Do We Really Need Diffusion for Generative Object Detection? A Minimal Prototype Perspective

Yu Hong[1,2] **Xiaosong Jia**[3] **Yihan Wang**[3]
Wenlong Liao[4] **Tao He**[4] **Junchi Yan**[1,2,✉]

[1]Shanghai Jiao Tong University [2]Shanghai Innovation Institute
[3]Fudan University [4]COWAROBOT [✉]Corresponding author

Abstract

Generative object detection, exemplified by DiffusionDet, combines two appealing properties: (P1) data-independent random-box initialization with a learned noise-to-data transport, and (P2) test-time scaling (TTS), the ability to trade inference compute for accuracy at a fixed checkpoint by adjusting the number of proposals or refinement steps. This paper asks: *How can we preserve both properties with a minimal conceptual design?* We dissect DiffusionDet component by component and find that, in our controlled interventions, the diffusion schedule, the initial noise distribution, and DDIM sampling stochasticity have only a small effect on inference results. In contrast, detection performance is more sensitive to the denoising head, iterative refinement, and proposal coverage. Based on these diagnostic results, we propose **LinearDet**, a compact generative detection prototype: *linear noising at training, linear denoising at inference*—no diffusion-specific schedule, no diffusion ODE/SDE solver, only a symmetric pair of linear mixtures. LinearDet matches DiffusionDet under standard COCO settings, achieves higher overall AP under a very limited proposal budget, improves mAP in zero-shot CrowdHuman transfer, and produces smoother refinement trajectories that admit clean heatmap visualization. The same linear replacement also preserves accuracy in a DINO-based generative detector and in diffusion-based semantic segmentation. More broadly, we hope this diagnosis and compact prototype offer the object detection community a cleaner baseline for studying generative detection.

1 Introduction

Object detection has long been formulated as a structured regression problem, evolving from region-based two-stage pipelines [4, 13, 14, 18, 49] and one-stage dense detectors [29, 34, 46–48, 58, 59, 61, 63] to modern end-to-end set prediction frameworks such as DETR and its variants [5, 24, 32, 39, 60, 65], with large-scale pretraining also enabling open-vocabulary detection beyond fixed label sets [21, 26, 33, 41, 45, 64]. In parallel, *generative object detection* has emerged as an alternative view: a detector generates boxes through iterative refinement rather than a single feed-forward pass. DiffusionDet [6], the representative instance, reinterprets detection as a denoising trajectory from Gaussian noise to ground-truth boxes and brings in two properties that distinguish it from conventional detectors: **(P1)** data-independent random-box initialization with learned noise-to-data transport, and **(P2)** test-time scaling (TTS).

This paper asks: *How can we preserve both P1+P2 with a minimal conceptual design?* We answer this by dissecting DiffusionDet at inference and asking which parts actually affect performance. In our controlled interventions, the diffusion schedule, initial noise distribution, and DDIM sampling stochasticity have only a small effect on inference results, while the factors that most consistently affect AP are a learned denoiser, iterative refinement, and sufficient proposal coverage.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Based on these findings, we propose **LinearDet**, a compact generative detection prototype that also tests the same minimal design from the training side: starting from scratch, it replaces DiffusionDet's diffusion noising/sampling machinery with a symmetric linear noising/denoising path while keeping the architecture and recipe matched. We adopt the linear probability path as the simplest continuous interpolation between noise and data, so that training and inference share a single operation—linear mixture:

- *Training*: $x_t = (1-t)x_0 + t\,x_1$, a linear mixture of clean boxes x_0 and random noise x_1, where $t \in [0,1]$ is the noise level (t=0: data; t=1: noise).
- *Inference*: $x_{t_{\text{next}}} = \alpha\, x_{t_{\text{curr}}} + (1-\alpha)\,\hat{x}_0$ with $\alpha = t_{\text{next}}/t_{\text{curr}}$, a linear mixture of the current noisy state and the predicted clean boxes $\hat{x}_0$, where $t_{\text{curr}} > t_{\text{next}}$ are the current and next noise levels.

No diffusion-specific schedule, no diffusion ODE/SDE solver—only a symmetric pair of linear mixtures. On MS COCO, LinearDet matches DiffusionDet under standard settings, transfers more favorably from COCO to CrowdHuman, and remains stronger under a very limited proposal budget. By removing the stochasticity of diffusion sampling, LinearDet also exposes behaviors that are obscured in the diffusion formulation: refinement trajectories are noticeably smoother and more predictable across runs, which in turn enables aggregating them into clean spatial heatmaps. The same replacement carries over to DiffuDINO [42], a DINO-based generative detector, and to DDP [20] for semantic segmentation, suggesting that the diagnosis is not tied to a single architecture. Our contributions are threefold:

- **Component-wise diagnosis of DiffusionDet.** We probe DiffusionDet's components and find that the diffusion schedule, initial noise distribution, and DDIM sampling stochasticity have only a small effect in our controlled interventions, whereas the learned denoiser, iterative refinement, and sufficient coverage consistently affect detection. Further analysis suggests that proposal renewal and cross-step ensembling, rather than sampling noise, supply the exploration that refinement relies on.
- **LinearDet: a compact prototype.** Informed by the diagnosis above, we instantiate this design with a simple construction: *linear noising at training, linear denoising at inference*. LinearDet serves as a prototype model for the generative object detection community.
- **Empirical validation of the compact prototype.** On COCO, LinearDet matches DiffusionDet under standard proposal budgets and outperforms it under a very limited proposal budget and in zero-shot transfer, while also producing smoother refinement trajectories that enable clean heatmap aggregation. Matched linear replacements in DiffuDINO and DDP provide further support across detector architectures and prediction tasks.

Scope. LinearDet is a prototype within the generative detection family defined by P1+P2. It shares DiffusionDet's architecture and per-step FLOPs under matched settings, so its contribution is **a conceptual simplification rather than a computational one**. Our performance reference point is DiffusionDet; discriminative detectors such as DINO or DN-DETR improve detection along orthogonal axes (query denoising, deformable attention) and address a different research question from the compact-prototype question studied here.

2 Dissecting DiffusionDet Components

2.1 Related Work and Setup

Discriminative detection. Object detection has evolved from two-stage region-based pipelines [4, 13, 14, 18, 28, 49] and one-stage dense detectors [3, 12, 29, 34, 46–48, 58, 59, 61, 63] to end-to-end set-prediction transformers [5, 24, 32, 39, 57, 60, 65], supported by large-scale benchmarks and backbones [9–11, 17, 27, 36]. Open-vocabulary and foundation-model-based detectors [1, 15, 21, 23, 25, 26, 31, 33, 41, 44, 45, 62, 64] extend detection beyond fixed label sets.

Generative object detection. Denoising diffusion models [19, 22, 43, 51, 53, 55, 56] define a forward noising process $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\mathbf{z}_{t-1}, \beta_t\mathbf{I})$ and learn a reverse denoising process, with accelerated samplers [37, 54] for efficient inference. DiffusionDet [6] adapts this

Table 1: **Component diagnosis of DiffusionDet.** The table summarizes the tested components, their AP effect, and the resulting diagnosis. COCO val2017, inference only; DiffusionDet baseline AP is 0.465 with Normal schedule, Gaussian init, η=1, N_{step}=4, N_{box}=500, renewal on. Δ denotes the AP range (max−min, in AP points) across the listed variants together with the baseline, except for the Faster R-CNN iteration row, where Δ reports the best/worst deviation relative to its own 1-step reference. Full metrics are reported in Appendix Table 6. Results are averaged from 3 different random inference seeds.

Component	Intervention	AP effect	Diagnosis
Diffusion-specific inference choices			
Timestep schedule $\{t_k\}$	Random / reverse / shuffled / constant	0.464–0.465 ($\Delta = 0.1$)	Low sensitivity
Initial distribution shape	Uniform / grid	0.462–0.465 ($\Delta = 0.3$)	Low sensitivity
DDIM stochasticity η	$0.0/0.5/2.0$	0.465–0.465 ($\Delta = 0.0$)	Low sensitivity
AP-sensitive ingredients			
Learned denoiser	Iterate non-denoiser head	0.413–0.422 ($\Delta = 0.4/-0.5$)	Denoiser needed for stable gains
Iterative refinement N_{step}	$1/2/4/8$ steps at N_{box}=500	0.457–0.466 ($\Delta = 0.9$)	AP-sensitive P2 axis
Proposal coverage N_{box}	50/100/300/500 boxes	0.411–0.465 ($\Delta = 5.4$)	AP-sensitive P2 axis

paradigm to object detection by treating a set of bounding boxes $\mathbf{b} \in \mathbb{R}^{N\times 4}$ as the "data" to generate: training perturbs ground-truth boxes, the model learns to denoise them, and inference iteratively refines a set of random noise boxes. Its two defining properties are **P1** (data-independent random-box initialization with learned noise-to-data transport) and **P2** (test-time scaling: varying proposal count or refinement steps at inference without retraining). Subsequent work further extends this generative formulation along multiple axes, including task generalization, box encoding, and transport dynamics [7, 16, 38, 40, 50]; contemporaneously, FlowDet [2] studies generative detection from a Conditional Flow Matching perspective.

