher et al., 2026)	9.8	56.7	1.099	80.9	57.6	520.1	5.085	81.7
	GTR (ours)	12.1	82.8	1.946	80.1	46.3	324	3.960	81.3
Semantic Seg. Cityscapes mIoU ↑	YOLO26-sem (Jocher et al., 2026)	6.5	88.8	0.758	80.8	40.2	861.7	4.325	83.6
	GTR (ours)	6.9	82.8	1.495	81.5	32.2	317.1	3.482	83.6
Depth NYU δ_1 ↑	YOLO26-depth (Jocher et al., 2026)	13.2	67.9	0.850	0.896	57.0	302.0	2.730	0.933
	GTR (ours)	11.5	65.0	1.296	0.946	41.1	159.4	2.158	0.954

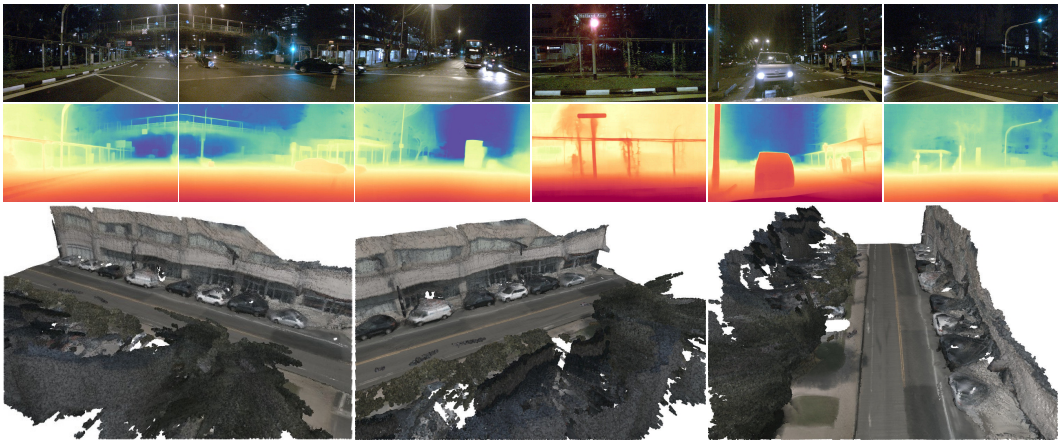

Figure 5: **GTR-L depth prediction and reconstruction on nuScenes (Caesar et al., 2020), without dataset-specific fine-tuning.** *Top/middle:* Six nighttime surround-view images and the corresponding GTR-L depth predictions. *Bottom:* Multi-view visualization of a reconstructed street scene, obtained by temporally fusing predicted depth after LiDAR-assisted per-frame scale alignment (Appendix F.5.2).

2026). For pose estimation, GTR achieves 70.1/75.5 keypoint AP, improving over ECPose (Liu et al., 2026) at both S and X scales while also reducing latency. Full results are provided in Appendices C.2 and C.3.

Oriented detection and semantic segmentation. GTR also transfers effectively to oriented detection and semantic segmentation. On DOTA-v1.0 (Xia et al., 2018), GTR-X achieves 81.3 AP_{50} with lower median latency than YOLO26x-obb (Jocher et al., 2026) (3.960 vs. 5.085 ms). On Cityscapes (Cordts et al., 2016), GTR-S/X achieve 81.5/83.6 mIoU with a lightweight FCN head (Long et al., 2015). Compared with YOLO26-sem, GTR achieves higher mIoU at smaller scales, while GTR-X matches 83.6 mIoU with 19.5% lower latency. Full comparisons across all scales are provided in Appendices C.4 and C.5.

Monocular depth estimation. We further evaluate GTR on monocular depth estimation using an eight-source mixture of measured and pseudo-labeled depth, without NYU-specific fine-tuning. On

Table 3: **Cross-architecture distillation on GTR-S without Objects365 pre-training.** We include a from-scratch baseline for reference. Methods use the same teacher, student, and COCO training schedule.

Initialization / recipe	Distill. stages	#Distill losses	COCO AP ↑					
			AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
Random initialization (from scratch)	0	0	31.1	44.9	33.3	17.0	33.1	43.7
ViT-AdaLA (adapted) (Li et al., 2026b)	2	1	48.2	65.6	52.1	28.2	52.4	66.9
ViT-Linearizer (adapted) (Wei & Chellappa, 2025)	1	2	50.0	67.7	53.6	**31.1**	54.7	69.4
Final-output ℓ_2 (ours)	**1**	**1**	**50.7**	**68.5**	**54.7**	**31.1**	**55.5**	**70.2**

Table 4: **Ablation of spatial modeling in GTR.** We study local spatial mixing with S-SwiGLU on GTR-S and scan directions on GTR-L. LR/RL/TB/BT denote left-to-right, right-to-left, top-to-bottom, and bottom-to-top scans.

Setting	Variant	COCO AP ↑					
		AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
Local spatial mixing (GTR-S)							
768^2	Learned positional embedding	53.9	71.6	58.6	38.0	57.8	70.2
768^2	**S-SwiGLU (ours)**	**55.1**	**72.5**	**60.2**	**38.7**	**59.6**	**70.7**
1024^2	Learned positional embedding	55.3	72.7	60.3	40.6	58.7	**70.7**
1024^2	**S-SwiGLU (ours)**	**56.6**	**73.9**	**61.9**	**42.1**	**60.8**	70.4
Scan direction (GTR-L)							
Horizontal	LR/RL + S-SwiGLU	55.1	73.4	59.8	36.1	60.5	**75.2**
Vertical	TB/BT + S-SwiGLU	55.0	73.3	59.8	35.8	60.7	74.6
Single direction	S-SwiGLU	54.4	72.6	59.0	35.1	59.7	73.9
Single direction	Learned positional embedding	45.6	62.3	49.1	27.5	49.1	63.5
Four directions	**LR/RL/TB/BT + S-SwiGLU**	**55.5**	**73.7**	**60.3**	**36.8**	**61.1**	74.8

NYU Depth V2 (Silberman et al., 2012; Eigen et al., 2014), GTR improves δ_1, AbsRel, and RMSE over the evaluated YOLO26 depth variants (Jocher et al., 2026) at both scales. As shown in Figure 5, GTR also produces qualitatively consistent depth predictions across diverse scenes. These aligned-depth results use multi-view test-time augmentation and per-image log-affine alignment to ground-truth depth. Full results and evaluation details are provided in Appendices C.6 and D.1.7.

3.3 Ablation Studies and Discussion

We ablate the main design choices of GTR, including the distillation objective, spatial modeling, token mixer, input resolution, and detector pre-training. All detection results are evaluated on COCO (Lin et al., 2014) `val2017`. For architectural ablations, each variant is independently distilled using the same teacher and image pool, followed by the same downstream training schedule. Full settings and results are provided in Appendix C.7.

Distillation recipe. Table 3 compares our final-output alignment with ViT-Linearizer (Wei & Chellappa, 2025) and ViT-AdaLA (Li et al., 2026b) under the same teacher, GTR-S student, and COCO training schedule. For ViT-Linearizer, we use the best mask ratio (0.5) from a sweep over 0.2–0.9. Despite its simple single-stage design and objective, our method achieves 50.7 AP, outperforming ViT-Linearizer (50.0) and ViT-AdaLA (48.2), while training from scratch reaches only 31.1 AP. This suggests that simple final-output alignment is effective for distilling GTR.

Spatial modeling. Table 4 shows that replacing learned positional embeddings with S-SwiGLU improves GTR-S by 1.2/1.3 AP at $768^2/1024^2$, with only 0.2 (47.6 to 47.8) /0.5 (83.2 to 83.7) additional GFLOPs. Scan direction provides a complementary benefit: on GTR-L, four-direction scanning achieves 55.5 AP, compared with 55.1/55.0 for horizontal/vertical bidirectional scanning and 54.4 for a single direction. Moreover, under single-direction scanning, replacing S-SwiGLU with learned positional embeddings further reduces AP to 45.6. Together, these results highlight the importance of local spatial mixing, with multi-directional scanning providing additional gains.

Table 5: **GLA versus hybrid GLA–softmax attention on GTR-L.** Each hybrid replaces GLA with softmax attention in three of the twelve backbone blocks.

Global-mixer layout	Latency (ms) ↓	COCO AP ↑					
		AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
Pure GLA (ours)	**1.908**	55.5	73.7	60.3	36.8	61.1	74.8
Softmax in last 3 blocks	2.117	**55.6**	73.8	**60.7**	37.2	**61.7**	74.6
Softmax in 3 uniformly spaced blocks	2.109	**55.6**	**74.0**	60.0	**37.3**	60.9	**75.1**
Softmax in first 3 blocks	2.094	55.4	73.6	60.1	35.5	61.0	74.6

Table 6: **Resolution scaling of GTR-S.** Higher resolution consistently improves accuracy, particularly for small objects, at increased computational cost.

Resolution	GFLOPs	Memory (GB)	Latency (ms) ↓	AP ↑	AP_S ↑	AP_M ↑	AP_L ↑
640^2	33.8	0.096	1.225	53.6	36.4	58.3	70.2
704^2	40.4	0.121	1.317	54.3	37.5	59.0	70.2
768^2	47.8	0.138	1.421	55.1	38.7	59.6	70.7
832^2	55.8	0.142	1.550	55.5	39.8	60.0	**70.8**
896^2	64.5	0.156	1.714	56.0	41.4	60.0	70.5
960^2	73.7	0.172	1.784	56.2	40.7	60.4	70.0
1024^2	83.7	0.171	1.844	56.6	42.1	60.8	70.4
1152^2	105.4	0.224	2.157	56.7	41.9	60.8	70.3
1280^2	129.8	0.249	2.529	**57.0**	**42.4**	**61.0**	70.0

GLA versus hybrid attention. Table 5 examines whether adding softmax attention benefits GTR. Replacing GLA with softmax attention in three of the twelve backbone blocks yields similar accuracy (55.4–55.6 vs. 55.5 AP), but increases median latency from 1.908 to 2.094–2.117 ms and computation from 106 to 119 GFLOPs. All variants have the same 37.2M parameters and 0.19 GB peak memory. Thus, softmax attention provides no meaningful accuracy gain while introducing additional computation and latency, supporting our homogeneous GLA design.

Resolution scaling. GTR-S scales effectively with input resolution (Table 6). Increasing the resolution from 640^2 to 1280^2 improves AP from 53.6 to 57.0, with a larger gain of 6.0 AP_S on small objects (36.4 to 42.4). Meanwhile, median latency increases from 1.225 to 2.529 ms and peak memory from 0.096 to 0.249 GB. Notably, latency grows by only $2.06\times$ despite a $4\times$ increase in image tokens, demonstrating favorable scaling to high-resolution inputs.

Edge deployment on DRIVE AGX Thor High-performance linear-attention kernels are typically designed for LLMs on server GPUs using Python/Triton stacks (Qwen Team, 2026; Team et al., 2025), whereas automotive deployment requires efficient execution within a production inference engine. We deploy GTR on NVIDIA DRIVE AGX Thor with TensorRT (NVIDIA, 2026) through graph–kernel co-optimization: standard TensorRT operators are combined with fused CUDA kernels for GLA and S-SwiGLU to reduce kernel launches and global-memory traffic without changing the computation. This enables low-latency inference across all six dense-prediction tasks, while maintaining at least 0.9989 cosine similarity to the PyTorch FP32 reference across all 48 task–scale–batch configurations. Appendix E.1 provides platform details, Roofline analysis (Williams et al., 2009), and complete latency results.

4 Conclusion

We presented GTR, an efficient softmax-free backbone for dense prediction. GTR combines a homogeneous GLA architecture with simple final-output distillation and optimized inference execution. Across object detection and five additional dense-prediction tasks, GTR achieves a strong accuracy–efficiency trade-off, while deployment on NVIDIA DRIVE AGX Thor demonstrates its practicality on automotive hardware. These results show that linear attention can serve as an effective and efficient alternative to softmax attention for dense visual prediction.

REPRODUCIBILITY STATEMENT

Appendix B describes the detection architecture; Appendix D gives training and evaluation settings; Appendix C provides full results and ablations; and Appendix E specifies the latency protocols and deployment implementation. Code will be released upon publication.

AI USE STATEMENT

Generative AI tools were used to assist with literature review, manuscript organization, language editing, interpretation and synthesis of the reported experimental results, and limited code development and debugging. Their use in result interpretation was limited to summarizing and describing trends and comparisons already present in the figures and tables. Generative AI tools were not used to generate, modify, or fabricate experimental data. All AI-assisted text and code were reviewed and verified by the authors, who take full responsibility for the final manuscript.

REFERENCES

Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. Arkitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. In *NeurIPS*, 2021. URL https://arxiv.org/pdf/2111.08897.pdf.

Aviv Bick, Kevin Y. Li, Eric P. Xing, J. Zico Kolter, and Albert Gu. Transformers to ssms: Distilling quadratic knowledge to subquadratic models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0999.

Aviv Bick, Tobias Katsch, Nimit Sohoni, Arjun Desai, and Albert Gu. Llamba: Scaling distilled recurrent models for efficient language processing. In *SCOPE Workshop at ICLR*, 2025.

Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2, 2020.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11618–11628, 2020. doi: 10.1109/CVPR42600.2020.01164.

Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. EfficientViT: Lightweight multi-scale attention for high-resolution dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17302–17313, 2023.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pp. 213–229, 2020. doi: 10.1007/978-3-030-58452-8_13.

Qiang Chen, Xiaokang Chen, Jian Wang, Shan Zhang, Kun Yao, Haocheng Feng, Junyu Han, Errui Ding, Gang Zeng, and Jingdong Wang. Group detr: Fast detr training with group-wise one-to-many assignment. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023.

Qiang Chen, Xiangbo Su, Xinyu Zhang, Jian Wang, Jiahui Chen, Yunpeng Shen, Chuchu Han, Ziliang Chen, Weixiang Xu, Fanrong Li, Shan Zhang, Kun Yao, Errui Ding, Gang Zhang, and Jingdong Wang. LW-DETR: A transformer replacement to YOLO for real-time detection. *arXiv preprint arXiv:2406.03459*, 2024.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3223, 2016.

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In John Fujii (ed.), *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*, pp. 303–312. ACM, 1996. doi: 10.1145/237170.237269. URL https://doi.org/10.1145/237170.237269.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Zeyu Ding, Yong Zhou, Jiaqi Zhao, Wen-Liang Du, Xixi Li, Rui Yao, and Abdulmotaleb El Saddik. Real-time oriented object detection transformer in remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 2026. doi: 10.1109/TGRS.2026.3671683.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.11929.

David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Zhe Feng, Sen Lian, Changwei Wang, Muyang Zhang, Tianlong Tan, Rongtao Xu, Weiliang Meng, and Xiaopeng Zhang. Laplacianformer: Rethinking linear attention with laplacian kernel. In *ICLR*, 2026.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling*, 2024.

Ali Hatamizadeh and Jan Kautz. MambaVision: A hybrid mamba–transformer vision backbone. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 25261–25270, 2025.

Zhangchi Hu, Yifan Zhao, Yansong Peng, Wenzhang Sun, Xiangchen Yin, Jie Chen, Peixi Wu, Hebei Li, Xinghao Wang, Dongsheng Jiang, et al. Rio-detr: Detr for real-time oriented object detection. In *European Conference on Computer Vision*, 2026.

Shihua Huang, Yongjie Hou, Longfei Liu, Xuanlong Yu, and Xi Shen. Real-time object detection meets dinov3. *arXiv*, 2025a.

Shihua Huang, Zhichao Lu, Xiaodong Cun, Yongjun Yu, Xiao Zhou, and Xi Shen. DEIM: DETR with improved matching for fast convergence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025b.

Sebastian Janampa and Marios Pattichis. DETRPose: Real-time end-to-end transformer model for multi-person pose estimation. In *European Conference on Computer Vision*, 2026.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 406–413, 2014. doi: 10.1109/CVPR.2014.59.

Glenn Jocher. YOLOv8. https://docs.ultralytics.com/models/yolov8/, 2023.

