UNDER REVIEW

RESCIENCE C

Replication / Computer Vision, Medical Imaging

[¬Re] Diffusion-Based Hierarchical Multi-Label Object Detection to Analyze Panoramic Dental X-rays

Christopher Huang[1], Spursh Deshpande[2], Wenhao Lu[3]

[1]Heritage High School, Brentwood, California, USA – [2]Crescenta Valley High School, La Crescenta, California, USA – [3]Millburn High School, Millburn, New Jersey, USA

Edited by
(Editor)

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
19 August 2026

Published
–

DOI
–

A replication of I. E. Hamamci, S. Er, E. Simsar, A. Sekuboyina, M. Gundogar, B. Stadlinger, A. Mehl and B. Menze. Diffusion-Based Hierarchical Multi-Label Object Detection to Analyze Panoramic Dental X-rays. Medical Image Computing and Computer Assisted Intervention (MICCAI 2023), pp. 389-399, 2023..

Abstract HierarchicalDet trains one diffusion-based detector across three tiers of DENTEX panoramic X-ray annotation and attributes its accuracy gain to a noisy-box manipulation scheme that feeds predictions from an earlier tier into the denoising process of the next. No trained weights are released, so any reproduction has to retrain the whole three-stage pipeline. We did so on free-tier Kaggle hardware under a 900-iteration budget per stage. The resulting models reach 6 to 13 percent of the reported AP, so this is a failed reproduction, and the shortfall is attributable to our budget rather than to the original work. Within that budget the direction of the manipulation ablation held at all three tiers, although per-class results show the tier-level gap is carried almost entirely by one easy class. The weight-transfer ablation inverted: the variant that is strongest in the paper collapsed to zero AP for us, which is what a short schedule predicts. Two findings do not depend on our budget. The public DENTEX test release carries no Deep Caries label at all, so it supports three of the four diagnosis classes and the reported diagnosis numbers cannot be recomputed from public data by anyone. The release also states two different licences for the same dataset. We report per-class AP, sampling-step latency, corruption sensitivity and failure-mode counts, none of which the original paper covers, and we say plainly which of them are informative at this training scale and which are not.

1 Introduction

HierarchicalDet[1] attacks a real annotation problem in dental radiology. The DENTEX panoramic X-ray collection[2] is labelled at three different depths: one part carries only quadrant boxes, a second adds tooth enumeration, and only the smallest part carries the full quadrant-enumeration-diagnosis triple. Rather than throw away the shallow annotations, the authors adapt DiffusionDet[3] into a multi-label detector trained across all three tiers in sequence, and they introduce a noisy-box manipulation step: at each stage, boxes predicted by the model trained on the previous tier are mixed into the noisy boxes that seed the denoising process. Their Table 1 reports 37.6 AP at the diagnosis tier and credits the manipulation step with the margin over the ablated variants.

We wanted to know two things. Whether the manipulation ablation holds up, and whether the pipeline can be retrained end-to-end by someone without a GPU cluster. The second question is not incidental. The repository publishes no trained weights, so there is no path to the reported numbers that does not go through retraining all three stages. That makes the compute floor of this paper an accessibility question, and answering it was the point of running the study on free-tier Kaggle hardware.

Correspondence should be addressed to Wenhao Lu (wenhao@nxthorizon.org)
The authors have declared that no competing interests exists.
Code is available at https://github.com/Alscend-Research/dental-repro.
Data is available at https://huggingface.co/datasets/ibrahimhamamci/DENTEX.

The short version of what happened: we retrained, we got 6 to 13 percent of the reported AP, and the models never became good enough for most of the questions we wanted to ask. That is a failed reproduction, and we label it as one. It is not evidence against HierarchicalDet. We spent a small fraction of a sensible training schedule and got a model that behaves exactly like a detector trained for a small fraction of a sensible training schedule.

What survives the budget is worth reporting anyway. The direction of the manipulation ablation held at every tier, though a per-class breakdown shows the gap is carried almost entirely by one geometrically easy class. The weight-transfer ablation inverted, for a reason that says more about short schedules than about the claim. And two problems we found have nothing to do with our compute at all: the public DENTEX test split does not contain the label set the reported diagnosis numbers are computed over, and the dataset ships with two contradictory licence statements. Neither is fixable by spending more GPU hours. Both affect anyone who tries this.