Diagnostic scope. We probe six operational components of DiffusionDet: three diffusion-specific inference choices—timestep schedule, initial distribution shape, and DDIM stochasticity—and three candidate AP-sensitive ingredients—a learned denoiser, iterative refinement, and proposal coverage. Throughout, we use a single DiffusionDet model reproduced with the official implementation under its default recipe. Results are averaged from 3 different random inference seeds.

2.2 Low-Sensitivity Inference-Time Components

We first ask how sensitive DiffusionDet is to its diffusion-specific inference components. Using this reproduced DiffusionDet model and keeping all other settings matched, we individually disrupt three axes of DiffusionDet's inference procedure:

1. **Timestep schedule.** The default monotone schedule t=1 $\rightarrow$ t=0 is replaced by: *Random* ($t \sim \mathcal{U}(0,1)$ sampled independently per step), *Reverse*, *Shuffled* (the normal timesteps in a random permutation), and *Constant* (t=0.5 at every step).
2. **Initial proposal distribution.** The default $\mathcal{N}(0, I)$ is replaced by *Uniform* ($\mathcal{U}(-3,3)$, comparable support) or *Grid* (a fixed regular grid of box centers with a small uniform perturbation, removing both Gaussian shape and per-image randomness).
3. **DDIM sampling stochasticity.** We sweep the DDIM η parameter across $\{0, 0.5, 1, 2\}$, where η=0 is fully deterministic, η=1 is DiffusionDet's default, and larger η injects progressively more stochastic noise.

All 9 diffusion-specific variants in Table 1 land within 0.3 AP of the baseline. Under this operational criterion, the exact inference-time timestep schedule, the initial distribution shape, and DDIM sampling stochasticity show low sensitivity in our inference-time interventions: collapsing the schedule to a single fixed t=0.5 produces zero AP change, the DDIM η sweep is identical to three decimals, and Grid init still matches the Gaussian baseline within 0.1 AP.

2.3 AP-Sensitive Components in DiffusionDet

As a counterpart to §2.2, we now identify the variables that move AP by more than our 0.5 threshold when perturbed. Three emerge: iteration count N_{step}, proposal coverage N_{box} (together with the box-renewal mechanism that maintains it), and the denoising training of the head itself.

Iteration and proposal coverage. P2 (TTS) refers to scaling inference compute, and the two direct allocation axes in this setting are N_{step} and N_{box}. Table 1 sweeps both axes on the same DiffusionDet model and Normal schedule.

In sharp contrast with the low-sensitivity rows, the AP-sensitive rows reveal a very different profile. Increasing N_{step} from 1 to 8 at N_{box}=500 yields a monotone gain (AP $45.7 \rightarrow 46.6$), showing that iterative refinement keeps producing residual improvements even when proposals are abundant. Reducing N_{box} from 500 to 50 at N_{step}=4 instead causes a substantial drop of 5.4 AP ($46.5 \rightarrow 41.1$), with proportionally larger losses on small/medium objects. Box renewal further acts within this iteration-and-coverage axis: it drops low-confidence boxes between refinement steps and replenishes them with fresh random noise, preventing dead boxes from accumulating across iterations. Disabling renewal drops AP from 0.465 to 0.457 (-0.8), a noticeably larger effect than any schedule/init/η variant. **The two axes also interact**: under tight proposal budgets, additional refinement steps recover much of the lost accuracy, whereas at N_{box}=500 a single step already saturates—the resulting $N_{\text{step}} \times N_{\text{box}}$ trade-off is precisely the TTS (P2) behavior we exploit in §4.3.

Denoising training of the head. Iteration and coverage are AP-sensitive in these interventions—but is iteration alone sufficient to produce gains, or must the head itself be trained specifically as a denoiser? We test this by replacing the diffusion head with standard Faster R-CNN [49] head, which is trained in a one-shot regression setting rather than a denoising setting. We then apply the same style of iteration as DiffusionDet: the boxes predicted by the head at step k are fed back in place of the RPN proposals at step $k+1$. The results are reported in Table 1.

AP changes non-monotonically across iterations: the best case occurs at 2 steps (+0.4 AP relative to 1 step), while the 8-step run falls by 0.5 AP relative to 1 step (full metrics in Appendix Table 6). A head trained in the one-shot regression setting does not show a stable monotone improvement from self-feeding. This supports the idea that denoising training is important for making iterative refinement stable and consistently useful, even though repeated looping by itself can give only limited gains or even worse results.

Why the diffusion-specific choices matter little. The two groups are linked by a simple mechanism, which we instrument on the same model (Appendix F). First, the learned denoiser quickly attenuates the initial state: across proposal priors, the initial recall of ground-truth boxes spans 5.32 points, but only 0.08 points after a single refinement step. Second, exploration is already supplied by proposal renewal and cross-step ensembling: with renewal on, about 487 of 500 proposals are replaced at each early step. In a full $\eta \times$renewal$\times$ensembling factorial, DDIM stochasticity changes AP by at most 0.8, and its largest effect, where stochastic sampling is worse, occurs only when renewal and ensembling are both disabled. Renewal and ensembling are largely interchangeable: with either one enabled, the other adds only about 0.7 AP, but with the other disabled, either mechanism can contribute up to 4.1 AP. We read this as an empirical account of the diagnosis rather than a formal theory.

Takeaway. Applying the $\Delta\text{AP} < 0.5$ threshold under the default renewal and ensembling setting, §2.2 and §2.3 separate the tested variables into low-sensitivity and AP-sensitive groups: exact inference-time timestep schedule, initial noise shape, and DDIM stochasticity show little effect, while iteration count, proposal coverage, and denoising head training consistently affect AP. We use the AP-sensitive group as the design target for a compact prototype: a learned denoiser, iterative refinement, and sufficient proposal coverage. This inference-time diagnosis identifies the components a minimal replacement should preserve. §3 and §4 then provide the complementary training-side replacement test by training LinearDet from scratch with a linear noising/denoising construction under the same architecture and recipe.

3 LinearDet

3.1 From the Diagnostic Set to LinearDet

§2 highlighted three AP-sensitive ingredients for generative object detection: (1) a learned box denoiser, (2) iterative refinement, and (3) sufficient proposal coverage. LinearDet instantiates this set with a simple construction, while keeping proposal renewal as the same coverage-maintenance mechanism used by DiffusionDet.

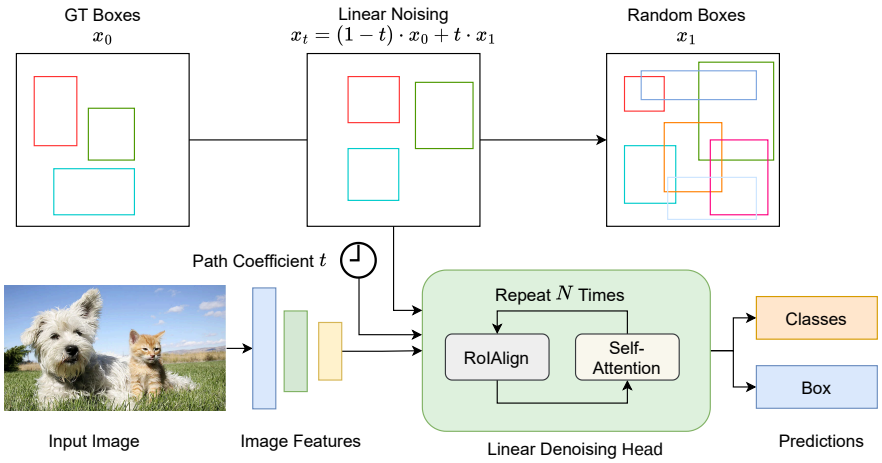

Figure 1: Architecture of the LinearDet prototype. It consists of (1) an image backbone producing multi-scale features, (2) a linear noising mechanism that constructs noisy boxes as a linear mixture of clean boxes and random noise during training, and (3) a Linear Denoising Head that predicts classification logits and a clean-box estimate conditioned on boxes, image features, and the path coefficient t.