Glenn Jocher. YOLO11. https://docs.ultralytics.com/models/yolo11/, 2024.

Glenn Jocher, Jing Qiu, Mengyu Liu, Shuai Lyu, Fatih Cagatay Akyon, and Muhammet Esat Kalfaoglu. Ultralytics YOLO26: Unified real-time end-to-end vision models, 2026. URL https://arxiv.org/abs/2606.03748.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5156–5165, 2020.

Hakjin Lee, MinKi Song, Jamyoung Koo, and Junghoon Seo. Hausdorff distance matching with adaptive query denoising for rotated detection transformer. In *Proceedings of the Winter Conference on Applications of Computer Vision*, pp. 1872–1882, 2025.

Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising. In *CVPR*, 2022.

Ruibang Li, Guan Luo, Yiwei Zhang, Jin Gao, Bing Li, and Weiming Hu. SoLA-Vision: Fine-grained layer-wise linear softmax hybrid attention. *arXiv preprint arXiv:2601.11164*, 2026a.

Yifan Li, Seunghyun Yoon, Viet Dac Lai, Franck Dernoncourt, Jason Kuen, Yu Kong, and Trung Bui. ViT-AdaLA: Adapting vision transformers with linear attention. *arXiv preprint arXiv:2603.16063*, 2026b. URL https://arxiv.org/abs/2603.16063.

Bencheng Liao, Xinggang Wang, Lianghui Zhu, Qian Zhang, and Chang Huang. ViG: Linear-complexity visual sequence learning with gated linear attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 5182–5190, 2025. doi: 10.1609/aaai.v39i5.32550. URL https://ojs.aaai.org/index.php/AAAI/article/view/32550.

Zijun Liao, Yian Zhao, Xin Shan, Yu Yan, Chang Liu, Lei Lu, Xiangyang Ji, and Jie Chen. RT-DETRv4: Painlessly furthering real-time object detection with vision foundation models. In *European Conference on Computer Vision*, 2026.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 141261–141285, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e4cd50120b6d7e8daff1749d6bbaa889-Paper-Conference.pdf.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pp. 740–755, 2014. URL https://arxiv.org/abs/1405.0312.

Longfei Liu, Yongjie Hou, Yang Li, Qirui Wang, Youyang Sha, Yongjun Yu, Yinzhi Wang, Peizhe Ru, Xuanlong Yu, and Xi Shen. EdgeCrafter: Compact ViTs for edge dense prediction via task-specialized distillation. *Transactions on Machine Learning Research*, 2026. URL https://arxiv.org/abs/2603.18739v4. arXiv version 4.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015. doi: 10.1109/CVPR.2015.7298965. URL https://openaccess.thecvf.com/content_cvpr_2015/html/Long_Fully_Convolutional_Networks_2015_CVPR_paper.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Peng Lu, Tao Jiang, Yining Li, Xiangtai Li, Kai Chen, and Wenming Yang. RTMO: Towards high-performance one-stage real-time multi-person pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Wenyu Lv, Yian Zhao, Qinyao Chang, Kui Huang, Guanzhong Wang, and Yi Liu. RT-DETRv2: Improved baseline with bag-of-freebies for real-time detection transformer. *arXiv preprint*, 2024.

Chengqi Lyu, Wenwei Zhang, Haian Huang, Yue Zhou, Yudong Wang, Yanyi Liu, Shilong Zhang, and Kai Chen. RTMDet: An empirical study of designing real-time object detectors. *arXiv preprint arXiv:2212.07784*, 2022.

Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence. In *ICCV*, 2021.

Abhinav Moudgil, Ningyuan Huang, Eeshan Gunesh Dhekane, Pau Rodríguez, Luca Zappella, and Federico Danieli. Attention to mamba: A recipe for cross-architecture distillation. *arXiv preprint arXiv:2604.14191*, 2026. URL https://arxiv.org/abs/2604.14191.

NVIDIA. NVIDIA TensorRT SDK. https://developer.nvidia.com/tensorrt, 2026. Accessed: 2026-09-08.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP*, pp. 14048–14077, 2023a.

Yansong Peng, Hebei Li, Peixi Wu, Yueyi Zhang, Xiaoyan Sun, and Feng Wu. D-fine: Redefine regression task of detrs as fine-grained distribution refinement. In *International Conference on Learning Representations*, 2025. Spotlight.

Zhiliang Peng, Li Dong, Hangbo Bao, Furu Wei, and Qixiang Ye. A unified view of masked image modeling. *Transactions on Machine Learning Research*, 2023b. ISSN 2835-8856. URL https://openreview.net/forum?id=wmGlMhaBe0.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12179–12188, 2021.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations*, 2025.

Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10912–10922, 2021.

Isaac Robinson, Peter Robicheaux, Matvei Popov, Deva Ramanan, and Neehar Peri. RF-DETR: Neural architecture search for real-time detection transformers. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5bed8703db85ab27dc32f6a42f8fbdb6-Abstract-Conference.html.

Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8430–8439, 2019. URL https://openaccess.thecvf.com/content_ICCV_2019/html/Shao_Objects365_A_Large-Scale_High-Quality_Dataset_for_Object_Detection_ICCV_2019_paper.html.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *Computer Vision – ECCV 2012*, volume 7576 of *Lecture Notes in Computer Science*, pp. 746–760, 2012.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025. URL https://arxiv.org/abs/2508.10104.

Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 567–576, 2015.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Yunjie Tian, Qixiang Ye, and David Doermann. YOLOv12: Attention-centric real-time object detectors. In *Advances in Neural Information Processing Systems*, 2025.

Fabio Tosi, Luca Bartolomei, Matteo Poggi, and Stefano Mattoccia. ZipDepth: Bringing lightweight zero-shot monocular depth anywhere, on any device. In *European Conference on Computer Vision (ECCV)*, 2026. URL https://arxiv.org/abs/2607.08771.

Igor Vasiljevic, Nick Kolkin, Shanyi Zhang, Ruotian Luo, Haochen Wang, Falcon Z. Dai, Andrea F. Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R. Walter, and Gregory Shakhnarovich. DIODE: A dense indoor and outdoor depth dataset, 2019.

Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. YOLOv10: Real-time end-to-end object detection. In *Advances in Neural Information Processing Systems*, 2024a.

Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. YOLOv9: Learning what you want to learn using programmable gradient information. In *European Conference on Computer Vision*, 2024b.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models. In *Advances in Neural Information Processing Systems*, volume 37, 2024c. doi: 10.52202/079017-1996.

Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. TartanAir: A dataset to push the limits of visual SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4909–4916, 2020.

Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based detector. In *AAAI*, 2022.

Guoyizhe Wei and Rama Chellappa. ViT-Linearizer: Distilling quadratic knowledge into linear-time vision models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20737–20747, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Wei_ViT-Linearizer_Distilling_Quadratic_Knowledge_into_Linear-Time_Vision_Models_ICCV_2025_paper.html.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, 4 2009. ISSN 0001-0782. doi: 10.1145/1498765.1498785. URL https://doi.org/10.1145/1498765.1498785.

Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3974–3983, 2018.

Jie Yang, Ailing Zeng, Shilong Liu, Feng Li, Ruimao Zhang, and Lei Zhang. Explicit box detection unifies end-to-end multi-person pose estimation. In *International Conference on Learning Representations*, 2023.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. In *Advances in Neural Information Processing Systems*, volume 37, pp. 21875–21911, 2024a.

Songlin Yang and Yu Zhang. FLA: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL https://github.com/fla-org/flash-linear-attention.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56501–56523, 2024b. URL https://proceedings.mlr.press/v235/yang24ab.html.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, volume 37, 2024c.

Xue Yang, Xiaojiang Yang, Jirui Yang, Qi Ming, Wentao Wang, Qi Tian, and Junchi Yan. Learning high-precision bounding box for rotated object detection via kullback–leibler divergence. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18381–18394, 2021.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Zeyang Zhao, Qilong Xue, Yuhang He, Yifan Bai, Xing Wei, and Yihong Gong. Projecting points to axes: Oriented object detection via point-axis representation. In *European Conference on Computer Vision*, pp. 161–179, 2024.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.

GTR: Gated Token Recurrence for Efficient Dense Prediction

Appendix

A Related Work A1

B GTR Architecture Details A1

B.1 Multi-scale Feature Encoder A1

B.2 Query-based Detection Decoder A2

C Complete Experimental Results A2

C.1 Full COCO Object Detection Comparison A2

C.2 Full COCO Instance Segmentation Comparison A4

C.3 Full COCO Human Pose Estimation Comparison A4

C.4 Full DOTA-v1.0 Oriented Object Detection Comparison A5

C.5 Full Cityscapes Semantic Segmentation Comparison A6

C.6 Full NYU Depth Estimation Comparison A7

C.7 Detailed Ablation Results A7

D Experimental Details A8

D.1 Training Configurations A8

E Efficiency and Reproducibility Details A15

E.1 DRIVE AGX Thor Deployment A15

E.2 GPU Latency Measurement Protocol A16

E.3 Custom CUDA Operator A18

F Qualitative Results A19

F.1 Object Detection Results A19

F.2 Instance Segmentation Results A19

F.3 Human Pose Estimation Results A19

F.4 Semantic Segmentation Results A19

F.5 Zero-Shot Monocular Depth Estimation Results A22

F.6 Oriented Object Detection Results A23

A Related Work

Object detection with Transformers. DETR (Carion et al., 2020) formulates object detection as set prediction with Transformers, eliminating the need for separate non-maximum suppression (NMS). To address the slow convergence and high computational cost of the original DETR, a series of follow-up works has been proposed (Zhu et al., 2021; Li et al., 2022; Wang et al., 2022; Meng et al., 2021; Chen et al., 2023). RT-DETR (Lv et al., 2024) introduces a hybrid encoder for real-time object detection. Building upon this framework, D-FINE (Peng et al., 2025) further enhances detection performance through fine-grained distribution refinement and self-distillation. Beyond architectural improvements, several studies have explored incorporating Vision Transformers (ViTs) (Dosovitskiy et al., 2021) into the DETR framework. LW-DETR (Chen et al., 2024) adopts a self-supervised pre-trained ViT as the backbone, while RF-DETR (Robinson et al., 2026) combines neural architecture search with the visual foundation model DINOv2 (Oquab et al., 2024) to strengthen feature representation. More recently, EdgeCrafter (Liu et al., 2026) improves compact ViT backbones via task-specific distillation from DINOv3 (Siméoni et al., 2025).

Linear-complexity attention. Linear attention factorizes kernelized attention and exploits associativity to avoid materializing the dense token-to-token matrix, reducing complexity from $O(L^2)$ to $O(L)$ and admitting recurrent evaluation (Katharopoulos et al., 2020). Modern linear-time recurrent families, including RetNet (Sun et al., 2023), RWKV (Peng et al., 2023a), Mamba (Gu & Dao, 2024), DeltaNet (Yang et al., 2024c), and GLA (Yang et al., 2024b), enrich the recurrent state with retention, selectivity, delta-rule updates, or data-dependent gates while retaining parallel or chunkwise training forms. ViG (Liao et al., 2025) is the closest architectural precedent: it adapts GLA to vision with within-block bidirectional modeling, direction-wise gating, and local two-dimensional interaction, and provides non-hierarchical and hierarchical variants. Its fused bidirectional kernel also targets practical efficiency. GTR uses one scan per block, alternates directions across depth, and introduces local mixing in the SwiGLU channel mixer. We evaluate these choices together with a detection-specialized distillation recipe and task-specific deployment.

Hybrid approaches such as MambaVision (Hatamizadeh & Kautz, 2025) and SoLA-Vision (Li et al., 2026a) combine recurrent or linear mixers with softmax attention. Our hybrid ablation tests three softmax-layer placements within GTR-L. Kernel-based vision alternatives include LaplacianFormer's low-rank Laplacian attention (Feng et al., 2026). GTR builds on the established GLA recurrence and chunkwise formulation (Yang et al., 2024b) and contributes a fused, hardware-aware execution schedule for low-latency inference.

Heterogeneous representation distillation. Cross-architecture transfer often uses several stages or complementary targets to bridge different token mixers. ViT-Linearizer combines intermediate token-affinity matching at multiple depths with Smooth-ℓ_1 prediction of final teacher features at masked positions (Wei & Chellappa, 2025). ViT-AdaLA (Li et al., 2026b) first matches the output of every softmax- and linear-attention module with MSE, then aligns the final representations of the complete models with MSE, and finally performs supervised downstream fine-tuning. Attention to Mamba (Moudgil et al., 2026) first learns a Hedgehog feature map through per-layer cosine output matching, uses it to initialize an adapted Mamba model, and then fine-tunes the model with next-token cross-entropy. Related Transformer-to-SSM studies likewise use staged conversion or additional architecture-specific objectives (Bick et al., 2024; 2025; Wang et al., 2024c). In the detection-specialized DINOv3-to-GLA setting studied here, we use a single squared ℓ_2 loss on the final output representations, without intermediate or stage-specific distillation targets and without input masking.

B GTR Architecture Details

B.1 Multi-scale Feature Encoder

Given an input image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$, the GLA backbone extracts features from three depths:

$$\{\mathbf{X}_4, \mathbf{X}_8, \mathbf{X}_{12}\} = \mathrm{GTRBackbone}(\mathbf{I}), \tag{4}$$

where $\mathbf{X}_4, \mathbf{X}_8, \mathbf{X}_{12} \in \mathbb{R}^{\frac{H}{16}\times\frac{W}{16}\times C}$ are the restored and reshaped feature maps from the 4th, 8th, and 12th backbone blocks. For each target stride $s \in \{8, 16, 32\}$, we first align all selected features to

the same spatial resolution:

$$\widetilde{\mathbf{X}}_l^{(s)} = \mathcal{R}_s(\mathbf{X}_l), \quad l \in \{4, 8, 12\}, \tag{5}$$

where $\mathcal{R}_s(\cdot)$ resizes each feature map to the target stride $s \in \{8, 16, 32\}$ using bilinear interpolation; no resampling is applied when the source and target strides are identical. For each target scale, the aligned features from different semantic depths are concatenated along the channel dimension and fused by a lightweight C2f module (Jocher, 2023):

$$\mathbf{P}^{(s)} = \mathrm{C2f}_s\left(\mathrm{Concat}[\widetilde{\mathbf{X}}_4^{(s)}, \widetilde{\mathbf{X}}_8^{(s)}, \widetilde{\mathbf{X}}_{12}^{(s)}]\right), \quad s \in \{8, 16, 32\}. \tag{6}$$

This produces a three-level feature pyramid $\{\mathbf{P}^{(8)}, \mathbf{P}^{(16)}, \mathbf{P}^{(32)}\}$ with unified channel dimensions. Compared with heavier feature-pyramid encoders, this design keeps the encoder compact while aggregating early, middle, and late backbone representations at every detection scale. The three pyramid levels are then flattened and concatenated as the multi-scale memory for the decoder.

B.2 Query-based Detection Decoder

The decoder follows the DETR-style set prediction paradigm (Carion et al., 2020; Peng et al., 2025). Given the multi-scale features $\{\mathbf{P}^{(8)}, \mathbf{P}^{(16)}, \mathbf{P}^{(32)}\}$, we flatten and concatenate them into memory tokens:

$$\mathbf{M} = \mathrm{Concat}[\mathrm{Flatten}(\mathbf{P}^{(8)}), \mathrm{Flatten}(\mathbf{P}^{(16)}), \mathrm{Flatten}(\mathbf{P}^{(32)})]. \tag{7}$$

Initial anchor proposals are generated from the multi-scale memory, and the top-k candidates with the highest classification responses are selected to initialize object queries.

During training, we adopt grouped queries (Chen et al., 2023) by dividing detection queries into $G = 3$ groups with shared decoder parameters. This provides denser supervision and improves query diversity, while only the primary group is used during inference.

C Complete Experimental Results

The main paper uses representative detector families or model scales to keep the quantitative presentation concise. This section reports the complete comparisons used in our analysis for COCO object detection, instance segmentation, and human pose estimation, DOTA-v1.0 oriented object detection, Cityscapes semantic segmentation, and ground-truth-aligned monocular depth on NYU Depth V2.