Every number, figure and table below was produced by a notebook that actually ran. Assets that were planned but not produced are recorded as such in the release manifest and discussed in Section 4.2, rather than quietly dropped.

2 Scope of reproducibility

We take the following claims from the original paper. The status column reflects what our budget actually permitted, decided before we looked at the numbers.

- **C1.** The hierarchical multi-label model outperforms base DiffusionDet at every tier. *Not tested.* Training a base DiffusionDet control was cut from the plan to stay inside the compute budget.
- **C2.** Noisy-box manipulation is the component responsible for the accuracy gain. *Tested, direction reproduced, magnitude not.*
- **C3.** Weight transfer on its own does not improve accuracy. *Tested and contradicted, but confounded by the shortened schedule.*
- **C4.** The full model outperforms RetinaNet, Faster R-CNN and DETR at every tier[4,5,6]. *Not tested.* Three additional detector trainings did not fit the budget.
- **C5.** SimMIM pretraining[7] on 1,571 unlabelled X-rays contributes to the result. *Out of scope.* The authors' SimMIM checkpoint is not published, so this cannot be isolated by anyone working from the release.
- **C6.** The reported numbers are obtained on the DENTEX test split. *Tested, and this is where we found a problem that does not depend on our budget.* See Section 3.2.

C1, C4 and C5 are cited but untested here, and we do not want a reader to come away thinking otherwise. A reproduction that trains three ablation variants and skips four baselines is a partial one, and the partiality is a budget decision we made openly rather than a result.

3 Methodology

3.1 Model

The model is DiffusionDet with a ResNet-50 + FPN backbone[8,9] and a detection head that emits three label vectors per box, one per tier. Detection is posed as a denoising process over box coordinates[10,11]: the model is handed 1,000 random noisy boxes and

iteratively refines them. We measured 173.44 M parameters and a 662 MB checkpoint, which matches the released configuration.
We used the authors' code without modifying the model. The one behavioural change we made is documented in Section 4.2: we re-enabled the inference-time prior-box injection path for a single experiment, and it is off everywhere else, so no main-table number depends on it.

3.2 Data

We wrote a DENTEX-to-COCO converter, because the release does not include one and the configs point at absolute paths that exist only on the authors' machines. Table 1 gives the resulting splits. Every JSON we produced is SHA-256 hashed in the release so a reviewer can confirm they are looking at the same data.

Split	Images	Annotations	Label depth
Quadrant (train)	693	2,772	quadrant only
Quadrant-enumeration (train)	634	—	quadrant + tooth
Diagnosis (train)	705	—	full triple
Diagnosis (val)	50	—	full triple
Diagnosis (test)	250	1,043	full triple
Unlabelled	1,571	—	none

Table 1. Converted DENTEX splits. The quadrant training split contains exactly four annotations per image, one per quadrant region, which matters for Section 4.1.4.

Two things surfaced during conversion that a reader should know about.

The test split cannot support the reported diagnosis evaluation — The released test annotations use a nine-code Turkish clinical labelling scheme and contain no Deep Caries label anywhere. Every carious tooth is plain *çürük* (code 1). The public release therefore cannot distinguish Caries from Deep Caries, and test-split evaluation covers three of the paper's four diagnosis classes. Deep Caries has no ground truth at all, so its per-class AP is undefined rather than zero, and the diagnosis-tier mean averages over the classes that have ground truth.
This is a property of the data release and not of our conversion. The file the released configuration expects, `test_merged_disease_coco3class.json`, was never published. Its name is consistent with a three-class merge, which would match what the public data can support, but we cannot confirm what it contains without the file. The consequence stands either way: the reported diagnosis numbers cannot be recomputed from public data by anyone, including us, and any future comparison against them inherits that gap.

The dataset ships two different licences — The HierarchicalDet GitHub repository states CC BY-SA for DENTEX; the HuggingFace dataset card states CC BY-NC-SA. We assumed the stricter non-commercial reading throughout. We report this rather than resolve it, since only the authors can.

3.3 Hyperparameters

We changed one thing: the iteration budget. Everything else is the repository default, including the training seed of 40244023 that ships with the code. Training ran in mixed precision[12] on two Tesla T4 GPUs with two dataloader workers. The diagnosis stage ran for 900 iterations. Sampling used one denoising step at inference except in the step sweep of Section 4.2.1. cuDNN was set to deterministic with benchmarking off.