Design motivation. The diagnosis in §2 shows that, under the tested inference settings, detection accuracy is largely insensitive to the timestep schedule, initial noise distribution, and DDIM stochasticity. This motivates a training-side hypothesis: that diffusion-specific noising and sampling can also be replaced when a detector is trained from scratch. We test it under matched architecture and training recipe by adopting the simplest remaining transport—linear interpolation between clean boxes and noise for training noising, and the corresponding linear mixture of the current state and the predicted clean boxes for inference refinement—while retaining the path coefficient t to preserve DiffusionDet's head interface. This transport is an endpoint-parameterized special case of Flow Matching [30], closely related to rectified flow [35]; unlike standard Flow Matching, however, the predicted endpoint is trained with DiffusionDet's set-prediction objective rather than a pointwise regression loss. The closely related FlowDet [2] arrives at a linear-transport design from the Conditional Flow Matching perspective. Note that, although both works use a linear transport path, their core questions, experimental scope, and conclusions differ. LinearDet is *subtractive*: it derives a minimal prototype by removing low-sensitivity components from DiffusionDet and tests whether the resulting model preserves P1/P2 behavior under matched architecture and recipe. FlowDet instead starts from CFM and explores a broader transport-design space with choices such as priors, source–target couplings, and ODE solvers.

Figure 1 summarizes the resulting architecture, which consists of three modules: **Image backbone.** An image encoder (backbone + feature pyramid) extracts multi-scale feature maps $\mathcal{F}$ from the input image. **Linear noising.** At training time, noisy boxes are constructed as a linear mixture of clean boxes and random noise, with the mixture coefficient sampled per step so the head sees the full range of noise levels. **Linear Denoising Head.** Following DiffusionDet's head interface, the head receives the current noisy boxes, the path coefficient t, and RoI features extracted from $\mathcal{F}$, then outputs classification logits and a clean-box estimate that drives refinement.

Training and inference details are provided in §3.2.

3.2 Training and Inference

This section details how we construct training samples (linear noising), optimize the Linear Denoising Head, and perform linear denoising at test time.

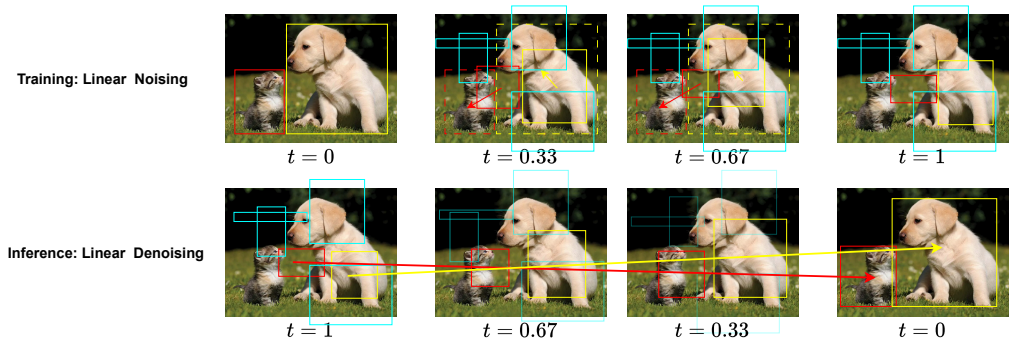

Figure 2: **LinearDet's symmetric pair of linear mixtures.** Yellow/red boxes denote positive samples; blue boxes denote negatives; -> arrows visualize the implied $x_t \to \hat{x}_0$ direction.

3.2.1 Training

Latent box space. We represent boxes in normalized cxcywh $\in [0,1]^4$, map them to a scaled latent space, and clamp to a bounded range for numerical stability. During training, each image uses a fixed number of proposals; ground-truth boxes are padded/truncated to match this count.

Linear noising. For each image, we sample a continuous time $t \sim \mathcal{U}(0,1)$ and Gaussian noise $x_1 \sim \mathcal{N}(0, I)$ in latent space. We encode the ground-truth boxes to latent variables x_0 and form the noisy boxes as a linear mixture:

$$x_t = (1-t)\, x_0 + t\, x_1. \tag{1}$$

This follows the convention that t=0 corresponds to data (x_0) and t=1 corresponds to noise (x_1).

Forward and targets. We decode x_t to pixel-space boxes for RoI feature extraction, and feed RoI features together with the path coefficient t into the Linear Denoising Head. The head predicts (i) classification logits and (ii) a clean-box estimate $\hat{b}_0$ in pixel-space xyxy, which is supervised via a set prediction loss against matched ground-truth boxes.

Losses. We supervise $\hat{b}_0$ with a set prediction objective (matching + classification + box regression). Since x_1 is fixed at sampling time and independent of model parameters, predicting $\hat{b}_0$ is algebraically equivalent, up to the corresponding latent encoding, to predicting the linear-path direction $u = x_0 - x_1$. We choose to regress $\hat{b}_0$ directly as it provides a target that does not depend on the noise sample and yields slightly better AP in our ablation (Appendix D.1).

3.2.2 Inference

Initialization and schedule. At test time, we initialize x as pure Gaussian noise in latent space and iteratively refine it along a monotone time grid from t=1 to t=0. We adopt these as default choices for simplicity and consistency with DiffusionDet; §2.2 indicates that the trained model is robust to the precise schedule and initial noise distribution.

Linear denoising update. At a current time t_{curr}, we decode the current latent boxes to pixel-space boxes, extract RoI features, and predict $\hat{b}_0$ in pixel-space xyxy. We then re-encode $\hat{b}_0$ to the same latent box space as $x_{t_{\text{curr}}}$, obtaining $\hat{x}_0$, and update the latent state to t_{next} as a linear mixture of the current state and the predicted clean boxes:

$$x_{t_{\text{next}}} = \frac{t_{\text{next}}}{t_{\text{curr}}} x_{t_{\text{curr}}} + \left(1 - \frac{t_{\text{next}}}{t_{\text{curr}}}\right) \hat{x}_0, \quad (t_{\text{curr}} > 0). \tag{2}$$

This is the unique linear interpolation between $x_{t_{\text{curr}}}$ and $\hat{x}_0$ satisfying two boundary conditions: the state is unchanged when $t_{\text{next}} = t_{\text{curr}}$, and reaches $\hat{x}_0$ exactly when $t_{\text{next}} = 0$. It is the symmetric counterpart of the linear noising in Eq. 1: training mixes clean boxes with noise, and inference mixes the current noisy latent state with the re-encoded clean-box estimate—both by linear interpolation (Figure 2). Written with the endpoint-derived velocity $v = (x_{t_{\text{curr}}} - \hat{x}_0)/t_{\text{curr}}$, Eq. 2 is exactly one Euler step from t_{curr} to t_{next}.

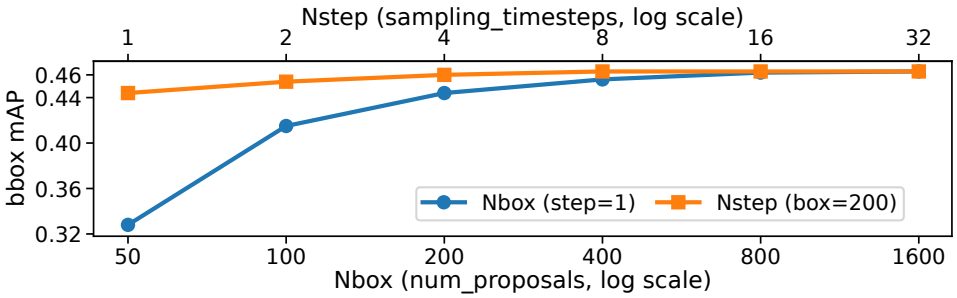

Figure 3: **LinearDet preserves the TTS property of generative detection.** At a fixed checkpoint, increasing N_{box} (with N_{step}=1, blue) or N_{step} (with N_{box}=200, orange) monotonically improves AP, with a clean fast-then-saturate shape along each axis.

Box renewal. As identified in §2.3, proposal renewal is part of how P2's coverage is maintained under tight per-iteration budgets. LinearDet inherits DiffusionDet's renewal scheme unchanged: after each refinement step, low-confidence boxes are dropped and replaced by fresh random samples. Intermediate predictions can additionally be ensembled, again following DiffusionDet.

Summary. LinearDet in one line: *linear noising at training, linear denoising at inference.* No diffusion-specific schedule, no diffusion ODE/SDE solver—only a symmetric pair of linear mixtures (Eq. 1 and Eq. 2), wrapped around a learned box denoiser with proposal renewal. Step-by-step pseudocode for both training (Algorithm 1) and inference (Algorithm 2) is provided in Appendix A.

4 Experiments

4.1 Experimental Setup

Dataset. We conduct main experiments on MS COCO [27] using the standard train/val splits and COCO-style AP metrics. We further report zero-shot transfer results on CrowdHuman [52] by evaluating COCO-trained models on CrowdHuman without fine-tuning.