C.1 Full COCO Object Detection Comparison

Accuracy–latency comparison. Table 7 expands the main-paper comparison with a broader set of CNN- and Transformer-based detectors, model complexity, all three latency statistics, and scale-specific AP. With Objects365 pre-training, GTR-S/M/L/X attain the highest overall AP in all four scale groups, reaching 53.6/57.3/58.9/59.4 AP. GTR-L and GTR-X also provide the lowest median latency in their groups. Without Objects365 pre-training, GTR-S/M/L/X achieve 50.7/54.0/55.5/56.2 AP under the same scale-specific COCO configurations.

Table 7: **Comparison with real-time object detectors on COCO (Lin et al., 2014) `val2017`, grouped by model scale.** Detailed latency settings are provided in Appendix E.2. For comparison methods, all non-latency entries are taken from EdgeCrafter (Liu et al., 2026), while latency is measured under our unified protocol. The epoch column indicates the COCO training schedule, where * denotes additional epochs without strong augmentation. ‡ marks models without Objects365 (Shao et al., 2019) detector pre-training. Within each comparison group, the top three distinct results for median latency and every reported accuracy metric are highlighted in dark, medium, and light blue, respectively; rankings use unrounded values.

Model	#Epochs	#Params (M)	GFLOPs	Latency (ms) ↓			COCO AP ↑					
				Min.	Mean	Median	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv9-S‡ (Wang et al., 2024b)	500	7	26	1.651	2.326	2.294	46.8	61.8	48.6	25.7	49.9	61.0
YOLOv10-S‡ (Wang et al., 2024a)	500	7	22	0.870	0.934	0.933	46.3	63.0	50.4	26.8	51.0	63.8
YOLO11-S‡ (Jocher, 2024)	500	9	22	1.078	1.523	1.490	46.6	63.4	50.3	28.7	51.3	64.1
YOLOv12-S-Turbo‡ (Tian et al., 2025)	600	9	19	1.339	1.802	1.772	47.6	64.5	51.5	28.3	52.7	65.9
RT-DETRv2-S‡ (Lv et al., 2024)	120	20	60	1.184	1.279	1.291	48.1	65.1	52.1	30.2	51.5	63.9
DEIM-S‡ (Huang et al., 2025b)	132*	10	25	1.090	1.171	1.180	49.0	65.9	53.1	30.4	52.6	65.7
DEIMv2-S‡ (Huang et al., 2025a)	132*	10	26	1.550	1.676	1.687	50.9	68.4	55.1	31.3	55.3	70.2
RT-DETRv4-S‡ (Liao et al., 2026)	132*	10	25	1.079	1.166	1.174	49.7	66.8	54.1	30.2	53.6	66.9
LW-DETR-S (Chen et al., 2024)	60	15	17	1.035	1.115	1.123	48.0	66.9	51.7	26.8	52.5	65.5
D-FINE-S (Peng et al., 2025)	–	10	25	1.103	1.198	1.194	50.7	67.6	55.1	32.7	54.6	66.5
RF-DETR-S (Robinson et al., 2026)	–	32	60	1.207	1.285	1.299	52.9	71.9	57.0	32.0	58.3	73.0
YOLO26-S (Jocher et al., 2026)	70	10	21	0.922	0.977	0.962	47.8	64.6	52.1	29.1	52.5	64.3
ECDet-S‡ (Liu et al., 2026)	74	10	26	1.511	1.641	1.652	51.7	69.4	55.8	32.3	56.4	70.5
GTR-S (ours)‡	30	12.1	33.8	1.215	1.225	1.225	50.7	68.5	54.7	31.1	55.5	70.2
GTR-S (ours)	30	12.1	33.8	1.215	1.225	1.225	53.6	71.1	58.3	36.4	58.3	70.2
YOLOv9-M‡ (Wang et al., 2024b)	500	20	76	2.023	2.713	2.675	51.4	67.2	54.6	32.0	55.7	66.4
YOLOv10-M‡ (Wang et al., 2024a)	500	15	59	1.415	1.437	1.426	51.1	68.1	55.8	33.8	56.5	67.0
YOLO11-M‡ (Jocher, 2024)	500	20	68	1.643	2.086	2.061	51.2	67.9	55.3	33.0	56.7	67.5
YOLOv12-M-Turbo‡ (Tian et al., 2025)	600	20	60	1.917	2.368	2.342	52.5	69.9	57.1	35.2	57.8	69.7
RT-DETRv2-M‡ (Lv et al., 2024)	120	31	92	1.593	1.727	1.741	49.9	67.5	54.1	32.0	53.2	66.5
DEIM-M‡ (Huang et al., 2025b)	102*	19	57	1.612	1.732	1.744	52.7	70.0	57.3	35.3	56.7	69.5
DEIMv2-M‡ (Huang et al., 2025a)	102*	18	52	2.193	2.390	2.398	53.0	70.2	57.6	34.2	57.4	71.5
RT-DETRv4-M‡ (Liao et al., 2026)	102*	19	57	1.598	1.731	1.742	53.5	71.1	58.1	34.9	57.7	72.1
LW-DETR-M (Chen et al., 2024)	60	28	43	1.342	1.433	1.449	52.6	69.9	56.7	32.6	57.7	70.7
D-FINE-M (Peng et al., 2025)	–	19	57	1.603	1.732	1.740	55.1	72.6	59.7	37.9	59.4	71.7
RF-DETR-M (Robinson et al., 2026)	–	34	79	1.326	1.428	1.431	54.7	73.5	59.2	36.1	59.7	73.8
YOLO26-M (Jocher et al., 2026)	80	20	68	1.459	1.481	1.473	52.5	69.8	57.2	36.2	56.9	68.5
ECDet-M‡ (Liu et al., 2026)	62	18	53	1.901	2.080	2.095	54.3	72.2	58.7	35.9	59.1	72.7
GTR-M (ours)‡	30	22.7	62.6	1.450	1.462	1.462	54.0	72.2	58.5	35.3	59.1	73.1
GTR-M (ours)	30	22.7	62.6	1.450	1.462	1.462	57.3	75.0	62.4	41.1	62.0	74.1
YOLOv9-C‡ (Wang et al., 2024b)	500	25	102	2.049	2.716	2.680	53.0	70.2	57.8	36.2	58.5	69.3
YOLOv10-L‡ (Wang et al., 2024a)	500	24	120	2.042	2.059	2.056	53.2	70.1	58.1	35.8	58.5	69.4
YOLO11-L‡ (Jocher, 2024)	500	25	87	2.195	2.641	2.617	53.4	70.1	58.2	35.6	59.1	69.2
YOLOv12-L-Turbo‡ (Tian et al., 2025)	600	27	82	2.879	3.328	3.305	53.8	71.0	58.6	36.9	59.4	71.0
RT-DETRv2-L‡ (Lv et al., 2024)	72	42	136	2.127	2.317	2.332	53.4	71.6	57.4	36.1	57.9	70.8
DEIM-L‡ (Huang et al., 2025b)	58*	31	91	2.251	2.446	2.460	54.7	72.4	59.4	36.9	59.6	71.8
DEIMv2-L‡ (Huang et al., 2025a)	68*	32	97	2.560	2.759	2.776	56.0	73.5	61.1	37.6	60.9	74.9
RT-DETRv4-L‡ (Liao et al., 2026)	58*	31	91	2.264	2.450	2.464	55.4	73.0	60.3	37.1	60.1	72.9
LW-DETR-L (Chen et al., 2024)	60	47	72	2.010	2.152	2.174	56.1	74.6	60.9	37.2	60.4	73.0
D-FINE-L (Peng et al., 2025)	–	31	91	2.253	2.452	2.462	57.1	74.7	62.0	40.0	61.5	74.2
RF-DETR-L (Robinson et al., 2026)	–	34	126	1.805	1.952	1.968	56.5	75.1	61.3	39.0	61.0	73.9
YOLO26-L (Jocher et al., 2026)	60	25	86	1.953	1.969	1.967	54.3	71.5	59.4	37.8	58.6	70.3
ECDet-L‡ (Liu et al., 2026)	50	31	101	2.493	2.713	2.730	57.0	75.1	61.7	38.7	62.5	75.0
GTR-L (ours)‡	30	37.2	106	1.902	1.908	1.908	55.5	73.7	60.3	36.8	61.1	74.8
GTR-L (ours)	30	37.2	106	1.902	1.908	1.908	58.9	76.8	64.3	42.4	64.0	76.0
YOLOv9-E‡ (Wang et al., 2024b)	500	57	189	3.960	4.633	4.598	55.6	72.8	60.6	40.2	61.0	71.4
YOLOv10-X‡ (Wang et al., 2024a)	500	30	160	2.462	2.484	2.482	54.4	71.3	59.3	37.0	59.8	70.9
YOLO11-X‡ (Jocher, 2024)	500	57	195	3.331	3.771	3.747	54.7	71.6	59.5	37.7	59.7	70.2
YOLOv12-X-Turbo‡ (Tian et al., 2025)	600	59	185	4.350	4.791	4.769	55.4	72.5	60.3	38.9	60.8	70.9
RT-DETRv2-X‡ (Lv et al., 2024)	72	76	259	3.209	3.260	3.259	54.3	72.8	58.8	35.8	58.8	72.1
DEIM-X‡ (Huang et al., 2025b)	58*	62	202	3.336	3.388	3.384	56.5	74.0	61.5	38.8	61.4	74.2
DEIMv2-X‡ (Huang et al., 2025a)	58*	50	152	3.218	3.288	3.276	57.8	75.3	63.2	39.1	62.9	75.9
RT-DETRv4-X‡ (Liao et al., 2026)	58*	62	202	3.332	3.367	3.363	57.0	74.6	62.1	39.5	61.9	74.8
LW-DETR-X (Chen et al., 2024)	60	118	174	3.643	3.689	3.686	58.3	76.9	63.3	40.9	63.3	74.8
D-FINE-X (Peng et al., 2025)	–	62	202	3.331	3.397	3.385	59.3	76.8	64.6	42.3	64.2	76.4
RF-DETR-X (Robinson et al., 2026)	–	126	300	2.971	3.227	3.243	58.6	77.4	63.8	40.3	63.9	76.2
YOLO26-X (Jocher et al., 2026)	40	55	194	3.069	3.088	3.087	56.9	74.1	62.1	41.3	61.2	72.7
ECDet-X‡ (Liu et al., 2026)	50	49	151	2.950	3.025	3.021	57.9	76.0	62.9	38.7	63.4	76.1
GTR-X (ours)‡	30	46.5	130.2	2.104	2.117	2.115	56.2	74.5	61.1	37.3	61.8	74.9
GTR-X (ours)	30	46.5	130.2	2.104	2.117	2.115	59.4	77.3	64.7	42.1	64.6	76.4

C.2 Full COCO Instance Segmentation Comparison

Table 8 gives the full mask-metric breakdown. GTR has the highest overall mask AP in each reported scale group. GTR-X improves both mask AP and latency over RF-DETR-Seg-X. GTR attains these results without the SAM2 pseudo-mask supervision used by the †-marked baselines.

Table 8: **Comparison with real-time instance segmentation models on COCO `val2017` (Lin et al., 2014).** Baseline results follow EdgeCrafter (Liu et al., 2026), with latency re-measured under our unified protocol (Appendix E.2). † denotes Objects365 pre-training (Shao et al., 2019) with SAM2 pseudo masks (Ravi et al., 2025) (35× COCO boxes). Dark-to-light blue highlights the top three results within each model-size group.

Model	#Params (M)	GFLOPs	Latency (ms) ↓			Mask AP ↑					
			Min.	Mean	Median	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLO26-Seg-S† (Jocher et al., 2026)	10.4	34.2	1.151	1.194	1.163	40.0	61.5	43.0	21.0	44.5	57.3
RF-DETR-Seg-S† (Robinson et al., 2026)	33.7	70.6	1.339	1.427	1.445	43.1	66.2	45.9	21.9	48.5	64.1
ECInsSeg-S (Liu et al., 2026)	10.3	33.1	1.626	1.739	1.755	43.0	65.7	46.0	20.8	46.3	65.9
GTR-S (ours)	12.6	46.8	1.455	1.465	1.465	45.0	67.9	48.3	23.8	48.4	65.9
YOLO26-Seg-M† (Jocher et al., 2026)	23.6	121.5	1.992	2.006	2.004	44.1	66.8	47.7	25.6	48.9	60.2
RF-DETR-Seg-M† (Robinson et al., 2026)	35.7	102.0	1.548	1.647	1.672	45.3	68.4	48.8	25.5	50.4	65.3
ECInsSeg-M (Liu et al., 2026)	20.1	64.2	2.077	2.252	2.252	45.2	68.2	48.3	22.9	49.0	68.1
GTR-M (ours)	23.6	84.2	1.797	1.807	1.807	47.7	71.3	51.6	27.7	51.3	69.3
YOLO26-Seg-L† (Jocher et al., 2026)	28.0	139.8	2.574	2.598	2.597	45.5	68.7	49.2	27.1	50.4	62.8
RF-DETR-Seg-L† (Robinson et al., 2026)	36.2	151.1	1.907	2.033	2.055	47.1	70.5	50.9	28.4	52.1	65.6
ECInsSeg-L (Liu et al., 2026)	33.6	110.8	2.662	2.862	2.869	47.1	70.9	50.5	24.8	51.1	69.6
GTR-L (ours)	38.1	127.6	2.241	2.252	2.250	49.5	73.5	53.6	28.5	53.5	71.3
YOLO26-Seg-X† (Jocher et al., 2026)	62.8	313.5	4.096	4.143	4.139	47.0	70.8	51.1	29.7	51.8	63.1
RF-DETR-Seg-X† (Robinson et al., 2026)	38.1	260.0	3.064	3.299	3.311	48.8	72.2	53.1	30.6	53.3	65.9
ECInsSeg-X (Liu et al., 2026)	49.9	168.1	3.125	3.147	3.140	48.4	72.2	52.0	26.3	52.7	71.1
GTR-X (ours)	47.4	151.6	2.458	2.473	2.472	49.8	74.2	53.8	28.5	53.7	71.4

C.3 Full COCO Human Pose Estimation Comparison

Table 9 complements the representative main-paper comparison with RTMO, YOLO11-Pose, and ED-Pose where the corresponding model scales are available. Across this broader comparison, GTR attains the highest keypoint AP and AR at every scale.

Table 9: **Complete comparison with human pose estimators on COCO (Lin et al., 2014) `val2017`.** Non-latency baseline values follow EdgeCrafter (Liu et al., 2026); latencies use our unified protocol. † denotes additional Objects365 supervision (Shao et al., 2019). Dark-to-light blue marks the first three ranks for median latency and accuracy; ties share a rank, and rankings use unrounded values.