Inference starts from random noisy boxes, so a single evaluation is a draw from a distribution rather than a fixed number. Every result below is a mean over inference seeds 0, 1 and 2, with the standard deviation reported. Residual nondeterminism we did not remove: AMP reduction order during training, atomics in the torchvision ROIAlign and NMS CUDA kernels, and dataloader worker interleaving.

One seed is one seed. We varied inference seeds and not training seeds, so the spreads below understate the true variance of the comparison[13]. Any statement we make about ablation ordering should be read with that in mind.

3.4 Experimental setup and code

Code, configs and the full asset manifest are at github.com/AIscend-Research/dental-repro. The pipeline is ten Kaggle notebooks over a shared `src/` package. The repository vendors modified copies of detectron2[14] and pycocotools; installing either from PyPI over the checkout breaks the build, which is worth knowing before anyone tries.

We trained five checkpoints: two intermediate stage models (*Stage quadrant*, *Stage enumeration*) and three diagnosis-stage variants (*Ours full*, *Ours w/o Manipulation*, *Ours w/o Transfer*). All five were evaluated on the 250-image diagnosis test split at all three tiers using COCO AP[15]. Environment: Python 3.12.13, PyTorch 2.10.0+cu128[16], CUDA 12.8, two Tesla T4s.

Re-running the evaluation does not require a GPU or any retraining. Attaching the released checkpoint and data artifacts and running notebook 03 rebuilds every table and figure in this paper.

3.5 Computational requirements

We report two numbers and keep them apart, because only one of them is a measurement.

Evaluation compute is **1.96 GPU-hours**, summed from per-run wall-clock records that are archived alongside the results. Training compute is **not recoverable** from our archive. The `run_record.json` files written during training were produced in earlier Kaggle sessions and were never carried into the released repository. Our own recollection of the total study cost is roughly 22 GPU-hours, and we label that as recollection, not measurement, because nothing we archived can confirm it.

That is a reproducibility failure in our own artifact, and it is the most embarrassing thing in this paper. We flag it in Section 5.2 rather than bury it, and the pipeline now retains run records at write time so the gap does not recur.

At inference the model needs 1.20 GB of GPU memory and runs at 124 images per minute on a single T4 at one sampling step, or 0.48 s per image. That part is genuinely cheap. The training is what costs.

4 Results

4.1 Results reproducing the original paper

Table 2 gives every number we produced. Table 3 places the three trained variants next to the reported values.

The scale of the shortfall — Between 6 and 13 percent of the reported AP. The gap narrows at AP_{50}, where we reach 30 percent at the diagnosis tier, which is the expected signature of a detector that has learned roughly where objects are but not yet how to place a box tightly around one. The AP_{75} column makes the same point more bluntly: 1.02 against a reported 44.0. Nothing here is close.

Method	AR	AP	AP_{50}	AP_{75}	AP_l
Quadrant tier					
Ours full	0.257 ± 0.004	2.65 ± 0.07	9.74 ± 0.49	0.55 ± 0.09	2.72 ± 0.07
Ours w/o Manipulation	0.193 ± 0.004	1.30 ± 0.02	5.15 ± 0.19	0.21 ± 0.03	1.33 ± 0.02
Ours w/o Transfer	0.000 ± 0.000	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Stage enumeration	0.044 ± 0.002	0.04 ± 0.00	0.22 ± 0.01	0.00 ± 0.00	0.04 ± 0.00
Stage quadrant	0.000 ± 0.000	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Enumeration tier					
Ours full	0.162 ± 0.003	1.83 ± 0.05	6.57 ± 0.29	0.39 ± 0.10	1.92 ± 0.04
Ours w/o Manipulation	0.151 ± 0.004	1.37 ± 0.02	5.31 ± 0.19	0.23 ± 0.05	1.40 ± 0.02
Ours w/o Transfer	0.000 ± 0.000	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Stage enumeration	0.042 ± 0.001	0.03 ± 0.01	0.15 ± 0.01	0.01 ± 0.01	0.03 ± 0.01
Stage quadrant	0.000 ± 0.000	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Diagnosis tier					
Ours full	0.266 ± 0.002	4.97 ± 0.16	18.09 ± 0.89	1.02 ± 0.26	5.23 ± 0.16
Ours w/o Manipulation	0.226 ± 0.011	2.78 ± 0.27	11.44 ± 0.46	0.46 ± 0.20	2.82 ± 0.29
Ours w/o Transfer	0.000 ± 0.000	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Stage enumeration	0.010 ± 0.001	0.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Stage quadrant	0.000 ± 0.000	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00

Table 2. Reproduced results on the 250-image DENTEX test split, mean ± std over three inference seeds. AP_s and AP_m are omitted throughout: the COCO evaluator returns both as undefined on this split, because every ground-truth box falls in the large area range.