Model. For fair comparison, we align LinearDet with the official DiffusionDet implementation and training recipe, and only replace the diffusion head with our `LinearDenoiserHead`.

Training schedule. We use an iteration-based training loop with 450,000 iterations for main results. We adopt the official DiffusionDet optimizer/scheduler settings: AdamW with learning rate 2.5×10^{-5} and weight decay 10^{-4}, gradient clipping with max-norm 1.0, and a linear warm-up for the first 1,000 iterations (start factor 0.01), followed by a multi-step decay with milestones at 350,000 and 420,000 iterations (gamma 0.1). Unless otherwise specified, the training batch size is 16. All models are trained on 8 NVIDIA RTX 4090 GPUs (24 GB each); a full 450k-iteration run takes roughly 24–48 hours depending on the backbone. For ablation studies, we use a shorter schedule for fast iteration: 28,125 iterations at batch size 64.

4.2 Main Results

Comparison on COCO. We compare LinearDet with DiffusionDet under matched settings, and include FlowDet [2] and GenDet [40] as recent generative-detection references, together with representative discriminative baselines, on COCO val2017. Table 2 summarizes the main results and separates conventional detectors from generative detectors. Under the standard setting (500 proposals with multi-step refinement), LinearDet matches DiffusionDet on ResNet-50 (both 0.464 AP) and achieves higher AP on ResNet-101 (0.476 vs. 0.471 AP). FlowDet results are taken from its paper, which reports a different operating point (100 proposals, 3 steps) and a different training recipe (about 532k vs. 450k iterations, 300 vs. 500 training proposals); this comparison is therefore not fully controlled.

Test-time scaling capability. Figure 3 reports LinearDet's test-time scaling curves under fixed N_{step} or fixed N_{box}. It highlights a practical test-time scaling capability of LinearDet: at inference, we can improve accuracy by increasing N_{box} and/or N_{step} without retraining. With N_{step}=1, increasing the

Table 2: **LinearDet matches DiffusionDet on COCO val2017** under standard settings (500 proposals, 4 refinement steps): this compact instantiation of the design target in §2 reaches DiffusionDet-level AP. * denotes our reproduction under matched training. Our DiffusionDet* and LinearDet results are averaged from 3 different random training seeds; other entries are taken from the respective papers.

Method	Backbone / setting	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Conventional detectors (R-50)							
RetinaNet [29]	R-50	0.387	0.580	0.415	0.233	0.423	0.503
Faster R-CNN [49]	R-50	0.402	0.610	0.438	0.242	0.435	0.520
DETR [5]	R-50	0.420	0.624	0.442	0.205	0.458	0.611
DAB-DETR [32]	R-50	0.422	0.625	0.449	0.216	0.455	0.604
Deformable DETR [65]	R-50	0.438	0.626	0.477	0.264	0.471	0.580
Cascade R-CNN [4]	R-50	0.443	0.622	0.480	0.266	0.477	0.577
DN-DETR [24]	R-50	0.444	0.635	0.482	0.223	0.486	0.644
Sparse R-CNN [57]	R-50	0.450	0.634	0.482	0.269	0.472	0.595
DINO [60]	R-50, 5-scale	0.494	0.669	0.538	0.323	0.525	0.639
Generative detectors (R-50)							
FlowDet [2]	R-50	0.455	0.641	0.492	0.279	0.484	**0.615**
DiffusionDet* [6]	R-50, 4 @ 500	$\mathbf{0.464}_{\pm 0.0006}$	**0.668**	**0.498**	0.295	0.487	**0.615**
LinearDet	R-50, 4 @ 500	$\mathbf{0.464}_{\pm 0.0014}$	**0.668**	**0.498**	**0.298**	**0.492**	0.614
Conventional detectors (R-101)							
RetinaNet [29]	R-101	0.404	0.602	0.432	0.240	0.443	0.522
Faster R-CNN [49]	R-101	0.420	0.625	0.459	0.252	0.456	0.546
DETR [5]	R-101	0.435	0.638	0.464	0.219	0.480	0.618
Cascade R-CNN [4]	R-101	0.455	0.637	0.499	0.276	0.492	0.591
Sparse R-CNN [57]	R-101	0.464	0.646	0.495	0.283	0.483	0.616
Generative detectors (R-101 / diffusion backbone)							
GenDet [40] (Feature-based)	Diffusion backbone	0.301	0.456	0.314	0.102	0.345	0.504
FlowDet [2]	R-101	0.461	0.651	0.500	0.287	0.495	0.626
GenDet [40] (Learning-based)	Diffusion backbone	0.464	0.642	0.505	0.277	0.496	0.632
DiffusionDet* [6]	R-101, 4 @ 500	$0.471_{\pm 0.0008}$	0.675	0.503	0.294	0.500	0.628
LinearDet	R-101, 4 @ 500	$\mathbf{0.476}_{\pm 0.0010}$	**0.680**	**0.512**	**0.298**	**0.506**	**0.637**

Table 3: **Zero-shot transfer from COCO to CrowdHuman** [52]: LinearDet outperforms DiffusionDet on mAP ($+0.3/+0.8/+0.8$ AP for $N_{\text{step}}=1/2/3$). JI and mMR values are close between the two methods. Both models use $N_{\text{box}}=500$ and are evaluated without fine-tuning. Results are averaged over 3 random inference seeds.

Method	N_{step}	mAP ↑	mMR ↓	JI ↑
DiffusionDet	1	$0.4974_{\pm 0.0005}$	0.8700	0.5133
DiffusionDet	2	$0.5121_{\pm 0.0003}$	**0.8691**	0.5184
DiffusionDet	3	$0.5182_{\pm 0.0002}$	**0.8679**	0.5156
LinearDet (Ours)	1	$\mathbf{0.5002}_{\pm 0.0003}$	**0.8676**	**0.5151**
LinearDet (Ours)	2	$\mathbf{0.5197}_{\pm 0.0001}$	0.8694	**0.5216**
LinearDet (Ours)	3	$\mathbf{0.5257}_{\pm 0.0001}$	0.8712	**0.5210**

proposals improves mAP rapidly (0.328 at $N_{\text{box}}=50$ to 0.444 at $N_{\text{box}}=200$) and then saturates as N_{box} grows further. Conversely, with $N_{\text{box}}=200$, increasing refinement steps improves mAP from 0.444 to 0.463 by $N_{\text{step}}=8$, exhibiting a clear "fast-then-saturate" trend. This is the defining P2 property of generative detection, faithfully preserved by LinearDet: at a fixed checkpoint, more proposals and/or more refinement steps can be spent at inference to obtain a monotone accuracy gain, with the scaling curve exhibiting a clean fast-then-saturate trend along each axis.

Zero-shot on CrowdHuman. We further evaluate cross-dataset generalization by directly transferring models trained on COCO to CrowdHuman [52] without fine-tuning. Table 3 reports the zero-shot results under a fixed proposal budget ($N_{\text{box}}=500$) with different refinement steps, using the three standard CrowdHuman metrics: mAP, mMR (log-average miss rate, lower is better), and JI (Jaccard Index, higher is better). Across refinement steps, LinearDet leads DiffusionDet on mAP ($+0.3/+0.8/+0.8$ AP for $N_{\text{step}}=1/2/3$). JI (differences within 0.006) and mMR are close between the two methods (differences within 0.004).

Training-side ablations. We further ablate LinearDet's own training-time choices (noise distribution and box renewal) and find that both confirm the §2 findings on the LinearDet side: Gaussian/Uniform/Laplace training noise distributions all land within 0.5 AP of one another, and enabling box renewal contributes $+0.4$ AP during 3-step inference. Full tables are in Appendix D. A natural alternative is to train the cascade head on randomly perturbed boxes instead of linearly noised ones. Under a matched 150k schedule and without timestep conditioning in either model, training on randomly perturbed boxes lowers AP from 0.442 to 0.354 at one step and from 0.451 to 0.360 at four steps, indicating that the benefit comes primarily from how training proposals are constructed (Appendix E).

Table 4: **The linear replacement beyond DiffusionDet.** DiffuDINO: COCO val2017 AP, 80k iterations, 900 queries. DDP: Cityscapes val mIoU, ConvNeXt-T, 15k iterations, averaged over five inference seeds. Within each pair, only the corruption path and the update rule differ; one training run per method.