Model	#Params (M)	GFLOPs	Latency (ms) ↓			Keypoint metrics ↑					
			Min.	Mean	Median	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{M}	AP^{val}_{L}	AR^{val}
RTMO-S (Lu et al., 2024)	9.9	30.7	0.645	0.648	0.646	67.7	87.8	73.7	–	–	71.5
YOLO11-Pose-S (Jocher, 2024)	9.9	23.2	0.701	0.704	0.703	58.9	86.3	64.8	54.0	68.0	66.1
YOLO26-Pose-S (Jocher et al., 2026)	10.4	23.9	0.811	0.814	0.813	63.1	86.6	68.8	56.5	73.7	69.0
DETRPose-S† (Janampa & Pattichis, 2026)	11.5	33.1	1.215	1.219	1.218	67.0	87.6	72.8	60.2	77.4	73.5
ECPose-S (Liu et al., 2026)	9.9	30.4	1.516	1.522	1.519	68.9	89.1	75.2	60.7	81.1	74.6
GTR-S (ours)	11.9	37.0	1.447	1.455	1.455	70.1	89.7	76.8	62.6	81.1	76.1
RTMO-M (Lu et al., 2024)	22.6	69	1.147	1.153	1.153	70.9	89.0	77.8	–	–	74.7
YOLO11-Pose-M (Jocher, 2024)	20.9	71.7	1.220	1.227	1.227	64.9	89.4	72.4	62.2	71.6	72.2
YOLO26-Pose-M (Jocher et al., 2026)	21.5	73.1	1.350	1.355	1.353	68.8	89.6	75.5	64.0	77.2	74.6
DETRPose-M† (Janampa & Pattichis, 2026)	20.8	67.3	1.846	1.853	1.851	69.4	89.2	75.4	63.2	79.0	75.5
ECPose-M (Liu et al., 2026)	19.8	62.8	2.099	2.110	2.109	72.4	90.9	78.6	65.2	83.6	78.2
GTR-M (ours)	22.8	69.8	1.849	1.860	1.860	74.1	91.5	80.7	67.5	84.1	79.6
RTMO-L (Lu et al., 2024)	44.8	136.7	1.769	1.779	1.778	72.4	89.9	78.8	–	–	76.8
YOLO11-Pose-L (Jocher, 2024)	26.2	90.7	1.701	1.708	1.705	66.1	89.9	73.6	63.2	73.1	73.3
YOLO26-Pose-L (Jocher et al., 2026)	25.9	91.3	1.780	1.788	1.788	70.4	90.5	77.4	65.7	78.4	75.9
DETRPose-L† (Janampa & Pattichis, 2026)	32.8	107.1	2.639	2.653	2.650	72.5	90.6	79.0	66.3	82.2	78.7
ECPose-L (Liu et al., 2026)	34.3	111.7	2.768	2.783	2.783	73.5	91.7	79.9	66.4	84.4	78.8
GTR-L (ours)	38.4	115.5	2.320	2.331	2.328	74.7	91.9	81.6	68.3	84.5	79.9
ED-Pose (Yang et al., 2023)	218	422.6	10.266	10.293	10.292	74.3	91.5	81.7	68.5	82.7	–
YOLO11-Pose-X (Jocher, 2024)	58.8	203.3	2.765	2.777	2.777	69.5	91.1	77.4	66.6	76.0	76.3
YOLO26-Pose-X (Jocher et al., 2026)	57.6	201.7	2.774	2.790	2.788	71.6	91.6	78.9	67.4	79.5	77.2
DETRPose-X† (Janampa & Pattichis, 2026)	73.3	239.5	4.129	4.147	4.145	73.3	90.5	79.4	67.5	82.7	79.4
ECPose-X (Liu et al., 2026)	50.6	172.2	3.275	3.286	3.281	74.8	92.2	81.5	68.0	85.4	80.1
GTR-X (ours)	47.6	142.9	2.573	2.588	2.590	75.5	92.4	81.8	69.1	85.3	80.7

C.4 Full DOTA-v1.0 Oriented Object Detection Comparison

Table 10 expands the representative main-paper comparison with the smaller YOLO26-obb and RTMDet-R variants. Compared with RTMDet-R-l, GTR-X gains 0.8 AP_{50} with 6.0M fewer parameters. Relative to the single-scale Oriented-DETR (Swin-T) (Zhao et al., 2024), GTR-X gains 1.5 AP_{50} and is 3.9× faster. In the class-wise breakdown, GTR-X attains the highest plane and harbor AP_{50} among all detectors with reported per-class results (89.5 and 84.5).

Table 10: **Complete oriented object detection comparison on the DOTA-v1.0 (Xia et al., 2018) test set.** We report overall and available per-class AP at an IoU threshold of 0.50. Methods use multi-scale training and testing unless marked †, which denotes single-scale training and testing. YOLO26 accuracy and complexity follow its official end-to-end results; RiO-DETR values are taken from Hu et al. (2026). Latencies use our RTX 4090 protocol; dashes denote unreported entries. Within each detector family, dark and light blue mark the first and second ranks, respectively, for median latency and accuracy; rankings are determined at the reported precision, and ties share a rank.

Model	Backbone	#Params (M)	GFLOPs	Latency (ms) ↓			AP_{50} ↑	Per-class AP_{50} ↑													
				Min.	Mean	Median		PL	BD	BR	GTF	SV	LV	SH	TC	BC	ST	SBF	RA	HA	SP	HC
CNN-based oriented object detectors																					
RTMDet-R-m (Lyu et al., 2022)	CSPNeXt-m	24.7	100	2.356	2.364	2.363	80.3	87.1	85.8	56.3	80.3	80.0	84.7	88.2	90.9	88.5	87.6	70.7	70.0	78.4	80.9	74.5
RTMDet-R-l (Lyu et al., 2022)	CSPNeXt-l	52.3	205	3.663	3.673	3.671	80.5	88.4	85.0	57.3	80.5	80.6	84.9	88.1	90.9	86.3	87.6	69.3	70.6	78.6	81.0	79.2
YOLO26n-obb (Jocher et al., 2026)	YOLO26n	2.4	14.8	0.691	0.694	0.693	78.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOLO26s-obb (Jocher et al., 2026)	YOLO26s	9.8	56.7	1.092	1.099	1.099	80.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOLO26m-obb (Jocher et al., 2026)	YOLO26m	21.2	184.9	2.082	2.093	2.093	81.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOLO26l-obb (Jocher et al., 2026)	YOLO26l	25.6	232.4	2.700	2.709	2.707	81.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOLO26x-obb (Jocher et al., 2026)	YOLO26x	57.6	520.1	5.073	5.087	5.085	81.7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DETR-based oriented object detectors																					
RHINO-DETR† (Lee et al., 2025)	R-50	47.6	566	9.287	9.319	9.314	78.7	88.2	85.1	55.8	72.7	80.2	83.1	89.0	90.8	87.1	86.8	65.3	71.6	77.7	81.2	64.7
RHINO-DETR† (Lee et al., 2025)	Swin-T	50.8	609	10.345	10.368	10.365	79.4	88.2	84.8	58.5	77.7	81.1	85.6	89.2	90.9	87.0	86.4	65.5	71.3	78.2	82.8	64.3
Oriented-DETR† (Zhao et al., 2024)	R-50	57.2	302	14.904	14.929	14.927	79.1	89.2	86.4	57.7	75.3	81.1	84.7	89.1	90.9	86.1	87.0	59.5	70.3	79.3	81.5	68.8
Oriented-DETR† (Zhao et al., 2024)	Swin-T	57.7	309	15.462	15.492	15.492	79.8	89.4	85.1	57.8	75.0	81.2	86.1	89.1	90.9	88.7	87.0	62.9	69.1	80.7	82.8	71.0
RiO-DETR-n† (Hu et al., 2026)	HGNetv2-B0	4.0	17	-	-	-	78.4	87.3	86.1	54.9	73.6	80.5	85.1	88.0	90.8	87.4	86.9	60.9	72.0	76.8	73.4	71.8
RiO-DETR-s† (Hu et al., 2026)	HGNetv2-B0	8.2	53	-	-	-	80.3	85.6	84.8	57.5	75.5	80.7	86.2	89.0	90.8	88.4	88.2	65.2	74.2	78.3	80.1	79.6
RiO-DETR-m† (Hu et al., 2026)	HGNetv2-B2	18.6	158	-	-	-	80.9	85.9	86.4	61.7	78.1	82.0	86.8	89.2	90.8	88.9	87.7	67.4	73.3	78.8	77.4	78.8
RiO-DETR-l† (Hu et al., 2026)	HGNetv2-B4	27.5	230	-	-	-	81.7	86.7	86.8	60.8	79.4	82.0	86.4	89.0	90.8	88.7	88.5	70.0	76.0	79.2	83.4	76.9
RiO-DETR-x† (Hu et al., 2026)	HGNetv2-B5	62.5	527	-	-	-	81.8	88.3	87.5	61.9	79.2	83.2	86.5	89.2	90.8	88.4	87.8	71.2	75.1	78.5	82.2	77.1
RiO-DETR-m (Hu et al., 2026)	HGNetv2-B2	18.6	158	-	-	-	81.5	87.3	87.2	61.7	81.8	82.5	86.2	89.2	90.6	88.7	88.7	74.5	71.4	78.8	74.4	79.7
RiO-DETR-x (Hu et al., 2026)	HGNetv2-B5	62.5	527	-	-	-	81.8	83.8	87.3	62.6	79.9	82.7	86.3	89.1	90.6	88.5	88.0	72.5	75.2	78.9	81.8	79.2
GTR-S (ours)	GLA	12.1	82.8	1.920	1.943	1.946	80.1	89.2	83.9	58.7	77.5	82.6	86.1	88.5	90.8	87.8	87.7	61.4	68.6	78.5	79.8	81.7
GTR-X (ours)	GLA	46.3	324	3.927	3.961	3.960	81.3	89.5	85.1	59.6	78.0	82.6	86.4	88.6	90.9	87.9	87.6	67.2	73.3	84.5	80.1	77.9

C.5 Full Cityscapes Semantic Segmentation Comparison

Table 11: **Complete semantic segmentation comparison on the Cityscapes (Cordts et al., 2016) validation set.** Latency is measured in FP16 with batch size 1. Dark blue indicates the better value within each matched model-size pair; ties share the same rank.

Model	#Params (M)	GFLOPs	Latency (ms) ↓			mIoU ↑
			Min.	Mean	Median	
YOLO26s-sem (Jocher et al., 2026)	6.5	88.8	0.752	0.758	0.758	80.8
GTR-S (ours)	6.9	82.8	1.489	1.495	1.495	81.5
YOLO26m-sem (Jocher et al., 2026)	14.3	304.5	1.601	1.617	1.617	82.0
GTR-M (ours)	12.9	153.0	1.961	1.971	1.970	83.0
YOLO26l-sem (Jocher et al., 2026)	17.9	384.7	2.155	2.167	2.167	82.9
GTR-L (ours)	25.0	258.7	2.922	2.958	2.958	83.2
YOLO26x-sem (Jocher et al., 2026)	40.2	861.7	4.291	4.326	4.325	83.6
GTR-X (ours)	32.2	317.1	3.442	3.483	3.482	83.6

Table 11 reports all four evaluated model scales. GTR-S/M/L outperform their size-matched YOLO26-sem counterparts by 0.7/1.0/0.3 mIoU points, respectively. GTR-X matches YOLO26x-sem at 83.6 mIoU while using 20.0% fewer parameters, 63.2% fewer FLOPs, and 19.5% lower median latency.

C.6 Full NYU Depth Estimation Comparison

Table 12: **Complete ground-truth-aligned monocular depth comparison on the NYU Depth V2 (Silberman et al., 2012) Eigen test split (Eigen et al., 2014).** ZipDepth is re-evaluated using the same filled ground-truth data as YOLO26 and GTR. YOLO26 and GTR scores use six test-time views and a per-image log-affine fit to ground truth, without NYU-specific fine-tuning (Appendix D.1.7). Latency measures a single forward pass. Dark blue marks the better value within each matched YOLO26/GTR pair for median latency and each accuracy metric.

Model	#Params (M)	GFLOPs	Latency (ms) ↓			δ_1 ↑	AbsRel ↓	RMSE ↓
			Min.	Mean	Median			
ZipDepth (Tosi et al., 2026)	6.1	7.9	0.332	0.333	0.334	0.919	0.091	0.385
YOLO26s-depth (Jocher et al., 2026)	13.2	67.9	0.847	0.850	0.850	0.896	0.104	0.399
GTR-S (ours)	11.5	65.0	1.290	1.297	1.296	0.946	0.074	0.336
YOLO26m-depth (Jocher et al., 2026)	23.3	130.7	1.304	1.308	1.307	0.921	0.089	0.364
GTR-M (ours)	19.4	93.2	1.501	1.509	1.508	0.952	0.069	0.319
YOLO26l-depth (Jocher et al., 2026)	27.7	157.2	1.744	1.747	1.747	0.930	0.083	0.351
GTR-L (ours)	33.9	136.6	1.936	1.948	1.948	0.951	0.069	0.328
YOLO26x-depth (Jocher et al., 2026)	57.0	302.0	2.721	2.730	2.730	0.933	0.080	0.344
GTR-X (ours)	41.1	159.4	2.144	2.159	2.158	0.954	0.067	0.317

Table 12 reports all four evaluated YOLO26/GTR scales and ZipDepth (Tosi et al., 2026). At every scale, GTR improves δ_1, AbsRel, and RMSE over the corresponding YOLO26 depth variant while using fewer GFLOPs. GTR-X additionally reduces median latency from 2.730 to 2.158 ms.

ZipDepth is re-evaluated on the same filled ground-truth data as YOLO26 and GTR, yielding $\delta_1 = 0.919$, AbsRel $= 0.091$, and RMSE $= 0.385$. The original ZipDepth paper reports $\delta_1 = 0.933$ and AbsRel $= 0.084$ under its own evaluation protocol.

C.7 Detailed Ablation Results

Spatial mechanism. Table 13 in Section 3.3 compares S-SwiGLU with learned positional embeddings. S-SwiGLU improves overall AP at both resolutions with a small FLOP increase.

Table 13: **S-SwiGLU vs. learned positional embeddings on GTR-S.** S-SwiGLU uses no explicit positional embedding.

Input resolution	Spatial mechanism	#Params (M)	GFLOPs	COCO AP ↑					
				AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
768^2	Learned positional embedding	12.1	47.6	53.9	71.6	58.6	38.0	57.8	70.2
	S-SwiGLU (ours)	12.1	47.8	**55.1**	**72.5**	**60.2**	**38.7**	**59.6**	**70.7**
1024^2	Learned positional embedding	12.1	83.2	55.3	72.7	60.3	40.6	58.7	**70.7**
	S-SwiGLU (ours)	12.1	83.7	**56.6**	**73.9**	**61.9**	**42.1**	**60.8**	70.4

Scan schedule. Table 14 reports GTR-L results for all evaluated scan schedules. Four directions yield the highest overall AP. The single-direction comparison shows a larger change when removing S-SwiGLU than when reducing the number of scan directions.

Table 14: **Effect of scan direction and spatial mechanism on GTR-L.** LR/RL/TB/BT denote left-to-right, right-to-left, top-to-bottom, and bottom-to-top scans.

Variant	#Params (M)	GFLOPs	COCO AP ↑					
			AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
Four directions (LR/RL/TB/BT) + S-SwiGLU	37.2	106	**55.5**	**73.7**	**60.3**	**36.8**	**61.1**	74.8
Horizontal bidirectional (LR/RL) + S-SwiGLU	37.2	106	55.1	73.4	59.8	36.1	60.5	**75.2**
Vertical bidirectional (TB/BT) + S-SwiGLU	37.2	106	55.0	73.3	59.8	35.8	60.7	74.6
Single direction + S-SwiGLU	37.2	106	54.4	72.6	59.0	35.1	59.7	73.9
Single direction + learned positional embedding	37.2	106	45.6	62.3	49.1	27.5	49.1	63.5

Hybrid attention. Table 5 replaces GLA with softmax attention in three backbone blocks at early, uniform, or late positions. The hybrids use FlashAttention-2 (Dao, 2024). They increase GFLOPs from 106 to 119 and median latency by 9.7–11.0%, while overall AP remains within 0.1 of pure GLA. The reported memory values coincide at the displayed precision.

Resolution scaling. Table 6 reports the full GTR-S sweep with Objects365-pre-trained detector initialization (Shao et al., 2019). Most of the accuracy gain occurs for small and medium objects. The batch-one rate is computed as the reciprocal of median latency.

Detector pre-training. Table 15 gives the GTR-S metric breakdown with and without Objects365 (Shao et al., 2019) detector pre-training. Overall AP increases by 2.9 and small-object AP by 5.3. The corresponding four-scale results appear in Table 7.

Table 15: **Effect of Objects365 (Shao et al., 2019) detector pre-training.** A filled circle indicates that detector pre-training on Objects365 is enabled.