		AP			AP_{50}		
Tier	Method	orig.	ours	%	orig.	ours	%
Quadrant	Ours full	43.2	2.65	6.1	65.1	9.74	15.0
	Ours w/o Manipulation	40.0	1.30	3.3	60.7	5.15	8.5
	Ours w/o Transfer	42.7	0.00	0.0	64.7	0.00	0.0
Enumeration	Ours full	30.5	1.83	6.0	47.6	6.57	13.8
	Ours w/o Manipulation	30.4	1.37	4.5	46.5	5.31	11.4
	Ours w/o Transfer	32.8	0.00	0.0	49.4	0.00	0.0
Diagnosis	Ours full	37.6	4.97	13.2	60.2	18.09	30.0
	Ours w/o Manipulation	36.3	2.78	7.7	55.5	11.44	20.6
	Ours w/o Transfer	39.4	0.00	0.0	61.3	0.00	0.0

Table 3. Original versus reproduced. The % column is the fraction of the reported value we recovered. Original values are quoted from Table 1 of the paper and were not produced here.

One statistic captures the state of these models better than the tables do. We had planned a per-tier label-error breakdown, which filters detections at a 0.50 confidence threshold before matching them to ground truth. *No detection from any checkpoint clears that threshold.* Against 1,043 ground-truth boxes, the qualifying prediction count is zero, so the label-error rate is undefined rather than zero. We dropped the analysis instead of publishing a chart of zeros, which would have implied a measurement we did not make.

C2: manipulation ordering held, magnitude did not — Ours full beats Ours w/o Manipulation at every tier, so the sign of the ablation reproduces. The magnitudes do not correspond in any useful way. At the diagnosis tier the original margin is 1.3 AP, a relative gain of 3.6 percent. Ours is 2.19 AP, a relative gain of 79 percent. Two arms that are both far from convergence mostly differ in how fast they get off the floor, which is not the same measurement as which one converges higher.

The per-class breakdown in Figure 1 sharpens this. The diagnosis-tier margin is carried almost entirely by ImpactedDisease, where the full model reaches 11.12 AP against 3.36 for the ablated one. On CariesDisease the ablated model is *ahead*, 4.79 to 3.62. Periapical Lesion sits near 0.2 for both. Impacted third molars are the largest, highest-contrast and most spatially stereotyped targets in the set, so what our data supports is the nar-

row statement that manipulation helps first on the easiest class. Whether the benefit generalises to the harder classes is a question about converged models, and we did not train any.

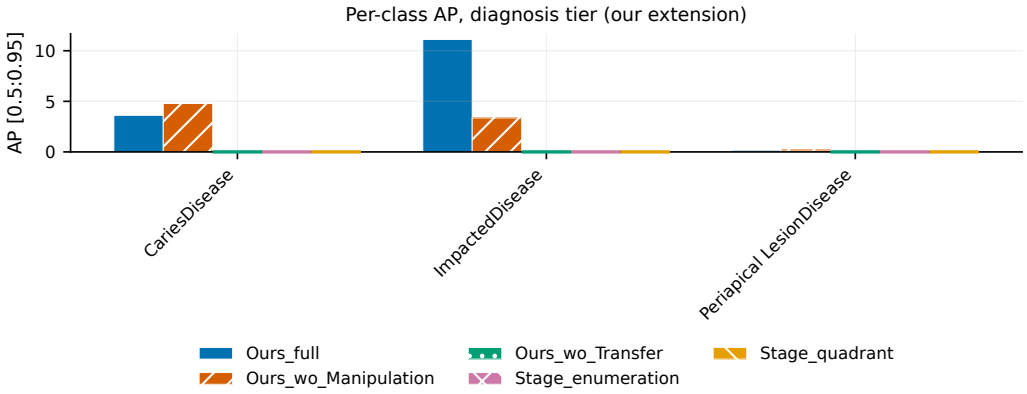

Figure 1. Per-class AP at the diagnosis tier, an extension: the original paper reports tier-level aggregates only. Deep Caries is absent because the released test split carries no ground truth for it (Section 3.2).