	DiffuDINO (AP)		DDP (mIoU)	
Method	N_{step}=1	N_{step}=3	N_{step}=1	N_{step}=3
Original (cosine + DDIM)	**0.4766**	0.4792	0.7687	0.7852
Linear (ours)	0.4763	**0.4800**	**0.7822**	**0.7855**

Table 5: **Under limited proposal budgets**: LinearDet and DiffusionDet match when proposal coverage is sufficient (N_{box}=500), while LinearDet leads on overall AP (and on AP_{50}/AP_{75}/AP_m/AP_l) in the sparse-coverage regime (N_{box}=20); AP_s is slightly lower. Results are averaged over three random inference seeds.

Method	N_{step}	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
			N_{box}=500				
DiffusionDet	1	$\mathbf{0.457}_{\pm 0.0005}$	**0.656**	0.489	**0.284**	0.482	0.613
DiffusionDet	2	$\mathbf{0.462}_{\pm 0.0000}$	**0.665**	**0.495**	**0.294**	0.483	**0.616**
DiffusionDet	3	$\mathbf{0.464}_{\pm 0.0006}$	**0.668**	0.494	**0.298**	0.486	0.618
LinearDet	1	$\mathbf{0.457}_{\pm 0.0012}$	**0.656**	**0.492**	0.282	**0.486**	**0.617**
LinearDet	2	$\mathbf{0.462}_{\pm 0.0006}$	0.664	**0.495**	0.288	**0.491**	**0.616**
LinearDet	3	$\mathbf{0.464}_{\pm 0.0008}$	0.667	**0.495**	0.291	**0.494**	**0.619**
			N_{box}=20				
DiffusionDet	1	$0.176_{\pm 0.0136}$	0.280	0.180	**0.087**	0.183	0.296
DiffusionDet	2	$0.243_{\pm 0.0070}$	0.382	0.251	**0.140**	0.261	0.377
DiffusionDet	3	$0.274_{\pm 0.0057}$	0.431	0.284	**0.152**	0.299	0.412
LinearDet	1	$\mathbf{0.191}_{\pm 0.0006}$	**0.298**	**0.199**	0.085	**0.200**	**0.328**
LinearDet	2	$\mathbf{0.252}_{\pm 0.0015}$	**0.388**	**0.263**	0.122	**0.268**	**0.403**
LinearDet	3	$\mathbf{0.283}_{\pm 0.0000}$	**0.435**	**0.297**	0.149	**0.311**	**0.438**

Beyond DiffusionDet. To test whether the linear replacement transfers beyond DiffusionDet, we apply the same replacement—linear noising in place of the cosine corruption and the linear update in place of DDIM—to two other diffusion models, keeping everything else matched (Table 4). In DiffuDINO [42], a DINO-based generative detector with 900 queries, we replace only the box path and keep its contrastive denoising and label noising; AP changes by -0.03 and $+0.08$ points at one and three steps, and both variants still gain from additional steps. In DDP [20], a diffusion head for semantic segmentation on Cityscapes [8], the linear version improves one-step mIoU by 1.4 points and matches the original at three steps. Each pair is a single training run under a matched budget (80k and 15k iterations), so we read these results as evidence of transfer rather than of superiority; protocols and full metrics are in Appendix E.

4.3 Diagnostic Study between LinearDet and DiffusionDet

This subsection reports three observations that accompany LinearDet's compact design: §4.3.1 shows that the linear prototype is stronger under a very limited proposal budget; §4.3.2 shows that deterministic linear updates yield smoother refinement trajectories, with a quantitative direction-consistency analysis provided in Appendix C; and §4.4 shows that these consistent trajectories can be aggregated into clean spatial heatmaps, a view that is obscured under stochastic diffusion sampling.

4.3.1 Comparison under sufficient and limited proposal budgets.

Table 5 compares LinearDet and DiffusionDet at two proposal-coverage extremes. When the random box set is sufficient (N_{box}=500), the two methods achieve nearly identical performance (both reach 0.464 AP at N_{step}=3). Under a very limited budget of 20 boxes per image (N_{box}=20, compared with 500 in the standard setting), LinearDet leads on overall AP at every step count: 0.176→0.191 at N_{step}=1, 0.243→0.252 at N_{step}=2, and 0.274→0.283 at N_{step}=3, with consistent gains on AP_{50}/AP_{75}/AP_m/AP_l; small objects are the exception, with AP_s lower by 0.2–1.8 points. This low-N_{box} setting directly tests behavior under sparse random initialization, where there is little coverage redundancy to fall back on. Since the two models match at N_{box}=500, these results suggest that LinearDet's coherent, stochasticity-free updates are more effective when the random-box set is thin.

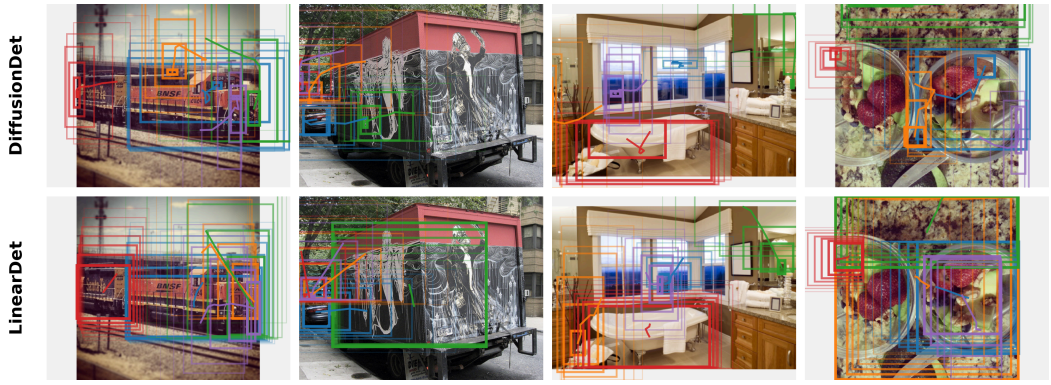

Figure 4: Qualitative comparison of box refinement trajectories between DiffusionDet (top) and LinearDet (bottom) on COCO val.



Figure 5: Visualization of LinearDet's learned box-update field and the derived heatmap. Top: box-update field visualization, where arrow colors from blue to red indicate classification confidence. Bottom: a heatmap obtained by weighting the box-update field.

4.3.2 Qualitative Visualization: Box Refinement Trajectories

We visualize the box refinement trajectories produced by LinearDet and DiffusionDet on several COCO images (examples are shown in Fig. 4). LinearDet updates boxes more smoothly, with less step-to-step jitter, while DiffusionDet shows more stochastic wander before convergence. This makes LinearDet's updates easier to inspect without sacrificing detection accuracy (Table 5).

4.4 Visualization of Learned Box-Update Fields

Figure 5 shows the learned update field and the confidence-weighted heatmap. Confident proposals exhibit coherent directions, and the aggregated field concentrates around object regions, suggesting that LinearDet steers random boxes toward object modes.

5 Conclusion

In this paper, we conducted a component-wise diagnosis of DiffusionDet. The evidence shows that the diffusion-specific choices commonly credited for generative detection—the timestep schedule, the initial noise distribution, and DDIM stochasticity—are not critical, whereas a learned denoiser, iterative refinement, and sufficient proposal coverage are the main drivers of performance. Based on this diagnosis, we proposed LinearDet, a minimal generative detection prototype that uses linear noising at training and linear denoising at inference. LinearDet matches DiffusionDet on COCO while preserving random-box initialization and test-time scaling, and the same replacement carries over to a DINO-based detector and to semantic segmentation. We hope LinearDet offers a clean baseline on which future generative detectors can add and evaluate new components, and that our diagnosis inspires further work across generative perception.

Acknowledgments and Disclosure of Funding

This work was supported by the National Science and Technology Major Project (Grant No. 2025ZD1606203), the Shanghai Municipal Universities Industry-Academia-Research Joint Innovation Program, and the National Natural Science Foundation of China (Grant No. 62272433). The authors declare no competing interests.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

[2] Enis Baty, C. P. Bridges, and Simon Hadfield. Flowdet: Unifying object detection and generative transport flows. *arXiv preprint arXiv:2512.16771*, 2025.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020*, 2020.

[6] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[7] Zhongxi Chen, Ke Sun, Xianming Lin, and Rongrong Ji. Camodiffusion: Camouflaged object detection via conditional diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, 2024. doi: 10.1609/aaai.v38i2.27890.

[8] 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 (CVPR)*, 2016.

[9] 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 (CVPR)*, 2009.

[10] 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 (ICLR)*, 2021.

[11] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision (IJCV)*, 2010.

[12] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: Exceeding YOLO series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[13] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[15] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *International Conference on Learning Representations (ICLR)*, 2022.

[16] Zhangxuan Gu, Haoxing Chen, Zhuoer Xu, Jun Lan, Changhua Meng, and Weiqiang Wang. Diffusioninst: Diffusion model for instance segmentation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024. doi: 10.1109/ICASSP48485.2024.10447191.