Objects365 pre-training	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
–	50.7	68.5	54.7	31.1	55.5	70.2
•	**53.6**	**71.1**	**58.3**	**36.4**	**58.3**	**70.2**

D Experimental Details

Datasets and metrics. For the main detector comparison, we report GTR with and without Objects365 (Shao et al., 2019) detector pre-training before COCO (Lin et al., 2014) `train2017` fine-tuning. We evaluate on COCO `val2017` with AP averaged over IoU thresholds from 0.50 to 0.95, together with AP_{50}, AP_{75}, and AP for small, medium, and large objects. We report box AP for detection and mask AP for instance segmentation. For transfer evaluation, we report standard keypoint AP and AR on COCO `val2017`, and AP_{50} for oriented object detection on the DOTA-v1.0 test set (Xia et al., 2018).

Training protocol. All models are trained on eight NVIDIA A100 GPUs. For the main comparison, the S/M/L/X backbones are initialized with the corresponding distilled GLA checkpoints, followed by pretraining of the full detectors on Objects365 for 36 epochs with a global batch size of 128.

We then train all four GTR scales for 30 epochs with a global batch size of 32. The direct-COCO variants instead start from the output-aligned backbone and otherwise use the same scale-specific COCO fine-tuning configurations as their Objects365-pre-trained counterparts. The Objects365 and COCO stages use AdamW (Loshchilov & Hutter, 2019) with a flat-cosine learning-rate schedule and an exponential moving average; evaluation uses 640×640 inputs. Appendix D.1 reports the remaining scale-specific learning rates, augmentations, loss weights, and model configurations.

Efficiency evaluation. We remeasure every detection and segmentation model in a unified FP16, batch-size-one environment on a single NVIDIA GeForce RTX 4090. Tables 7 and 8 report the minimum, arithmetic mean, and median per-image latency; all comparisons use the median. Compilation, graph capture, autotuning, and warm-up are excluded from timing. For DETR-style models, the measured latency covers the network forward pass alone; for YOLO variants that require NMS, it additionally includes CUDA NMS. Because deployment conversion, input resolution, and post-processing differ across model families, the complete protocol is specified in Appendix E.2.

D.1 Training Configurations

D.1.1 Architecture Configurations

All GTR variants employ a 12-layer, stride-16 GLA backbone and extract multi-scale features from intermediate layers. As summarized in Table 16, the decoder depth, object-query count, and number of training query groups remain fixed across scales. From GTR-S to GTR-L, the embedding dimension and number of GLA heads increase from 192/3 to 384/6. GTR-X retains the GTR-L token-mixing width while increasing the S-SwiGLU expansion ratio from 4 to 6 and the decoder FFN dimension from 1,024 to 2,048.

Table 16: **Architecture configurations of the four GTR variants.**

Setting	GTR-S	GTR-M	GTR-L	GTR-X
Embedding dimension	192	256	384	384
GLA heads	3	4	6	6
S-SwiGLU expansion ratio	4	4	4	6
Encoder width	192	256	256	256
Decoder FFN dimension	512	1024	1024	2048
Decoder layers	4	4	4	4
Object queries	300	300	300	300
Training query groups	3	3	3	3

D.1.2 Final-Output Representation Alignment

Teacher configuration. Following Liu et al. (2026), ECTeacher-S and ECTeacher-B are obtained by fine-tuning pretrained DINOv3-S and DINOv3-B in ECDet-L and ECDet-X, respectively, on COCO `train2017`. ECTeacher-S supervises GTR-S, while ECTeacher-B supervises GTR-M/L/X. During distillation, the adapted teacher backbone is frozen and provides the final patch-token targets in Equation 2.

Distillation data. Following EdgeCrafter (Liu et al., 2026), the image pool combines ImageNet-1K (Deng et al., 2009) and COCO (Lin et al., 2014) `train2017`.

Student optimization. We adopt the optimization recipe of Peng et al. (2023b) without additional hyperparameter tuning. GTR-S/M use the Small/Base presets, while GTR-L/X use the Large preset. As shown in Table 17, all variants are trained for 300 epochs with AdamW and a global batch size of 2,048. GTR-L/X use a longer warm-up (20 vs. 10 epochs) and a higher drop-path rate (0.2 vs. 0.1); all other optimization and augmentation settings are shared.

Table 17: **Training configuration for final-output representation alignment.**

Configuration	GTR-S/M	GTR-L/X
Optimizer	AdamW	
Peak learning rate	1.5×10^{-3}	
Minimum learning rate	1.0×10^{-5}	
Weight decay	0.05	
Training epochs	300	
Optimizer betas	(0.9, 0.999)	
Global batch size	2048	
Warm-up epochs	10	20
Drop-path rate	0.1	0.2
Layer-wise learning-rate decay	✗	
Label smoothing	✗	
Random erasing	✗	
RandAugment	✗	
Repeated augmentation	✓	
ThreeAugmentation	✓	

Figure 6 compares the supervision structure of GTR with representative cross-architecture approaches (Wei & Chellappa, 2025; Li et al., 2026b; Moudgil et al., 2026; Bick et al., 2024). GTR's single stage here means one representation-distillation stage; teacher preparation and downstream supervised training remain separate.

D.1.3 Objects365 Pre-training

Each detector is initialized with its corresponding output-aligned GLA backbone and subsequently pre-trained on Objects365. As shown in Table 18, all scales use the same 36-epoch schedule, global batch size of 128, base learning rate of 2×10^{-3}, and 2,000 warm-up iterations. The backbone

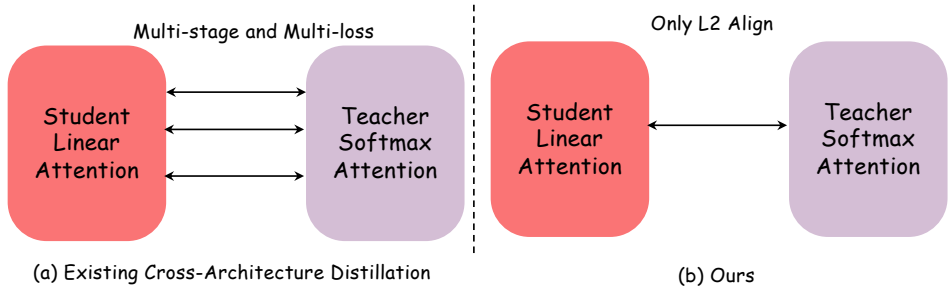

Figure 6: **Comparison of cross-architecture transfer designs.** Representative prior approaches use multiple alignment stages or losses. GTR instead aligns only the final representations of a frozen softmax teacher and a linear-attention student with a squared ℓ_2 objective.

learning rate decreases as model capacity increases, from 10^{-4} for GTR-S/M to 2×10^{-5} and 10^{-5} for GTR-L and GTR-X, respectively.

Table 18: **Objects365 (Shao et al., 2019) detector pre-training hyperparameters.**

Setting	GTR-S	GTR-M	GTR-L	GTR-X
Epochs	36	36	36	36
Global batch size	128	128	128	128
Base LR	2×10^{-3}	2×10^{-3}	2×10^{-3}	2×10^{-3}
Backbone LR	1×10^{-4}	1×10^{-4}	2×10^{-5}	1×10^{-5}
Weight decay	1×10^{-4}	1×10^{-4}	1.25×10^{-4}	1.25×10^{-4}
AdamW β	0.9/0.999	0.9/0.999	0.9/0.999	0.9/0.999
LR schedule	flat-cosine	flat-cosine	flat-cosine	flat-cosine
Flat phase ends at epoch	15	15	15	15
Minimum/base LR ratio	0.1	0.1	0.1	0.1
Warm-up iterations	2000	2000	2000	2000

D.1.4 COCO Fine-tuning

For COCO fine-tuning after Objects365 pre-training, we load the complete detector checkpoint except the classification head. As shown in Table 19, the 30-epoch schedule, global batch size, and base learning rate are fixed across scales. All variants are trained on `train2017` and evaluated on `val2017`.

Table 19: **COCO (Lin et al., 2014) fine-tuning hyperparameters.** Bias and normalization parameters use zero weight decay in both training stages.

Setting	GTR-S	GTR-M	GTR-L	GTR-X
Epochs	30	30	30	30
Global batch size	32	32	32	32
Base LR	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Backbone LR	1.5×10^{-5}	1.5×10^{-5}	2.5×10^{-6}	8×10^{-6}
Weight decay	1×10^{-4}	1×10^{-4}	1.25×10^{-4}	1.25×10^{-4}
LR schedule	flat-cosine	flat-cosine	flat-cosine	flat-cosine
Flat phase ends at epoch	6	6	6	6
Minimum/base LR ratio	0.5	0.5	0.5	0.1
Warm-up iterations	2000	2000	2000	2000

D.1.5 Human Pose Estimation

For human pose estimation, we initialize only the GTR backbone from the COCO detection checkpoint and train all pose-specific parameters from scratch. The training protocol is summarized in

Table 20. All variants use AdamW, a base learning rate of 5×10^{-4}, a global batch size of 32, and the same augmentation probabilities and loss weights.

Table 20: **Human pose estimation hyperparameters.**

Setting	GTR-S	GTR-M	GTR-L	GTR-X
Optimizer	AdamW	AdamW	AdamW	AdamW
Backbone LR	2.5×10^{-5}	2.5×10^{-5}	2.5×10^{-6}	2.5×10^{-6}
Base LR	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Weight decay	1×10^{-4}	1×10^{-4}	1.25×10^{-4}	1.25×10^{-4}
Epochs (with/without aug.)	90 + 2	90 + 2	72 + 2	72 + 2
$\text{Prob}_{\text{mosaic}}$	0.5	0.5	0.5	0.5
$\text{Epochs}_{\text{mosaic}}$	45	45	48	48
$\text{Prob}_{\text{mixup}}$	0.25	0.25	0.25	0.25
$\text{Epochs}_{\text{mixup}}$	45	45	48	48
$\text{Prob}_{\text{copypaste}}$	0.75	0.75	0.75	0.75
$\text{Epochs}_{\text{copypaste}}$	45	45	48	48
Global batch size	32	32	32	32
λ_{cls}	2	2	2	2
λ_{kpt}	10	10	10	10
λ_{oks}	4	4	4	4

D.1.6 Semantic Segmentation

For semantic segmentation, we initialize only the GTR backbone from the pretrained backbone checkpoint and train all task-specific parameters from scratch.

We train and evaluate semantic segmentation on Cityscapes (Cordts et al., 2016) with fine annotations only, and report mIoU on the validation set. During training, images are augmented with random resizing by a factor sampled from $[0.5, 2.0]$, random horizontal flipping, and random cropping to 1024×1024. At inference, we adopt sliding-window evaluation with a 1024×1024 window and a stride of 768 pixels.

For semantic segmentation, we retain only the `scale=[2.0]` branch of the semantic neck, which yields stride-8 features. On top of these features we attach a lightweight FCN head consisting of a 3×3 Conv-BN-ReLU block, dropout ($p = 0.1$), and a 1×1 convolutional classifier over the 19 Cityscapes classes. The output logits are bilinearly upsampled to the input resolution and supervised by a per-pixel cross-entropy loss. Quantitative results are reported in Section 3.2, and qualitative results are provided in Appendix F.4.

Table 21 lists the shared 30-epoch, batch-eight schedule and the scale-specific backbone learning rates. The reported task accuracy uses sliding-window inference, while latency measures one fixed-size network invocation.

Table 21: **Cityscapes (Cordts et al., 2016) semantic-segmentation fine-tuning hyperparameters.**

Setting	GTR-S	GTR-M	GTR-L	GTR-X
Epochs	30	30	30	30
Global batch size	8	8	8	8
Base LR	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Backbone LR	1.5×10^{-4}	1.8×10^{-4}	1.2×10^{-4}	1.2×10^{-4}
Weight decay	1×10^{-4}	1×10^{-4}	1.25×10^{-4}	1.25×10^{-4}
LR schedule	flat-cosine	flat-cosine	flat-cosine	flat-cosine
Flat phase ends at epoch	6	6	6	6
Minimum/base LR ratio	0.5	0.5	0.5	0.5
Warm-up iterations	2,000	2,000	2,000	2,000
Training crop size	1024^2	1024^2	1024^2	1024^2
Sliding-window stride	768	768	768	768

D.1.7 Monocular Depth Estimation with Diverse Training Data

For monocular depth estimation, we initialize only the GTR backbone from the pretrained backbone checkpoint and train all task-specific parameters from scratch.

Training sources and data acquisition. We adopt the same eight-source training mixture used for depth training in YOLO26 (Jocher et al., 2026), comprising ARKitScenes (Baruch et al., 2021), SUN RGB-D (Song et al., 2015), DIODE (Vasiljevic et al., 2019), Hypersim (Roberts et al., 2021), TartanAir (Wang et al., 2020), Virtual KITTI 2 (Cabon et al., 2020), KITTI (Geiger et al., 2013), and pseudo-labeled ImageNet (Deng et al., 2009). To reduce storage and bandwidth overhead, we adopt source-specific acquisition strategies. For Hypersim, we download only the `tonemap.jpg` and `depth_meters.hdf5` files instead of the full 1.9 TB release, reducing the local storage footprint to approximately 11 GB. For KITTI, we retrieve the raw archives with batched HTTP range requests and keep only the color-camera streams, discarding the Velodyne point clouds, grayscale images, and OXTS records. This reduces the amount of data transferred by about 75%.

Depth decoding. All depth targets are decoded as `float32` arrays in meters, and values ≤ 0 are treated as invalid. For SUN RGB-D, we follow the official toolbox and recover depth through the 16-bit rotation $d = ((r \gg 3) \,|\, (r \ll 13))/1000$. We use the official `depth_bfx` targets, which leave measured pixels unchanged while inpainting missing regions and therefore provide nearly dense supervision. Hypersim stores the Euclidean distance d from the optical center rather than planar depth, i.e., the z coordinate in camera space. Using the official 1024×768 intrinsics with focal length $f = 886.81$ pixels, we convert each pixel to planar depth as

$$z(u, v) = d(u, v) \frac{f}{\sqrt{(u - c_x)^2 + (v - c_y)^2 + f^2}}, \tag{8}$$

where (c_x, c_y) is the principal point. Any resulting NaN is set to zero and thus marked invalid. KITTI depth is decoded from `uint16` as $d = r/256$, with zeros retained as invalid. For DIODE, we multiply the depth by the provided validity mask, which maps invalid pixels to zero. All depth maps are resized using nearest-neighbor interpolation only, so that invalid zeros are never blended with valid metric depths.

We further apply dataset-specific preprocessing before enforcing the common training range. In Virtual KITTI 2, sky pixels are encoded with a depth of 655.35 m; we clip these values to 80 m while retaining them as valid. In contrast, sky depths in TartanAir are on the order of 10^4 m; these values are set to zero and treated as invalid. For DIODE and SUN RGB-D, depth values are clipped to 80 m and 10 m, respectively. Across all datasets, training and evaluation use a common maximum depth of 100 m, with depths in the range $[10^{-3}, 100]$ m considered valid.

Depth teacher. We generate metric-depth pseudo-labels with the nested variant of Depth Anything 3 (DA3) (Lin et al., 2026), which jointly estimates camera intrinsics and metric depth and therefore needs no additional scale calibration. The alternative canonical-depth DA3 variant instead predicts depth up to a focal-length-dependent scale and must be rescaled as $d_{\text{metric}} = (f/300)\, d_{\text{canonical}}$, where f is the focal length in pixels and 300 pixels is the canonical focal length used during training. This rescaling is not applicable in our setting: standard ImageNet preprocessing discards the camera metadata and resizes and crops each image, so the effective focal length cannot be recovered reliably. Estimating f from an assumed field of view would inject an uncontrolled, per-image scale error into the pseudo-labels, which the nested variant avoids by construction.

Batched teacher inference. The default DA3 inference pipeline assumes multi-view inputs from a common scene rather than a batch of independent images: it couples the views in a batch through cross-view attention, shared scale alignment, and joint percentile-based depth normalization. Batching unrelated ImageNet images therefore makes each prediction depend on the other images in the batch; we measured median relative deviations of 68–233% from single-image inference. To batch efficiently while preserving single-image behavior, we feed each sample as a single-view input, yielding tensors of shape $(B, 1, 3, H, W)$, and run global alignment and post-processing independently for each batch element. The resulting predictions match single-image inference to within 0.1% relative error. We further bucket images by their resolution after teacher preprocessing rather than by their original resolution, so that every batch shares a single resolution while images with common aspect ratios can still be grouped together. Overall, batching with resolution bucketing reduces the number of teacher forward passes by a factor of approximately 8.