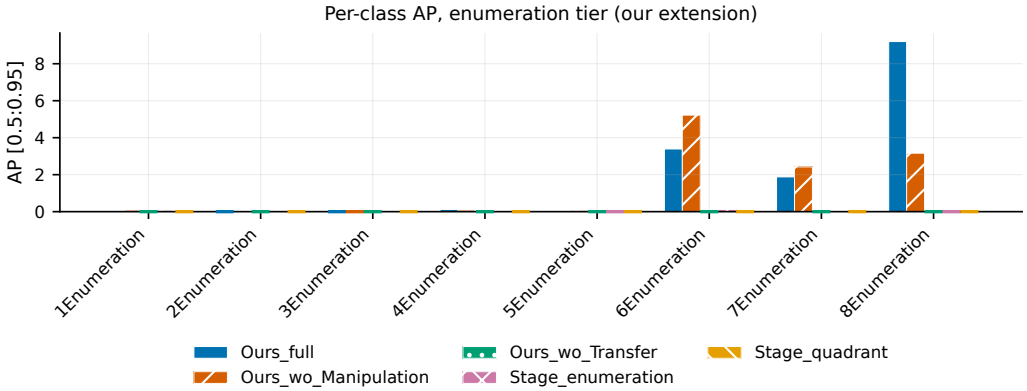

Figure 2. Per-class AP at the enumeration tier. Accuracy concentrates on tooth positions 6, 7 and 8; positions 1 through 5 are essentially at zero for every checkpoint.

C3: the transfer ablation inverted — In the original, Ours w/o Transfer is the *strongest* diagnosis model at 39.4 AP, which is the evidence for the claim that transfer alone contributes nothing. Ours scores 0.00 at every tier and every metric, with zero variance across seeds. We checked whether this was a broken checkpoint, and it is not. The model loads, runs, and emits 225,380 boxes across the 250 test images, with degenerate boxes present in every single image. It behaves like a network that never learned to localise, which is what a randomly initialised detector trained for 900 iterations should look like. Under a truncated schedule, initialisation dominates everything else, so this variant was always going to lose. Our result contradicts C3 for a reason that has nothing to do with whether C3 is true.

We had planned to defend the ablation ordering with a checkpoint-trajectory analysis, evaluating intermediate checkpoints to show the ordering is stable across training progress rather than an artifact of where we stopped. That experiment did not survive contact with the pipeline. Intermediate checkpoints were retained for only one of the three variants, so the trajectory plot has a single point for two of three curves. We are

not showing a figure that is one point wide, and the ordering-stability check the paper most needs remains unperformed.

A caveat on the stage checkpoints — Stage quadrant and Stage enumeration score at or near zero on all three tiers. Two tiers of that are expected, since neither checkpoint has seen diagnosis labels. The quadrant-tier number needs a caveat we did not anticipate.
Quadrant training data contains exactly four annotations per image, one large box per quadrant region. The flattened tier-0 test ground truth attaches quadrant labels to *tooth-level* boxes. These are different objects at different scales, so a quadrant-stage model can predict quadrant regions perfectly and still score 0.00 against tooth-level ground truth. Tier-0 AP for the stage checkpoints is therefore not a measure of how good the stage model is. The main-table variants are unaffected, since all three train through to the diagnosis stage and are evaluated against the same ground truth.

4.2 Results beyond the original paper

These experiments are not in the original. We report them because the artifacts exist and because two of them are informative about the pipeline even at this accuracy. We are explicit about which conclusions our accuracy level can and cannot support.

Sampling steps against latency — Table 4 and Figure 3. Going from one denoising step to four buys 0.16 diagnosis AP and costs 77 percent more wall-clock time. The gain sits inside the seed spread of the four-step measurement (±0.28), and the quadrant tier actually gets worse. At this training scale there is no measurable accuracy return on extra denoising.
One deviation constrains this result. The released cross-timestep ensembling path is unrunnable in the multi-label fork: `inference()` returns single-label variable names that do not exist there, and `ddim_sample` then treats the aggregate as both a tensor and a per-tier list. We ran the sweep without ensembling, so this measures denoising depth alone and not the full multi-timestep procedure the upstream single-label model would apply.