[17] Agrim Gupta, Piotr Dollár, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

[20] Yuanfeng Ji, Zhe Chen, Enze Xie, Lanqing Hong, Xihui Liu, Zhaoqiang Liu, Tong Lu, Zhenguo Li, and Ping Luo. DDP: Diffusion model for dense visual prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. URL `https://openaccess.thecvf.com/content/ICCV2023/html/Ji_DDP_Diffusion_Model_for_Dense_Visual_Prediction_ICCV_2023_paper.html`.

[21] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. MDETR: Modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[24] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M. Ni, and Lei Zhang. DN-DETR: Accelerate DETR training by introducing query denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[25] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[26] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[27] 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 *Computer Vision – ECCV 2014*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014. doi: 10.1007/978-3-319-10602-1_48.

[28] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[29] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[30] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2023.

[31] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[32] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. DAB-DETR: Dynamic anchor boxes are better queries for DETR. In *International Conference on Learning Representations (ICLR)*, 2022.

[33] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[34] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *Computer Vision – ECCV 2016*, 2016.

[35] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

[36] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[37] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. *arXiv preprint arXiv:2206.00927*, 2022. Accepted in NeurIPS 2022.

[38] Run Luo, Zikai Song, Lintao Ma, Jinlin Wei, Wei Yang, and Min Yang. Diffusiontrack: Diffusion model for multi-object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, 2024. doi: 10.1609/aaai.v38i5.28192.

[39] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yixuan Yuan, Shiliang Sun, and Jingdong Wang. Conditional DETR for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[40] Chen Min, Chengyang Li, Fanjie Kong, Qi Zhu, Dawei Zhao, and Liang Xiao. Gendet: Painting colored bounding boxes on images via diffusion model for object detection. *arXiv preprint arXiv:2601.07273*, 2026.

[41] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. Simple open-vocabulary object detection with vision transformers. In *Computer Vision – ECCV 2022*, 2022.

[42] Youssef Nawar, Mohamed Badran, and Marwan Torki. DiffuDETR: Rethinking detection transformers with denoising diffusion process. In *International Conference on Learning Representations (ICLR)*, 2026. URL `https://openreview.net/forum?id=nkp4LdWDOr`.

[43] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*, 2021.

[44] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[45] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

[46] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[47] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[48] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[49] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[50] Si-Dong Roh and Ki-Seok Chung. Diffusionvid: Denoising object boxes with spatio-temporal conditioning for video object detection. *IEEE Access*, 2023. doi: 10.1109/ACCESS.2023.3328341.

[51] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[52] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. CrowdHuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018.

[53] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.

[54] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[55] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[56] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[57] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse R-CNN: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[58] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[59] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[60] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. DINO: DETR with improved denoising anchor boxes for end-to-end object detection. In *International Conference on Learning Representations (ICLR)*, 2023.

[61] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[62] Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, and Jianfeng Gao. RegionCLIP: Region-based language-image pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[63] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[64] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detic: Detecting twenty-thousand classes using image-level supervision. In *Computer Vision – ECCV 2022*, 2022.

[65] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations (ICLR)*, 2021.

A LinearDet Pseudocode

For completeness, Algorithms 1 and 2 give the step-by-step training and inference procedures for LinearDet, corresponding to the description in §3.2 of the main paper.

B Full DiffusionDet Diagnostic Metrics

C Box-update Direction Consistency

This appendix provides a quantitative view of the smoother refinement trajectories discussed in the main paper. Figure 6 plots, for a $K{=}10$ inference trajectory, the cosine similarity between each step's box-update vector and the overall average box-update vector across the full trajectory. A higher value indicates a more consistent, smoother transport direction.

LinearDet maintains a consistently high cosine similarity throughout the trajectory (around 0.9 across all steps), meaning that each step's box update stays well aligned with the overall refinement direction. In contrast, DiffusionDet with stochastic sampling ($\eta{=}1$) drops to around 0.2 for most

Algorithm 1 LinearDet Training (Linear Noising + Linear Denoising Head)

Require: Image I; ground-truth boxes B_{gt}; proposal count N
Ensure: Updated parameters θ
Sample time $t \sim \mathcal{U}(0, 1)$
Encode $B_{gt} \to x_0$ in latent space; pad/truncate to N boxes
Sample Gaussian noise $x_1 \sim \mathcal{N}(0, I)$
Construct noisy boxes $x_t \leftarrow (1 - t)x_0 + tx_1$
Decode x_t to pixel-space boxes; extract RoI features from backbone/FPN
Linear Denoising Head predicts class logits and clean-box estimate $\hat{b}_0$ in pixel-space xyxy
Compute set prediction loss on $\hat{b}_0$
Backprop and update θ

Algorithm 2 LinearDet Inference (Iterative Box Refinement)

Require: Image I; proposal count N; time grid $\{t_k\}_{k=0}^{K}$ from $1 \to 0$
Ensure: Final boxes and scores
Sample initial noise $x_1^{\text{init}} \sim \mathcal{N}(0, I)$; set $x \leftarrow x_1^{\text{init}}$
for $k = 0$ to $K - 1$ **do**
  $t_{\text{curr}} \leftarrow t_k$; $t_{\text{next}} \leftarrow t_{k+1}$
  Decode x to pixel-space boxes; extract RoI features from backbone/FPN
  Linear Denoising Head predicts class logits and pixel-space clean boxes $\hat{b}_0$
  if $t_{\text{next}} > 0$ **then**
    Re-encode $\hat{b}_0$ to latent space as $\hat{x}_0$
    Update $x \leftarrow \frac{t_{\text{next}}}{t_{\text{curr}}}\, x + \left(1 - \frac{t_{\text{next}}}{t_{\text{curr}}}\right) \hat{x}_0$
    Optional: box renewal (drop low-confidence boxes; resample noise)
  end if
end for
Return the last step's predicted boxes $\hat{b}_0$ and class scores (optionally ensembling predictions across steps)

steps; switching to deterministic DDIM (η=0) improves early-step consistency but still degrades sharply at later steps (from 0.93 to 0.38). Under sufficient proposal budgets the two methods reach comparable AP (Table 5 in the main paper), consistent with the insensitivity established in §2.2; the smoother dynamics of LinearDet are a byproduct of the compact deterministic design rather than an accuracy advantage, and are what enable the clean trajectory visualizations and heatmap aggregation in the main paper.

D Additional LinearDet Ablations

This supplementary material contains additional training-side ablations for LinearDet that confirm, on the LinearDet side, findings established on DiffusionDet in §2. All results use the shortened ablation schedule (28,125 iterations at batch size 64; §4.1) unless otherwise noted.

D.1 Prediction target: $\hat{x}_0$ vs $\hat{u}$

As discussed in §3.2, predicting the clean-box estimate $\hat{x}_0$ is algebraically equivalent, up to the corresponding latent encoding, to predicting the linear-path direction $\hat{u} = x_0 - x_1$ when x_1 is fixed at sampling time. We ablate these two parameterizations on COCO under the shortened ablation schedule and otherwise identical settings.

Predicting $\hat{x}_0$ yields a small but consistent edge: AP improves from 0.396 to 0.398, AP_{50} from 0.585 to 0.587, and AP_s/AP_m/AP_l all move slightly in favor of $\hat{x}_0$; the two targets are comparable on AP_{75} (0.422 vs. 0.420). The gap is small, consistent with the algebraic equivalence of the two parameterizations. We adopt $\hat{x}_0$ prediction as the default because its target does not depend on the sampled noise x_1.

Table 6: **Full metrics for the DiffusionDet component diagnosis.** This table provides the per-metric values summarized in Table 1 of the main paper. COCO val2017, inference only; DiffusionDet baseline uses Normal schedule, Gaussian init, η=1, N_{step}=4, N_{box}=500, renewal on. Results are averaged from 3 different random inference seeds.