Sampling and augmentation. The measured-depth datasets differ by more than an order of magnitude in size, ranging from roughly 10K images in SUN RGB-D to over 300K in TartanAir. To mitigate this imbalance, we oversample each dataset with an integer repetition factor, using $[6, 3, 2, 2, 2, 1, 3]$ in data-loader order. The pseudo-labeled ImageNet pool is drawn from a separate sampling stream, so that its ratio to the measured-depth data can be controlled independently. For augmentation, we retain only the metric-depth-preserving transformations of Depth Anything V2 (Yang et al., 2024a): color jitter, horizontal flipping, aspect-ratio-preserving short-edge resizing, and random square cropping. Composition-based geometric augmentations such as Mosaic, ZoomOut, and IoUCrop are disabled, since they alter the relationship between image scale and metric depth and thus break geometric consistency.

Head, objective, and decoding. Our depth head follows a DPT-style RefineNet hierarchy (Ranftl et al., 2021): starting from stride-32 features, it progressively fuses those at strides 16, 8, and 4, and upsamples the final prediction to the input resolution. Let $\hat{d}_i$ and d_i denote the predicted and ground-truth depth at pixel i, let N be the number of valid pixels, and let $g_i = \log \hat{d}_i - \log d_i$. We minimize the scale-invariant logarithmic (SiLog) loss (Eigen et al., 2014), a standard objective for metric depth regression,

$$\mathcal{L}_{\text{SiLog}} = \sqrt{\frac{1}{N}\sum_i g_i^2 - \lambda\left(\frac{1}{N}\sum_i g_i\right)^2}, \qquad \lambda = 0.5. \tag{9}$$

Unlike the fully scale-invariant case ($\lambda = 1$), $\lambda = 0.5$ retains a partial penalty on the global log-scale error. SiLog is the only objective used to train the depth head. For decoding, we follow YOLO26 (Jocher et al., 2026) and regress log-depth, $\hat{d} = \exp(\text{clamp}(z, -4, 5))$, where z is the raw head output; this yields an effective range of approximately 0.02–148 m. The clamp serves only for numerical stability and its bounds are dataset-agnostic, so predictions are not saturated at a dataset-specific `max_depth`.

Ground-truth-aligned evaluation protocol. We evaluate on the 654-image Eigen test split of NYU Depth V2 (Silberman et al., 2012; Eigen et al., 2014) without NYU-specific fine-tuning. We use filled ground-truth depth maps for evaluation. Each RGB image is first resized with its shorter side set to 640 pixels while preserving aspect ratio; hence, a 480×640 image becomes 640×853. Because the backbone operates on a square patch grid, two overlapping 640×640 windows cover the resized image, and predictions in their overlap are averaged. We use three image scales, $\{0.75, 1, 1.25\}$, with and without horizontal flipping, yielding six test-time views. Each prediction is bilinearly resized back to the native 480×640 resolution, after which the six depth maps are averaged linearly. Within the Eigen crop [45:471, 41:601], we fit a two-parameter log-affine alignment to the ground-truth depth independently for each image by least squares. For an aligned prediction $\tilde{d}_i$, ground-truth depth d_i, and the set $\mathcal{V}$ of N valid pixels in the crop, we define the three metrics as

$$\delta_1 = \frac{1}{N}\sum_{i\in\mathcal{V}} \mathbf{1}\left[\max\left(\frac{\tilde{d}_i}{d_i}, \frac{d_i}{\tilde{d}_i}\right) < 1.25\right], \tag{10}$$

$$\text{AbsRel} = \frac{1}{N}\sum_{i\in\mathcal{V}} \frac{|\tilde{d}_i - d_i|}{d_i}, \tag{11}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i\in\mathcal{V}} (\tilde{d}_i - d_i)^2}. \tag{12}$$

We compute each metric separately for every image and report its arithmetic mean over all 654 test images. This protocol follows YOLO26 (Jocher et al., 2026). Section 3.2 reports the quantitative comparison, and Appendix F.5 provides additional zero-shot results on indoor and outdoor benchmarks.

Table 22 gives the mixed-data depth schedule. The four models share the 30-epoch duration and batch size of 256, while the L/X backbone learning rates are lower than those of S/M. The ground-truth alignment above is applied only when computing the reported NYU evaluation metrics.

Table 22: **Mixed-data depth pre-training hyperparameters.**

Setting	GTR-S	GTR-M	GTR-L	GTR-X
Epochs	30	30	30	30
Global batch size	256	256	256	256
Base LR	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}
Backbone LR	1×10^{-4}	1×10^{-4}	2×10^{-5}	1×10^{-5}
Weight decay	1×10^{-4}	1×10^{-4}	1.25×10^{-4}	1.25×10^{-4}
LR schedule	flat-cosine	flat-cosine	flat-cosine	flat-cosine
Flat phase ends at epoch	12	12	12	12
Minimum/base LR ratio	0.1	0.1	0.1	0.1
Warm-up iterations	2,000	2,000	2,000	2,000
No-augmentation phase starts at epoch	27	27	27	27
Valid depth range (m)	$[10^{-3}, 100]$	$[10^{-3}, 100]$	$[10^{-3}, 100]$	$[10^{-3}, 100]$
Output parameterization	log-depth	log-depth	log-depth	log-depth

D.1.8 Oriented Object Detection

For oriented object detection, we initialize only the GTR backbone from the pretrained backbone checkpoint and train all task-specific parameters from scratch.

For DOTA-v1.0, we follow the multi-scale training and testing protocol of RiO-DETR (Hu et al., 2026): during both training and testing, each image is rescaled by factors of $\{0.5, 1.0, 1.5\}$ and then cropped into 1024×1024 patches with a stride of 524 pixels, i.e., an overlap of 500 pixels. The task-specific OBB decoder consists of four layers and introduces three modifications for oriented object detection. First, we adopt angle distribution refinement (ADR) (Ding et al., 2026), which represents each oriented box by six discrete distributions: four distances to the sides of its axis-aligned enclosing box and two vertex offsets. These distributions are decoded into oriented boxes by a distance-to-rotated-box conversion, which avoids the angular boundary discontinuity of direct angle regression. Second, we rotate the sampling offsets of deformable attention by the predicted angle θ of each query, so that the sampling pattern aligns with the principal direction of the target. Third, oriented contrastive denoising (OCD) (Ding et al., 2026) perturbs only the axis-aligned coordinates (x_1, y_1, x_2, y_2) of the denoising queries and leaves the angle channel unchanged.

The training objective combines one classification and two regression terms. For classification, the matching-aware loss (MAL) (Huang et al., 2025b) uses a soft target weighted by the rotated IoU with the ground truth. For regression, an ℓ_1 loss supervises the five box parameters (x, y, w, h, θ), and the Gaussian Kullback–Leibler divergence (KLD) loss (Yang et al., 2021) replaces the GIoU loss used in the horizontal-box setting: it models an oriented box as a 2-D Gaussian and thus provides geometry-aware supervision for rotated boxes. As in the original formulation, the divergence D is mapped to a loss by $1 - 1/(\tau + \log(1 + D))$ with $\tau_{\mathrm{KLD}} = 1$. Table 23 summarizes the complete fine-tuning configuration.

Table 23: **DOTA-v1.0 (Xia et al., 2018) oriented-object-detection fine-tuning hyperparameters.**

Setting	GTR-S	GTR-X
Epochs	12	14
Global batch size	16	16
Base LR	5×10^{-4}	5×10^{-4}
Backbone LR	1.8×10^{-4}	8×10^{-6}
Weight decay	1×10^{-4}	1.25×10^{-4}
LR schedule	constant after warm-up	constant after warm-up
Warm-up iterations	2,000	2,000
Drop-path rate	0.0	0.1
Training patch size	1024^2	1024^2
Multi-scale resize ratios	$\{0.5, 1.0, 1.5\}$	$\{0.5, 1.0, 1.5\}$

Table 24: **DRIVE AGX Thor deployment environment.** All results in Tables 25 and 26 use FP16.

Component	Configuration
Board	NVIDIA DRIVE AGX Thor
GPU	Blackwell (`sm_110`)
Operating system	DRIVE OS 7.0.5.0 / Ubuntu 24.04.3 LTS
CUDA	13.0
TensorRT	10.14.2.2
Precision	FP16

Table 25: **FP16 TensorRT deployment on DRIVE AGX Thor at batch size one.** Cosine similarity compares FP16 TensorRT outputs with FP32 PyTorch reference outputs.

Task	Model	Input	Median (ms) ↓	Images/s ↑	Cosine sim. ↑
Detection	GTR-S	640^2	2.282	438	0.9997
	GTR-M	640^2	2.721	368	0.9999
	GTR-L	640^2	3.527	284	0.9998
	GTR-X	640^2	4.080	245	0.9999
Instance seg.	GTR-Seg-S	640^2	3.474	288	0.9992
	GTR-Seg-M	640^2	4.363	229	0.9997
	GTR-Seg-L	640^2	5.130	195	0.9993
	GTR-Seg-X	640^2	5.703	175	0.9993
Oriented det.	GTR-OBB-S	1024^2	4.310	232	0.9998
	GTR-OBB-M	1024^2	5.491	182	0.9998
	GTR-OBB-L	1024^2	7.602	132	0.9999
	GTR-OBB-X	1024^2	8.769	114	0.9998
Pose	GTR-Pose-S	640^2	2.459	407	0.9994
	GTR-Pose-M	640^2	3.082	324	0.9996
	GTR-Pose-L	640^2	3.909	256	0.9989
	GTR-Pose-X	640^2	4.455	224	0.9989
Semantic seg.	GTR-SemSeg-S	1024^2	3.579	279	0.9993
	GTR-SemSeg-M	1024^2	4.655	215	0.9998
	GTR-SemSeg-L	1024^2	6.792	147	0.9996
	GTR-SemSeg-X	1024^2	7.916	126	0.9996
Depth	GTR-Depth-S	640^2	2.675	374	1.0000
	GTR-Depth-M	640^2	3.071	326	1.0000
	GTR-Depth-L	640^2	3.872	258	1.0000
	GTR-Depth-X	640^2	4.389	228	1.0000

E Efficiency and Reproducibility Details

E.1 DRIVE AGX Thor Deployment

Platform and execution path. Table 24 specifies the on-device environment. The Thor-targeted operators are compiled for `sm_110` and integrated into TensorRT; they are distinct builds from the `sm_89` PyTorch-extension GLA operator benchmarked in Appendix E.3.

Roofline analysis. Figure 7 motivates optimizing memory traffic rather than peak compute. Across six tasks and four model scales, the plotted baseline and optimized points lie well left of the $1{,}032\,\mathrm{FLOP/byte}$ ridge point and near the bandwidth roofs, consistent with memory-traffic limitations on Thor. Compared with the TensorRT baseline, our graph transformations and fused CUDA kernels shift the operating points upward and to the right by eliminating intermediate memory traffic. The optimized models reach 89–141% of the nominal DRAM roof; values above 100% are compatible with on-chip reuse under the plotted traffic model.

Deployment results. Tables 25 and 26 report the complete TensorRT measurements. At batch size one, latency ranges from 2.282 to 8.769 ms across all tasks and model scales, while the cosine similarity between FP16 TensorRT outputs and their FP32 PyTorch references remains consistently high, ranging from 0.9989 to 1.0000. Increasing the batch size to eight raises the peak throughput to 571 images/s for GTR-S detection. For the 1024×1024 oriented-detection and semantic-segmentation workloads, batch-one inference provides higher throughput than batch eight, consistent with the memory-bound regime identified in Figure 7.

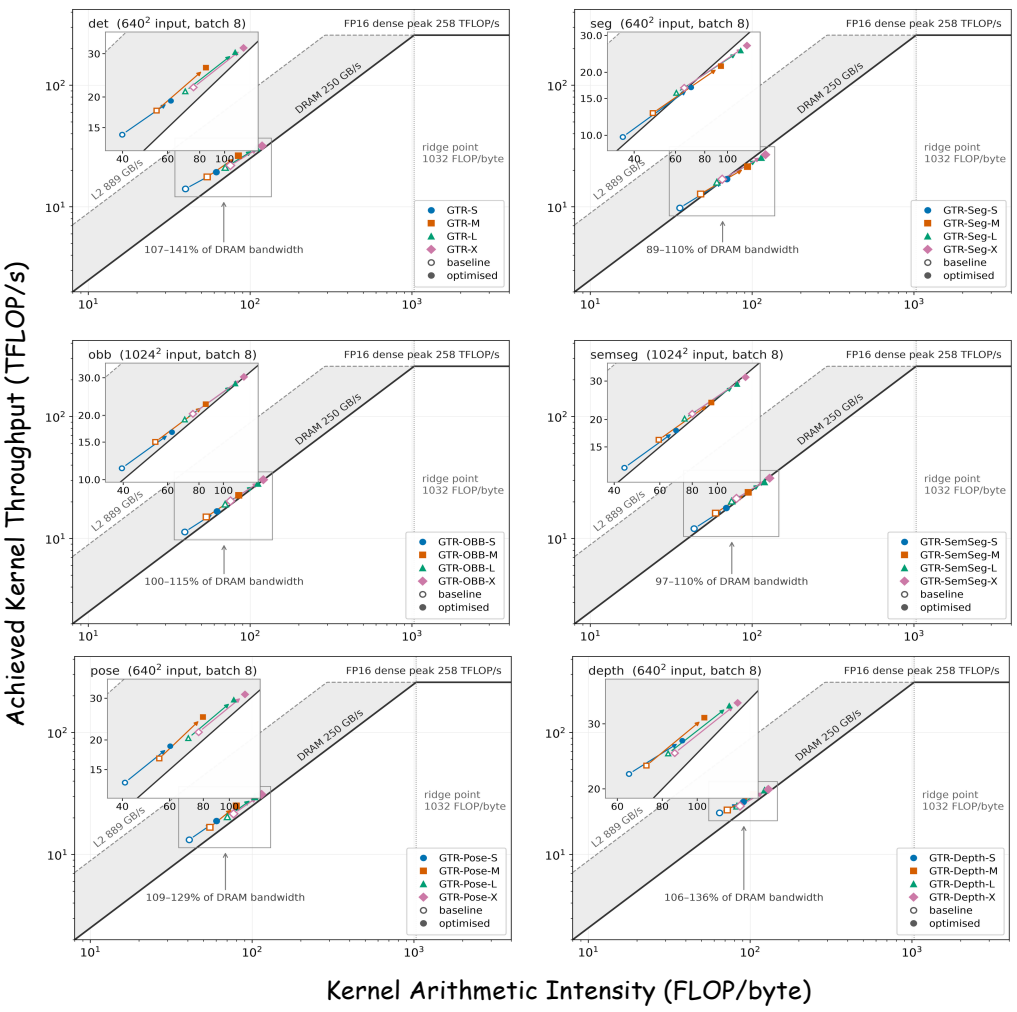

Figure 7: **Roofline analysis of the TensorRT deployment on DRIVE AGX Thor.** Open markers denote the original TensorRT baseline, which uses only TensorRT's built-in optimizations; filled markers denote our graph–kernel-optimized implementation.

E.2 GPU Latency Measurement Protocol

E.2.1 Hardware, Software, and Deployment

Our protocol measures steady-state GPU latency under a unified deployment, compilation, and execution pipeline. The model forward-pass measurements use a single NVIDIA GeForce RTX 4090 GPU with Python 3.11, PyTorch 2.11.0+cu128, torchvision 0.26.0+cu128, and CUDA 12.8. We use a batch size of one and FP16 precision throughout.