Steps	Quadrant AP	Enum. AP	Diagnosis AP	s / image
1	2.63 ± 0.07	1.80 ± 0.00	4.88 ± 0.04	0.484
4	2.22 ± 0.17	1.90 ± 0.13	5.04 ± 0.28	0.856

Table 4. Diffusion sampling steps against measured per-image latency, single T4, batch size 1, 250 images.

Corruption sensitivity — Table 5 and Figure 4. We degraded the test images with Gaussian blur, JPEG recompression and downscaling, in the spirit of standard corruption benchmarks[17], and got a result we did not expect. Mild blur, JPEG and downscaling all sit *at or above* the clean baseline. Only $\sigma = 4$ blur is clearly below it. AP goes *up* as JPEG quality drops from 50 to 20, and up again as the scale factor drops from 0.5 to 0.25.
We are not going to call this robustness. A model at 4.9 AP is close enough to the floor that the ranking of corruption conditions carries little information about the converged model. One hypothesis worth someone's time: a barely-trained backbone may be fitting high-frequency texture that low-pass corruption happens to remove, in which case mild blur is acting as an unintended test-time regulariser. We did not isolate that, and we would want the whole grid rerun on a converged checkpoint before anyone treats these curves as a robustness claim.

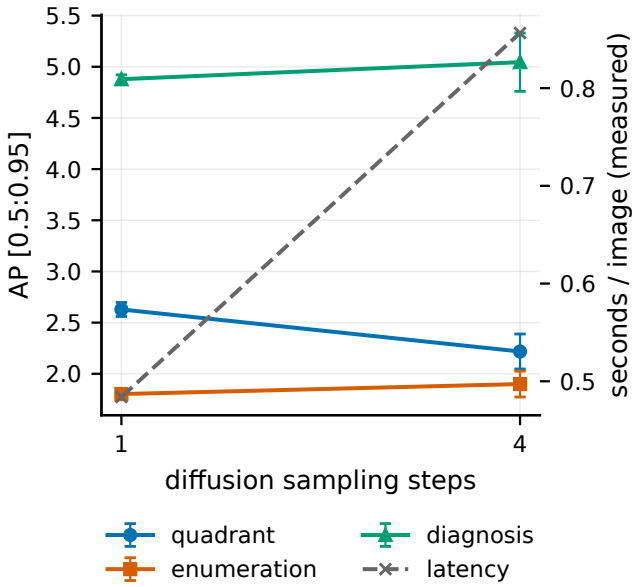

Figure 3. AP against denoising steps with measured latency on the right axis. Error bars are the standard deviation over inference seeds.

Condition	Diagnosis AP	Δ vs clean
Clean	4.88 ± 0.04	—
Blur $\sigma = 2$	5.23 ± 0.15	$+0.35$
Blur $\sigma = 4$	4.25 ± 0.36	-0.63
JPEG $q = 50$	4.86 ± 0.09	-0.02
JPEG $q = 20$	5.15 ± 0.31	$+0.27$
Downscale 0.5	5.10 ± 0.17	$+0.22$
Downscale 0.25	5.40 ± 0.08	$+0.52$

Table 5. Diagnosis-tier AP under input corruption, Ours full. Four of six corruptions score above the clean baseline, which we read as a statement about the training budget rather than about robustness.

Clean and stress subsets, and a flaw in our own protocol — We split the test set into a clean subset (143 images with typical annotation counts and at most two distinct diagnosis classes) and a stress subset (51 images that are unannotated, in the top decile by annotation count, or carry at least three distinct diagnosis classes). We expected stress to be harder. It scored higher at every tier: 6.60 against 5.05 diagnosis AP for Ours full, and 3.71 against 2.40 for Ours w/o Manipulation.

The subset rule is the problem, and it is ours. Selecting by annotation count moves the number of ground-truth boxes, which moves the AP denominator. What the split measures is object density, not difficulty. We report the numbers and the flaw together rather than presenting a comparison we no longer believe in. A usable version of this experiment would stratify on something that does not enter the metric, such as image acquisition quality or annotator agreement.

Hierarchy fault injection: a void result — This experiment was meant to test how much the diagnosis tier depends on the prior tier's boxes, by jittering and dropping the injected prior-tier boxes at inference. All six conditions returned bit-identical per-seed AP. Nothing was injected.