Axis	Variant	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
DiffusionDet baseline		0.465	0.669	0.498	0.299	0.488	0.617
Low-sensitivity variants ($\Delta AP \leq 0.3$)							
Schedule	Random	0.465	0.669	0.498	0.293	0.489	0.617
Schedule	Reverse	0.464	0.667	0.497	0.296	0.487	0.617
Schedule	Shuffled	0.464	0.669	0.496	0.298	0.488	0.618
Schedule	Constant (t=0.5)	0.465	0.669	0.498	0.294	0.489	0.617
Init	Uniform	0.462	0.666	0.495	0.293	0.487	0.615
Init	Grid	0.464	0.668	0.497	0.300	0.489	0.617
DDIM η	0.0	0.465	0.669	0.498	0.297	0.491	0.617
DDIM η	0.5	0.465	0.669	0.498	0.297	0.488	0.619
DDIM η	2.0	0.465	0.670	0.498	0.299	0.489	0.618
AP-sensitive variants							
N_{step}	1 (N_{box}=500)	0.457	0.656	0.489	0.284	0.482	0.613
N_{step}	2 (N_{box}=500)	0.462	0.665	0.495	0.294	0.483	0.616
N_{step}	8 (N_{box}=500)	0.466	0.669	0.499	0.296	0.490	0.619
N_{box}	50 (N_{step}=4)	0.411	0.598	0.440	0.251	0.440	0.573
N_{box}	100 (N_{step}=4)	0.449	0.642	0.481	0.279	0.475	0.612
N_{box}	300 (N_{step}=4)	0.464	0.669	0.497	0.296	0.488	0.619
Box renewal	Off	0.457	0.658	0.491	0.288	0.478	0.612
Head training: non-denoiser iteration stays within a narrow range							
Faster R-CNN	$N_{\text{step}} = 1$	0.418	0.625	0.458	0.247	0.462	0.537
Faster R-CNN	$N_{\text{step}} = 2$	0.422	0.626	0.464	0.248	0.464	0.548
Faster R-CNN	$N_{\text{step}} = 4$	0.419	0.620	0.461	0.244	0.462	0.547
Faster R-CNN	$N_{\text{step}} = 8$	0.413	0.608	0.453	0.232	0.457	0.543

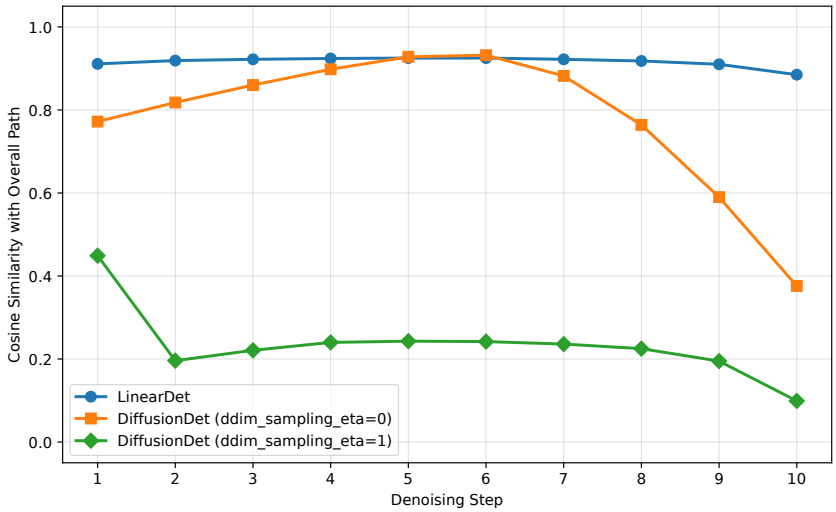

Figure 6: Box-update direction consistency across refinement steps (K=10). Cosine similarity between each step's box-update vector and the overall average across the full trajectory. Higher is smoother.

D.2 Training-time noise distribution

§2.2 showed that the trained DiffusionDet is largely insensitive to its *inference-time* initial proposal distribution. We extend the same probe to LinearDet by ablating the noise distribution used at *training time* to construct x_t, while keeping all other settings fixed.

All three distributions land within 0.5 AP of one another (Gaussian 0.398, Laplace 0.394, Uniform 0.393), with Gaussian slightly ahead on most scale-specific metrics including AP_l (0.545). This mirrors the inference-time observation in §2.2: the trained model absorbs very different noise sources at both training and inference, so the precise choice of noise distribution is not a critical lever for LinearDet. We adopt Gaussian noise by default for its small but consistent edge.

Table 7: **Prediction target ablation** (LinearDet, N_{box}=500, N_{step}=1). Predicting $\hat{x}_0$ matches or slightly exceeds predicting the path direction $\hat{u}$ on all metrics except AP_{75}.

Target	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Predicting $\hat{x}_0$	**0.398**	**0.587**	0.420	**0.227**	**0.421**	**0.545**
Predicting $\hat{u}$	0.396	0.585	**0.422**	0.225	0.420	0.542

Table 8: **Training-time noise distribution is not critical** (LinearDet, N_{box}=500, N_{step}=1): Gaussian, Uniform, and Laplace noise all land within 0.5 AP. Gaussian is adopted by default for its small consistent edge.

Noise	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Gaussian	**0.398**	**0.587**	**0.420**	0.227	**0.421**	**0.545**
Uniform	0.393	0.581	0.416	0.224	0.419	0.536
Laplace	0.394	0.583	0.417	**0.230**	**0.421**	0.537

D.3 Box renewal

Table 9: **Box renewal contributes** +0.4 **AP on LinearDet** (N_{box}=500, N_{step}=3), consistent with its role as a coverage-maintenance mechanism identified in §2.3.

Renewal	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
×	0.398	0.585	0.423	0.230	0.418	0.534
✓	**0.402**	**0.594**	**0.426**	**0.231**	**0.433**	**0.540**

Enabling box renewal improves AP from 0.398 to 0.402 (+0.4) with 3 refinement steps. By continuously replacing low-confidence proposals with fresh random boxes, box renewal maintains a higher level of proposal coverage across refinement steps, directly reinforcing the role of coverage identified in §2.3.

D.4 SNR scale

Table 10 shows a non-monotonic dependency on the SNR scale under the shortened ablation schedule: moderate values (1.0–2.0) achieve the best overall performance, while both overly large (3.0) and overly small (0.1) scales hurt accuracy. SNR scale 2.0 yields the best ablation AP (0.398) and strong AP_{50}/AP_{75} (0.587/0.420), whereas SNR scale 1.0 slightly improves AP_m (0.422) and matches the best AP_l (0.545). We adopt SNR scale 2.0 as the default design choice since it provides the best ablation AP and balanced performance across object sizes; extremely small scaling leads to a substantial drop (AP 0.337), highlighting the importance of proper noise conditioning.

D.5 Training vs inference proposal budget

Table 11 indicates that the train-time proposal budget affects how well LinearDet adapts to different inference budgets. For medium-to-large inference budgets (Eval 300/500/1000), training with N_{box}=500 consistently performs best (0.389/0.398/0.403), suggesting that richer proposal sets during training improve coverage and refinement robustness. Interestingly, when evaluating with only 100 proposals, training with 300 proposals yields the best result (0.353), while training with 500 slightly underperforms (0.345), implying a mild mismatch when the inference budget is much smaller than training. Training with a moderately large proposal budget (300–500) provides good robustness across inference settings, but for extremely low-budget inference, a closer match at training can be beneficial.

E Additional Controls and Transfer beyond DiffusionDet

Each comparison in this section uses one training run per method and is matched within its pair; inference-seed averages therefore do not measure training-seed variance. The random-perturbation controls below follow the original training configuration (batch size 16) for 150k iterations, whereas the ablations in Appendix D use the shorter fast-iteration schedule (28,125 iterations at batch size

Table 10: Ablation on SNR scale on COCO under the shortened ablation schedule (LinearDet, N_{box}=500, N_{step}=1).

SNR scale	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
3.0	0.384	0.567	0.408	0.220	0.407	0.522
2.0	**0.398**	**0.587**	**0.420**	**0.227**	0.421	**0.545**
1.0	0.393	0.580	0.419	**0.227**	**0.422**	**0.545**
0.1	0.337	0.491	0.358	0.149	0.355	0.509

Table 11: Ablation on training vs inference proposal budgets on COCO under the shortened ablation schedule (LinearDet). Rows indicate N_{box} at training and columns indicate N_{box} at inference.

Train \ Eval	100	300	500	1000
100	0.348	0.366	0.369	0.371
300	**0.353**	0.385	0.391	0.395
500	0.345	**0.389**	**0.398**	**0.403**

64); absolute AP is therefore not comparable across the two sets, and only differences within each pair should be compared.

DiffuDINO. DiffuDINO, introduced in DiffuDETR [42], generates boxes with a DINO-based decoder. Both variants share the DINO architecture, 900 queries, contrastive denoising, label noising, initialization, data, matcher, losses, optimizer, 80k training schedule, EMA usage, and evaluation protocol. Only the cosine box corruption and the DDIM box update are replaced by linear interpolation and the linear update of Eq. 2. Table 12 evaluates EMA checkpoints on the full COCO val2017 set.

Table 12: **DiffuDINO with a linear box path** (COCO val2017, 80k iterations, one training run per method).