For object detection, RF-DETR-S/M/L/X (Robinson et al., 2026) are evaluated at their released native resolutions of $512/576/704/700$, respectively; the non-monotonic L/X resolutions follow the official model configurations. All other detectors use 640×640 inputs. For instance segmentation, RF-DETR-Seg-S/M/L/X use their native resolutions of $384/432/504/624$, respectively, while all other models use 640×640 inputs.

Before compilation, we invoke the official `deploy()`, `export()`, or `fuse()` routine, when available, to apply deployment conversion and Conv–BN fusion. TorchInductor freezing is enabled for implementations without an explicit deployment interface. Each model is compiled with `torch.compile(mode="max-autotune", fullgraph=True)` and executed through a CUDA Graph. The `fullgraph=True` constraint prevents graph breaks or device synchronizations from silently causing a fallback to eager execution.

Table 26: **FP16 TensorRT deployment on DRIVE AGX Thor at batch size eight.** Cosine similarity compares FP16 TensorRT outputs with FP32 PyTorch reference outputs.

Task	Model	Input	Median (ms) ↓	Images/s ↑	Cosine sim. ↑
Detection	GTR-S	640^2	14.017	571	0.9998
	GTR-M	640^2	19.126	418	0.9998
	GTR-L	640^2	27.817	288	0.9999
	GTR-X	640^2	32.701	245	0.9998
Instance seg.	GTR-Seg-S	640^2	22.085	362	0.9993
	GTR-Seg-M	640^2	31.529	254	0.9996
	GTR-Seg-L	640^2	40.203	199	0.9990
	GTR-Seg-X	640^2	45.142	177	0.9990
Oriented det.	GTR-OBB-S	1024^2	39.619	202	0.9998
	GTR-OBB-M	1024^2	54.312	147	0.9998
	GTR-OBB-L	1024^2	74.545	107	0.9998
	GTR-OBB-X	1024^2	85.553	94	0.9998
Pose	GTR-Pose-S	640^2	16.227	493	0.9993
	GTR-Pose-M	640^2	23.108	346	0.9995
	GTR-Pose-L	640^2	31.877	251	0.9993
	GTR-Pose-X	640^2	37.228	215	0.9994
Semantic seg.	GTR-SemSeg-S	1024^2	36.953	216	0.9991
	GTR-SemSeg-M	1024^2	50.860	157	0.9998
	GTR-SemSeg-L	1024^2	70.900	113	0.9997
	GTR-SemSeg-X	1024^2	81.234	98	0.9994
Depth	GTR-Depth-S	640^2	19.212	416	1.0000
	GTR-Depth-M	640^2	23.953	334	1.0000
	GTR-Depth-L	640^2	32.588	245	1.0000
	GTR-Depth-X	640^2	37.328	214	1.0000

Profiler traces confirm that each steady-state forward pass incurs exactly one CUDA Graph launch, without GPU-to-CPU scalar synchronization or device-to-host transfer. The official RF-DETR (Robinson et al., 2026) and LW-DETR (Chen et al., 2024) implementations use GPU-resident shape scalars and tensor assertions that prevent full-graph compilation. We replace these operations with compile-time shape constants, producing bitwise-identical outputs. Compilation, autotuning, graph capture, and warm-up are excluded from all reported measurements.

E.2.2 Timed Scope and Post-processing

For DETR-style models, we measure only the forward pass and exclude output decoding and post-processing. For YOLOv9 (Wang et al., 2024b), YOLO11 (Jocher, 2024), and YOLOv12 (Tian et al., 2025), the reported latency includes both the forward pass and CUDA NMS implemented with `torchvision.ops.nms`. We use `conf=0.001`, `iou=0.7`, `multi_label=True`, `agnostic=False`, and `max_det=300`.

Because NMS cost depends on the number of candidate boxes, these models are evaluated over all 5,000 images in COCO (Lin et al., 2014) `val2017`. YOLOv10 (Wang et al., 2024a) and YOLO26 (Jocher et al., 2026) perform in-network top-k selection and require no separate NMS stage. All NMS-free detectors are evaluated on a fixed subset of 1,000 COCO images.

E.2.3 Warm-up and Statistics

YOLO models are warmed up for 200 forward iterations. YOLO11 and YOLOv12 receive an additional 30 warm-up iterations covering the complete prediction and NMS pipeline. YOLOv9 uses no separate NMS warm-up; its one-time NMS initialization cost is negligible when averaged over 5,000 images. All other detection and segmentation models are warmed up for 300 iterations.

Latency is measured using CUDA events on the default stream while the GPU is otherwise idle. Each model is benchmarked once in an independent process. Table 7 reports the minimum, arithmetic mean, and median per-image latency. We use the median for every model comparison.

E.3 CUSTOM CUDA OPERATOR

E.3.1 IMPLEMENTATION DETAILS

Scope and decomposition. The reference FLA implementation (Yang & Zhang, 2024) targets general training and inference settings, supporting variable head dimensions, sequence lengths, recurrent states, backward propagation, and final-state outputs. In contrast, our CUDA operator is specialized for GTR inference with batch size one, FP16 inputs, a zero initial state, no backward propagation or final-state output, chunk size $C = 64$, and per-head dimensions $d_k = 32$ and $d_v = 64$.

As described in Section 2.1, GTR fixes the value-side state-decay gate to $\boldsymbol{\beta}_t \equiv \mathbf{1}_{1\times d_v}$ during both training and inference. Consequently, the training-time FLA kernels and our custom operator consume identical inputs—$\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$, and the learned key-side log-decay gate $\boldsymbol{g}$—and evaluate the same recurrence. The data-dependent output gate is likewise preserved and may be fused into the deployment epilogue. Up to mixed-precision roundoff, our operator therefore changes only the execution schedule and fusion pattern, not the underlying computation.

A remaining inefficiency in FLA v0.5.0 is that each state-tile program processes chunks sequentially, performing a matrix contraction at every step and exposing limited parallelism in our small-batch regime. Let $N_c = \lceil L/C \rceil$ denote the number of chunks. We instead parallelize the expensive chunk summaries $\mathbf{U}_c$ in Equation 3 over the chunk–head grid, leaving only the lightweight boundary-state recurrence as a sequential scan; adjacent lanes process adjacent elements along the contiguous d_v dimension, enabling coalesced memory accesses. While a tree-based prefix scan achieves $\mathcal{O}(\log N_c)$ depth, it requires additional global-memory passes and strided accesses in our setting. We find the coalesced linear scan faster in the target deployment regime while still retaining $\mathcal{O}(L)$ work for fixed head dimensions and chunk size.

Fused tensor-core dataflow. Building on this schedule, our first kernel fuses chunk-local cumulative sums with the construction of $\mathbf{U}_c$. Prefix sums are computed in FP32 registers, while value tiles are staged in padded shared memory via `cp.async`. The fixed configuration $C = 64$, $d_k = 32$, $d_v = 64$ maps efficiently to `ldmatrix` loads and tensor-core MMA instructions with FP32 accumulation: chunk-summary and score–value contractions use FP16 operands, whereas the decay-rescaled query/key operands and boundary states use BF16 to accommodate their wider dynamic range. Kernel tiling and dispatch parameters are tuned for RTX 4090 `sm_89`.

For each chunk–head pair, a second kernel jointly computes the causal $\widetilde{\mathbf{Q}}\widetilde{\mathbf{K}}^\top$ scores, their product with $\mathbf{V}$, and the boundary-state readout within a single thread block. The resulting 64×64 causal score tile is consumed directly from shared memory rather than materialized to global memory, eliminating the standalone cumulative-sum launch and an intermediate intra-chunk score tensor of size $\mathcal{O}(LHC)$. Chunk-local cumulative sums are still exchanged between the summary and output kernels through an FP16 workspace.

Length-aware scheduling and deployment fusion. Because the relative cost of the scan and output stages depends on N_c, we adopt a length-aware dispatch policy. For short sequences, a fallback output kernel reconstructs chunk prefixes inline, avoiding a separate scan launch; under our default RTX 4090 dispatch policy, this path is used for $N_c < 6$. For eligible non-wave-aligned grids with $N_c \geq 6$, an occupancy-aware kernel instead combines the scan and output stages: output blocks compute the chunk-local contractions while scan blocks generate boundary states, with synchronization occurring only immediately before the boundary-state readout so that the two stages can overlap. All other configurations retain a separate scan kernel.

Beyond kernel scheduling, the deployment interface supports persistent caller-managed workspaces, preallocated output buffers, and CUDA Graph capture. In the end-to-end GTR implementation, the output epilogue further fuses RMSNorm and the SiLU-activated output gate—distinct from the subsequent S-SwiGLU channel mixer—directly into the GLA kernel, removing one kernel launch and its intermediate global-memory round trip. This epilogue fusion is excluded from the isolated operator measurements in Figure 4.



Figure 8: **Qualitative object detection results of GTR-X.**

F Qualitative Results

We present qualitative results for object detection, instance segmentation, human pose estimation, semantic segmentation, zero-shot monocular depth estimation, and oriented object detection. The COCO examples include occlusion, low illumination, motion blur, defocus, object rotation, and crowded scenes. The additional transfer examples use Cityscapes (Cordts et al., 2016) for semantic segmentation, the DTU Robot Image Data Set (Jensen et al., 2014) for indoor depth, nuScenes (Caesar et al., 2020) for outdoor depth, and the DOTA-v1.0 (Xia et al., 2018) test set for oriented object detection.

F.1 Object Detection Results

Detection behavior. Figure 8 illustrates selected GTR-X predictions under crowding and occlusion, motion blur, object rotation, and low illumination. Across these examples, the model localizes small and overlapping instances despite substantial appearance degradation.

F.2 Instance Segmentation Results

Mask quality. Figure 9 visualizes GTR-X masks on the same challenging images as Figure 8. The selected predictions follow visible object contours and separate adjacent instances; together with Table 8, they illustrate the transfer of the output-aligned representation to instance segmentation.

F.3 Human Pose Estimation Results

Pose behavior. The selected cases in Figure 10 show GTR-X predicting articulated poses for multiple overlapping people under crowding, low illumination, and motion blur.

F.4 Semantic Segmentation Results

Dense semantic predictions. Figure 11 compares GTR-X predictions with Cityscapes ground truth.



Figure 9: **Qualitative instance segmentation results of GTR-X.** Predictions are visualized on the same image set used in Figure 8.

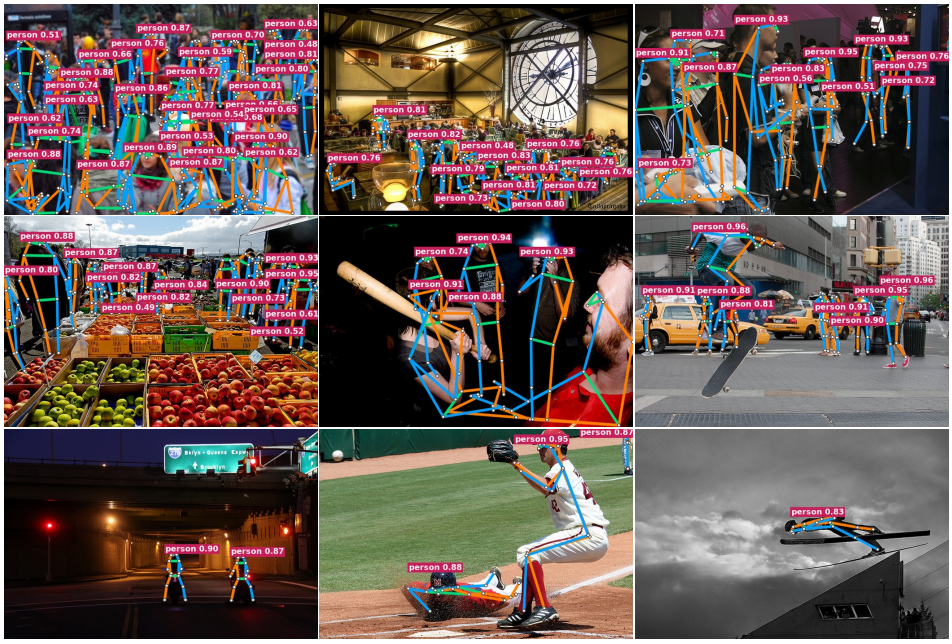

Figure 10: **Qualitative human pose estimation results of GTR-X.** The examples emphasize crowded scenes, low illumination, and motion blur.

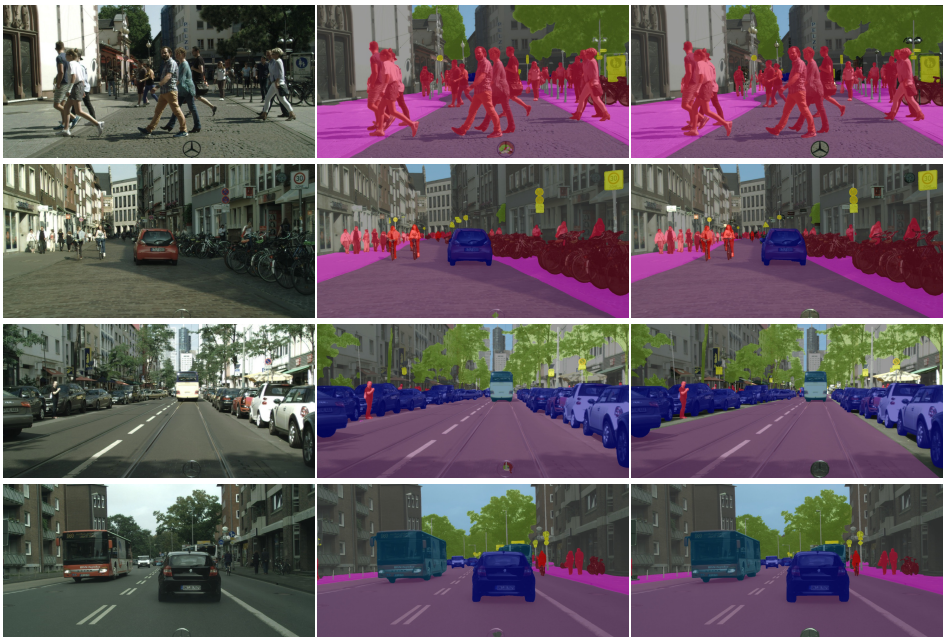

Figure 11: **Qualitative semantic segmentation results of GTR-X.** Each row shows, from left to right, the input RGB image, GTR-X prediction, and ground-truth annotation.

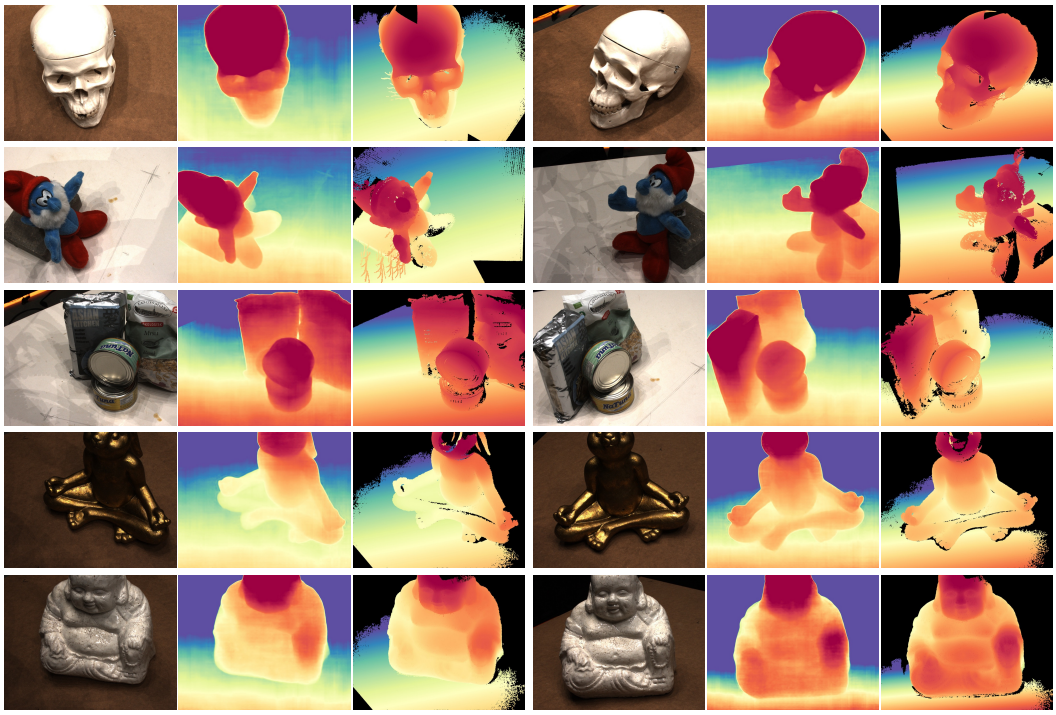

Figure 12: **Indoor monocular depth predictions of GTR-L on the DTU Robot Image Data Set (Jensen et al., 2014).** Each three-column group shows the input RGB image, GTR-L prediction, and reference depth obtained by projecting the calibrated DTU point cloud. Black regions in the reference maps contain no valid projected depth.