The mechanism: the released code injects prior-tier boxes during training only, and the inference-time path in `detector.py` is entirely commented out and references unpub-

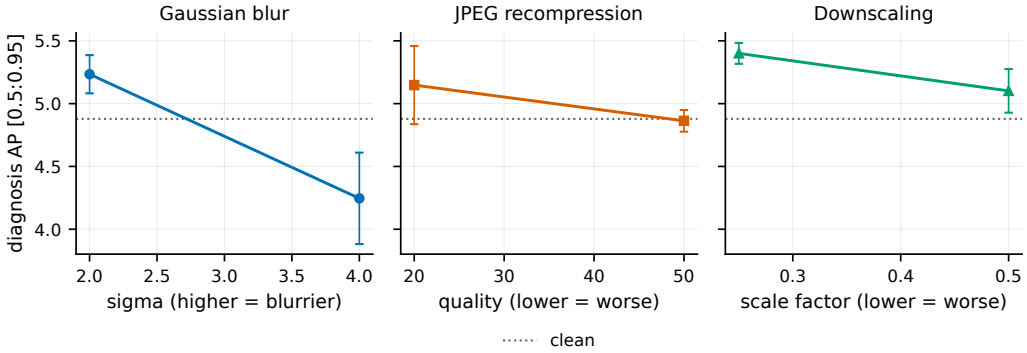

Figure 4. Diagnosis-tier AP under each corruption family. Separate panels because the severity axes are different quantities in different units. The dotted line is the clean baseline.

lished paths. We re-enabled it, but the injector contributes nothing when no prior box clears its score threshold, and at this accuracy level none does. It is the same condition that made the label-error analysis uncomputable. The experiment did not test what it was written to test, so we record it as void rather than present six identical rows as a robustness finding.

Failure-mode counts — Table 6. No run crashed and no image came back with zero detections, so the pipeline is at least stable. The interesting column is degenerate boxes. Ours full produces them in 111 of 250 images; Ours w/o Transfer and Stage quadrant produce them in all 250, which is the fingerprint of a model emitting essentially arbitrary geometry. Detection counts vary by a factor of fifteen across checkpoints from the same 1,000 initial proposals, which is itself a measure of how differently these variants score their outputs.

Method	Detections	No det.	Degen. box	Extreme AR	Crashes
Ours full	61,876	0	111	185	0
Ours w/o Manipulation	54,986	0	36	64	0
Ours w/o Transfer	225,380	0	250	0	0
Stage enumeration	74,712	0	244	223	0
Stage quadrant	15,597	0	250	0	0

Table 6. Failure modes over the 250-image test split. Columns count images affected. No box fell outside the image or covered the whole image for any checkpoint, so those columns are omitted.

5 Discussion

5.1 What was easy

The repository clones and runs. The config layout is readable and the three training stages map onto it cleanly. DENTEX downloads from HuggingFace without credentials. The diffusion detector trained without any instability at a reduced iteration count, which is not something every detector does. Inference is cheap enough that a full 250-image evaluation with three seeds fits comfortably in a free Kaggle session, and rebuilding every table and figure in this paper from archived predictions needs no GPU at all.

5.2 What was difficult

Seven obstacles, roughly in the order a reproducer will hit them.

1. **No released weights.** Every number requires retraining a three-stage pipeline from scratch. This is the single largest barrier and it is what turned a verification exercise into a compute project.
2. **Vendored, modified dependencies.** The repository ships patched copies of detectron2 and pycocotools. Installing either from PyPI over the checkout silently breaks things.
3. **Hardcoded absolute paths and no converter.** The configs reference paths from the authors' machines, and the DENTEX-to-COCO conversion is not in the release. We wrote one; it is in our repository.
4. **The test split does not match the reported label set.** Discussed in Section 3.2. The file the config expects is unpublished.
5. **Two contradictory dataset licences.** CC BY-SA on GitHub, CC BY-NC-SA on HuggingFace.
6. **Dead inference-time injection path.** Commented out, pointing at unpublished paths.
7. **Unrunnable cross-timestep ensembling.** Variable names from the single-label upstream survive into the multi-label fork, and the aggregate is then used as both a tensor and a per-tier list.

Items 4, 5, 6 and 7 are all cases where the released artifact and the described method have drifted apart. None of them are unusual, and all of them cost a reproducer time that a short errata note would have saved.