Method	N_{step}	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR_{100}
Original	1	0.4766	0.6472	0.5193	0.3017	0.5121	0.6351	0.6887
Linear	1	0.4763	0.6468	0.5174	0.3029	0.5155	0.6323	0.6900
Original	3	0.4792	0.6514	0.5212	0.3042	0.5124	0.6370	0.6927
Linear	3	0.4800	0.6520	0.5220	0.3045	0.5143	0.6343	0.6955

DDP on Cityscapes. We use the official DDP [20] semantic-segmentation implementation on Cityscapes [8]. Original-DDP and Linear-DDP share the ConvNeXt-T backbone, architecture, initialization, data, losses, optimizer, global batch 16, training seed, and a 15k-iteration schedule; the linear variant replaces the cosine log-SNR corruption and DDIM sampling in the head. Table 13 reports the Cityscapes validation split, averaged over five paired inference seeds (standard deviations below 0.0003). The 15k schedule is much shorter than the official 160k recipe, so this is short-schedule evidence of transfer rather than a full reproduction.

Linear noising versus random perturbation. Both models use the same R50-FPN and six-stage head, matching and detection losses, clean-box targets, optimizer, 150k training steps, global batch 16, and damped inference update; timestep conditioning is disabled in both, so the construction of training proposals is the only core difference. Linear noising outperforms random perturbation by 8.9 AP at one step (0.442 vs. 0.354) and 9.1 AP at four steps (0.451 vs. 0.360), with 500 proposals. Replacing the random model's damped update with direct stage-to-stage feedback changes AP by at most 0.2 points, so the gap is not explained by the inference update.

Cross-paper context: FlowDet. Table 14 evaluates our models at the operating points reported by FlowDet [2] (100 inference proposals, 1–3 steps). This is cross-paper context rather than a controlled comparison: the training recipes differ (about 532k vs. 450k iterations, 300 vs. 500 training proposals, and post-processing), and even the nominally identical DiffusionDet baseline differs between the two reports. No official FlowDet code or checkpoint was available to remove these confounders.

Table 13: **DDP with a linear path** (Cityscapes val mIoU, 15k iterations, one training run per method).

Method	N_{step}=1	N_{step}=3
Original-DDP	0.7687	0.7852
Linear-DDP	**0.7822**	**0.7855**

Table 14: **Cross-paper context at FlowDet's operating points** (COCO val2017 AP, R-50, 100 inference proposals). FlowDet columns are taken from its paper; LinearDet values are means over inference seeds.

N_{step}	FlowDet	DiffusionDet (in FlowDet)	DiffusionDet (ours)	LinearDet (ours)
1	0.432	0.431	0.414	0.418
2	0.447	0.430	0.439	0.441
3	0.455	0.451	0.448	0.450

F Mechanism of the Diagnosis

Linear update as an Euler step. For a fixed predictor, the endpoint-derived velocity $v_\theta(x,t) = (x - \hat{x}_0(x,t))/t$ gives

$$x + (t' - t)\, v_\theta(x,t) = \frac{t'}{t}\, x + \Big(1 - \frac{t'}{t}\Big)\hat{x}_0(x,t),$$

which is Eq. 2. This identity concerns the update rule; it does not imply that separately trained endpoint and velocity networks reach the same accuracy.

Attenuation of the initial state. Fix a matched target $x^\star$ and consider a segment without renewal. With $\delta_k = x_k - x^\star$, $e_k = \hat{x}_0(x_k, t_k) - x^\star$, and $\alpha_k = t_{k+1}/t_k$, the linear update gives $\delta_{k+1} = \alpha_k \delta_k + (1 - \alpha_k) e_k$, so the direct weight on the initial error δ_0 after K steps is $\prod_k \alpha_k = t_K/t_0$, which vanishes at $t_K = 0$. The prior can still act through the visited states and the prediction errors e_k, so this is not a proof of prior independence. Empirically, on the DiffusionDet model of §2, the initial recall of ground-truth boxes spans 5.32 points across proposal priors but only 0.08 points after one refinement step.

Stochasticity, renewal, and ensembling. On the same model, a full $\eta\times$renewal$\times$ensembling factorial shows that the absolute AP effect of stochasticity is at most 0.833; its largest effect occurs only when renewal and ensembling are both disabled, where stochastic sampling is worse. Renewal and ensembling are largely interchangeable: renewal contributes 0.667–4.133 AP and ensembling 0.700–4.100 AP, with the lower values when the other mechanism is enabled (the default) and the upper values when it is disabled. With renewal enabled, approximately 487 of 500 proposals are replaced at each early step, so renewal already supplies the exploration that per-step sampling noise would otherwise provide.

Linear-side controls. On the LinearDet side, removing timestep conditioning changes AP by 0.26–0.47 points (150k controls of Appendix E); on the full models at 500 proposals and four steps, the span across integration time grids is at most 0.30 AP, and state-reanchored versus fixed-source updates differ by 0.10 AP.

G Limitations and Broader Impact

Limitations. LinearDet is intended as a compact prototype for the generative-detection family defined by random-box initialization and test-time scaling, not as a state-of-the-art detector. Its matched comparison is primarily against DiffusionDet under the same architecture and per-step computation; modern discriminative detectors improve AP through orthogonal mechanisms such as learned queries, query denoising, and deformable attention. Our diagnostic conclusions are based on controlled interventions around DiffusionDet-style models and should not be read as a claim that diffusion is unnecessary for all detection or generation settings. In particular, tasks whose metrics reward multiple plausible outputs or calibrated uncertainty may still benefit from distributional sampling; the single-set detection metrics studied here do not address that case. The transfer and control experiments in Appendix E use one training run per method. Under very sparse proposals,

LinearDet's small-object AP is lower than DiffusionDet's, while late-stage refinement stability is comparable (late-oscillation rate 47.1% vs. 48.2%).

Broader impact. The positive impact of this work is to simplify a generative-detection baseline and make its test-time scaling behavior easier to study, reproduce, and visualize. As with object detection systems more broadly, potential negative uses include surveillance and other monitoring applications. This paper does not introduce new datasets, private data collection, or deployment mechanisms beyond evaluation on standard public benchmarks.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the diagnostic findings on DiffusionDet, the LinearDet formulation, and empirical claims, which are supported by the revisiting analysis, method, and experiment sections.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The paper states its scope explicitly: LinearDet is positioned as a compact prototype within the generative-detection family defined by data-independent random initialization and test-time scaling, rather than as a computationally cheaper or state-of-the-art detector. The empirical conclusions are limited to the studied DiffusionDet-style architecture, COCO/CrowdHuman evaluations, and the controlled interventions reported in the paper.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A].

   Justification: The paper proposes and empirically studies a detection framework, but does not present formal theoretical results or proofs.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The paper describes datasets, train/test splits, model alignment with DiffusionDet, optimizer settings, schedules, proposal budgets, and refinement-step settings in Sections 4.1 and 4.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No].

   Justification: The experiments use public benchmarks and are described with implementation details, but the current anonymous manuscript package does not include a public code archive with full reproduction scripts.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: Section 4.1 gives the dataset, model, optimizer, schedule, batch size, and evaluation protocol used for the reported results.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

Justification: Our main detector results in Table 2 are averaged over 3 training seeds with standard deviations on AP. Tables 3, 5, and 1 are averaged over 3 inference seeds (full per-metric values in Appendix Table 6). The DiffuDINO, DDP, and random-perturbation studies in Appendix E use one training run per method, so their results do not measure training-seed variance.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: All models are trained on 8 NVIDIA RTX 4090 GPUs (24 GB each); a full 450k-iteration run takes roughly 24–48 hours depending on the backbone, and ablation and transfer runs use shorter schedules. The manuscript reports the training schedule, batch size, inference proposal budgets, refinement-step budgets, and whether experiments use the full or shortened ablation schedule.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work studies object detection on standard public benchmarks and does not involve human-subject experiments, private data collection, or deceptive deployment.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The work may have positive impact by simplifying a generative-detection baseline and making test-time scaling behavior easier to study and reproduce. As with object detection systems more broadly, possible negative impacts include deployment in surveillance or other monitoring applications; the paper does not introduce new data collection or deployment mechanisms beyond standard detection benchmarks.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release a high-risk generative model, scraped dataset, or similar asset requiring special safeguards.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: The paper cites the original sources for the datasets, baseline methods, and model components used in the experiments, including COCO, CrowdHuman, DiffusionDet, and the relevant detector/backbone families. The work uses these public research assets for benchmark evaluation under their respective terms.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

Justification: The manuscript does not introduce or release a new dataset or benchmark asset.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The research does not involve crowdsourcing or human-subject studies.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The research does not involve human-subject studies, so IRB approval is not applicable.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: The core method development and experiments do not rely on LLMs as an original or non-standard methodological component.