F.5 Zero-Shot Monocular Depth Estimation Results

We apply GTR-L without dataset-specific fine-tuning to indoor DTU and outdoor nuScenes imagery.

F.5.1 Indoor DTU Results

Projected reference depth. For these examples, DTU provides calibrated point-cloud geometry rather than dense per-view depth maps. We therefore construct reference depth by projecting the point cloud through each camera using the provided intrinsics and calibrated pose. The resulting reference maps cover 67.9% of pixels on average; missing support is concentrated in the background fabric, which was not captured by structured light, and on specular surfaces. Figure 12 compares the RGB inputs, GTR-L predictions, and projected references.

Depth-based reconstruction. Figure 13 visualizes reconstructions obtained from all 49 calibrated views of each DTU object. Since a single image does not geometrically determine absolute depth scale, depth maps predicted independently across views may exhibit view-dependent scale drift. Prior to fusion, we therefore align the scale of each predicted depth map using a single scalar, computed as the median ratio between the projected structured-light reference depth and the prediction over valid pixels. This per-view rescaling preserves the relative depth structure within each view while correcting only its global scale.

We then back-project the rescaled depth maps using the calibrated camera intrinsics and extrinsics, fuse all 49 views into a TSDF (Curless & Levoy, 1996) volume with 2-mm voxels, and render the resulting surface as colored points. Because the structured-light reference supplies one global scale factor per view, these reconstructions illustrate the geometry after reference-assisted scale correction.

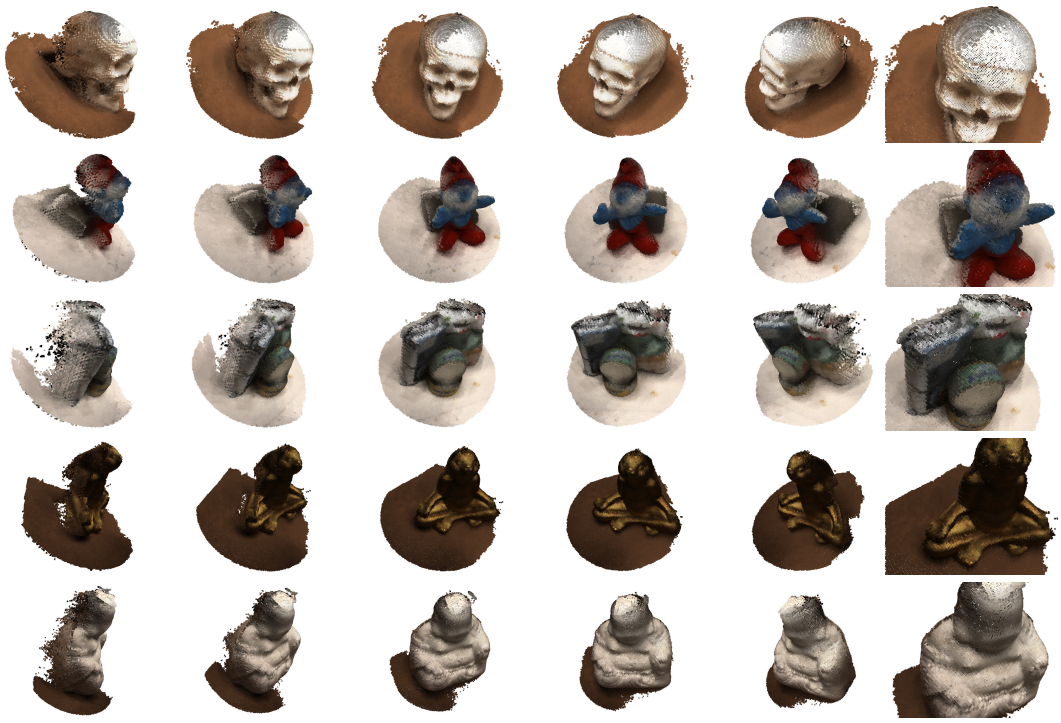

Figure 13: **Colored point clouds reconstructed from zero-shot GTR-L depth predictions on DTU (Jensen et al., 2014).** Before fusion, each view is scale-aligned using the projected structured-light reference.

F.5.2 Outdoor nuScenes Results

Surround-view depth. Figures 14–16 show GTR-L predictions for the six synchronized surround-view cameras in three nuScenes scenes. Each figure preserves the camera ordering between the RGB grid and the corresponding depth grid, covering daytime, nighttime, and roadside construction imagery without nuScenes-specific fine-tuning.

Temporally fused local maps. To visualize multi-frame geometric consistency, for each reference time t, we aggregate all available frames within $[t - 4\,\mathrm{s},\ t + 4\,\mathrm{s}]$. For each frame, we scale-align the GTR-L depth prediction using a single scalar estimated from the corresponding LiDAR measurements, back-project the corrected depth with the camera intrinsics, and transform the resulting points into a common coordinate frame using the calibrated pose. We then fuse all observations with TSDF (Curless & Levoy, 1996) to obtain a local map. Figures 17 and 18 show two examples from perspective and bird's-eye views. Since LiDAR is used only for per-frame scale alignment, these results qualitatively illustrate geometric consistency after scale correction.

F.6 Oriented Object Detection Results

Oriented localization. Figure 19 shows the GTR-X predictions in the DOTA-v1.0 test images. The oriented boxes track ships and vehicles at various angles and remain separated in densely packed harbor and road scenes, complementing the quantitative results in Table 10.

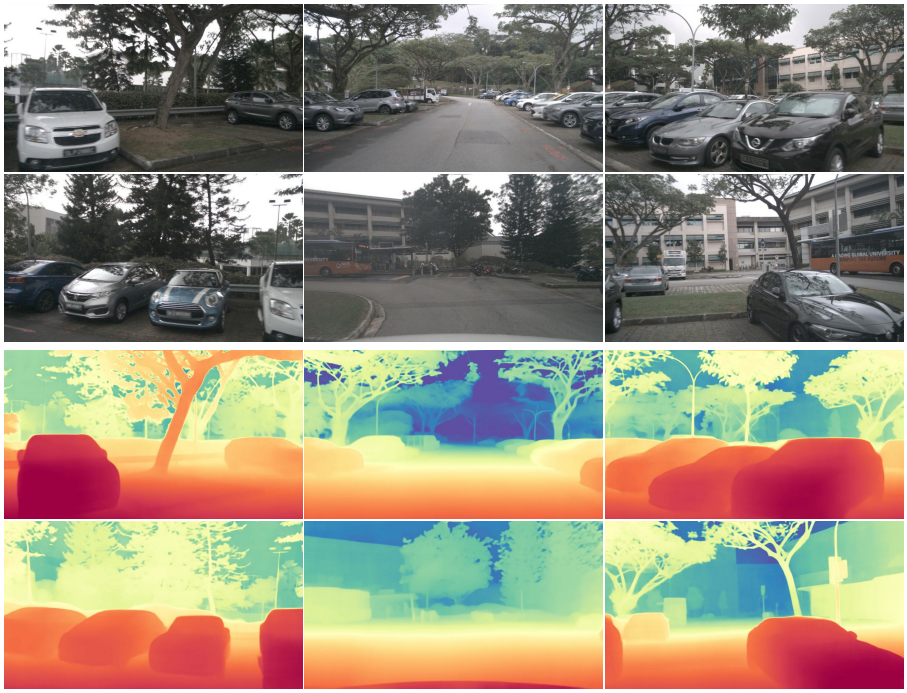

Figure 14: **Outdoor monocular depth predictions of GTR-L on nuScenes (Caesar et al., 2020) (set 1).** The first two rows show the six surround-view RGB images, and the last two rows show the corresponding predictions in the same row-major order.



Figure 15: **Outdoor monocular depth predictions of GTR-L on nuScenes (Caesar et al., 2020) (set 2).** RGB inputs occupy the first two rows and their corresponding predictions occupy the last two rows in the same row-major order.



Figure 16: **Outdoor monocular depth predictions of GTR-L on nuScenes (Caesar et al., 2020) (set 3).** The layout follows Figure 14: six RGB inputs are followed by their six corresponding depth predictions.

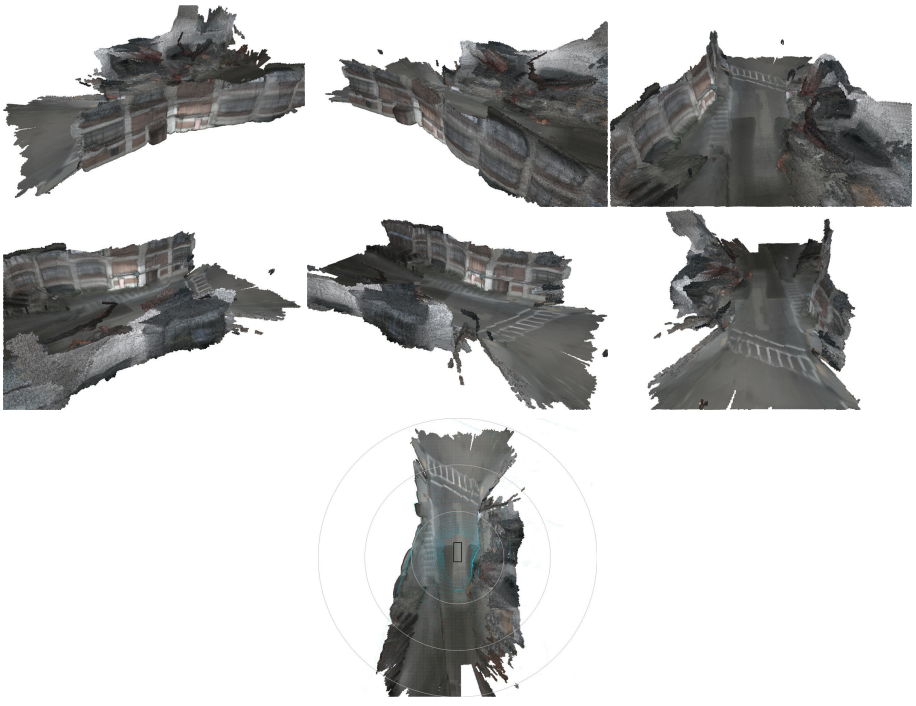

Figure 17: **Temporally fused local reconstruction on nuScenes (Caesar et al., 2020) (scene 1).** Each frame uses LiDAR-assisted scale alignment before TSDF fusion.

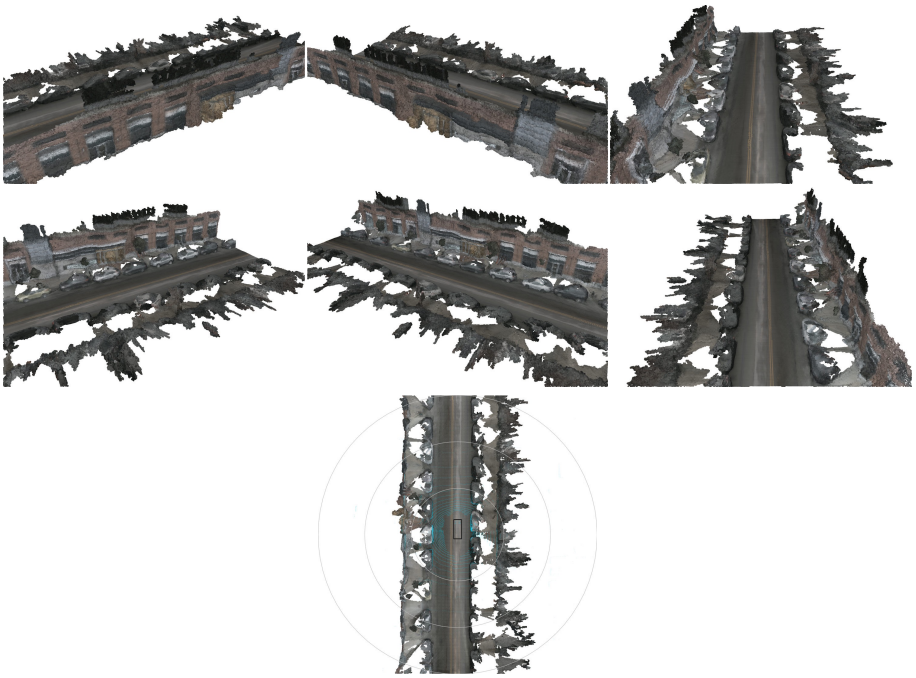

Figure 18: **Temporally fused local reconstruction on nuScenes (Caesar et al., 2020) (scene 2).** Each frame uses LiDAR-assisted scale alignment before TSDF fusion.

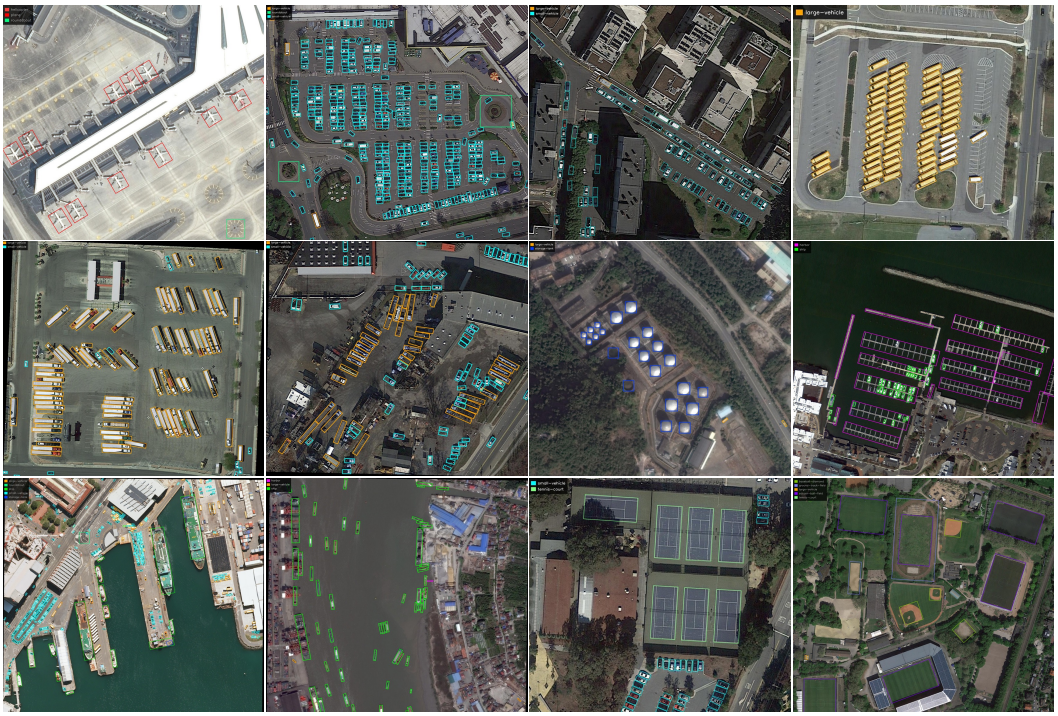

Figure 19: **Qualitative oriented object detection results on the DOTA-v1.0 (Xia et al., 2018) test set using GTR-X.**