Our own failure — We should be even-handed, so: the worst reproducibility defect in this study is ours. Training run records were written into Kaggle sessions and never carried into the archive, so we can report measured evaluation compute and cannot report measured training compute (Section 3.5). We are asking readers to take a 22 GPU-hour figure on trust, which is exactly the thing this journal exists to discourage. Anyone repeating this should retain run records at write time rather than at the end.

5.3 Communication with original authors

This reproduction used public artifacts only: the arXiv preprint, the GitHub repository, and the HuggingFace dataset release. No material was obtained from the authors, and no result reported here depends on private correspondence. Two questions would benefit from an answer and we record them so they can be resolved in open review:

1. Can `test_merged_disease_coco3class.json`, or its class definition, be released? Without it the reported diagnosis numbers are not recomputable from public data.
2. Which licence governs DENTEX? The repository and the dataset card disagree.

5.4 What a real reproduction would take

We deliberately do not give a GPU-hour estimate for a full reproduction. We did not retain training run records, and extrapolating linearly from a 900-iteration run to a full schedule would be a guess wearing the clothes of a measurement. What we can say concretely is what a next attempt should do differently: train a base DiffusionDet control so C1 becomes testable, retain intermediate checkpoints so the ablation ordering can

be checked against training progress, vary the training seed rather than only the inference seed[13], and restratify the clean/stress split on a property that does not enter the AP denominator.
Our contribution to that attempt is the pipeline rather than the numbers. The converter, configs, evaluation harness and manifest are released, and the evaluation half runs on a CPU session for free. The numbers in this paper are what a heavily truncated training budget buys, reported honestly so that nobody mistakes them for a verdict on the method.

5.5 Not a clinical claim

Every artifact here was produced under a severely constrained compute budget for research purposes. None of it is validated for clinical use, and the accuracy levels reported above should make clear why.

References

1. I. E. Hamamci, S. Er, E. Simsar, A. Sekuboyina, M. Gundogar, B. Stadlinger, A. Mehl, and B. Menze. "Diffusion-Based Hierarchical Multi-Label Object Detection to Analyze Panoramic Dental X-rays." In: **Medical Image Computing and Computer Assisted Intervention (MICCAI 2023)**. Cham: Springer Nature Switzerland, 2023, pp. 389–399.
2. I. E. Hamamci et al. "DENTEX: An Abnormal Tooth Detection with Dental Enumeration and Diagnosis Benchmark for Panoramic X-rays." In: **arXiv preprint arXiv:2305.19112** (2023).
3. S. Chen, P. Sun, Y. Song, and P. Luo. "DiffusionDet: Diffusion Model for Object Detection." In: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**. 2023, pp. 19830–19843.
4. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. "Focal Loss for Dense Object Detection." In: **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**. 2017, pp. 2980–2988.
5. S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." In: **Advances in Neural Information Processing Systems (NeurIPS)**. Vol. 28. 2015.
6. N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. "End-to-End Object Detection with Transformers." In: **European Conference on Computer Vision (ECCV)**. 2020, pp. 213–229.
7. Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu. "SimMIM: A Simple Framework for Masked Image Modeling." In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. 2022, pp. 9653–9663.
8. K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition." In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. 2016, pp. 770–778.
9. T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature Pyramid Networks for Object Detection." In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. 2017, pp. 2117–2125.
10. J. Ho, A. Jain, and P. Abbeel. "Denoising Diffusion Probabilistic Models." In: **Advances in Neural Information Processing Systems (NeurIPS)**. Vol. 33. 2020, pp. 6840–6851.
11. J. Song, C. Meng, and S. Ermon. "Denoising Diffusion Implicit Models." In: **International Conference on Learning Representations (ICLR)**. 2021.
12. P. Micikevicius et al. "Mixed Precision Training." In: **International Conference on Learning Representations (ICLR)** (2018).
13. X. Bouthillier et al. "Accounting for Variance in Machine Learning Benchmarks." In: **Proceedings of Machine Learning and Systems (MLSys)**. Vol. 3. 2021, pp. 747–769.
14. Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. **Detectron2**. https://github.com/facebookresearch/detectron2. 2019.
15. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft COCO: Common Objects in Context." In: **European Conference on Computer Vision (ECCV)**. 2014, pp. 740–755.
16. A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: **Advances in Neural Information Processing Systems (NeurIPS)**. Vol. 32. 2019.
17. D. Hendrycks and T. Dietterich. "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations." In: **International Conference on Learning Representations (ICLR)** (2019).