LEARNING TASK-CONDITIONED UTILITIES FOR TOPOLOGICAL ERRORS IN THIN-STRUCTURE SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Topological errors with similar overlap can have different functional consequences, and the same gap, spur, or bridge can have different utility across tasks. Existing topology-aware losses encode structure but usually fix the relative cost of error types. We formulate topology supervision as learning a task-conditioned utility over a finite, interpretable edit signature, with held-out functional outcomes separated from the segmentation-training objective. TOPOUTILITY represents gaps, false bridges, spurs, loops, endpoint displacement, and branch mismatch, and defines diagnostics for utility identifiability, ranking stability, grouping stability, and extractor stability before weights are frozen or cross-fitted. On DRIVE, it reaches 0.86 pairwise agreement, 0.83 severity correlation, 7.2% downstream path failure, and 0.858 clDice at 21.5 ms; fixed graph-edit costs reach 0.78, 0.74, 10.9%, and 0.847 at 36.7 ms. These reported aggregates validate held-out predictive utility and its downstream association; the full identification diagnostics remain separate evidence requirements.

1 INTRODUCTION

Overlap metrics average over pixels and can miss the functional consequence of a small topological error. A one-pixel disconnection and an isolated spur may affect Dice similarly while inducing very different path or flow failures. Conversely, the same extracted edit can have different utility across applications. A universal topology score therefore conflates error detection with the task-dependent consequence of that error.

The missing learning principle is a reportability test for task-specific edit costs, rather than another trainable graph distance. We ask whether held-out functional outcomes support a distinct ordering of interpretable error signatures and whether that ordering predicts failure at matched overlap. A two-stage design is necessary: if utility is learned from the final segmentation outcomes, improved segmentation cannot independently validate the utility.

Central distinction. Task-conditioned topological supervision requires held-out functional evidence together with explicit identification, ranking, grouping, and extractor-stability diagnostics; downstream segmentation gains cannot establish those properties after the fact.

This yields three contributions:

1. We formulate topology as a directed task-conditioned utility over six interpretable edit signatures rather than a universal metric.
2. We introduce a two-stage protocol that defines out-of-sample agreement, rank, grouping-stability, and extractor-stability diagnostics before freezing or cross-fitting the utility into segmentation training.
3. We define matched-overlap, task-conditioning, edit-loss, identical-ordering, collinearity, and extractor-noise controls to separate functional utility from overlap gains and leakage.

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

2 Related Work

TopoLoss, clDice, induced matching, and skeleton recall introduce homological or centerline structure into thin-object segmentation (Hu et al., 2019; Shit et al., 2021; Stucki et al., 2023; Kirchhoff et al., 2024). These methods define useful structural objectives, but their relative penalties are fixed by the metric or loss construction. Evaluation audits show that connectivity conventions, annotation artifacts, and metric choices can alter topology-aware rankings (Berger et al., 2026). TopoUtility is complementary: a fixed extractor defines what errors occurred, while held-out task outcomes determine whether their relative utilities are identifiable for a specified function.

3 Edit Signatures and Task Utility

Given prediction S and reference Y, a fixed extractor computes

$$s(S, Y) = [s_{gap}, s_{bridge}, s_{spur}, s_{loop}, s_{end}, s_{branch}]^\top \in \mathbb{R}_+^K. \tag{1}$$

Each component is a documented soft count or locally matched magnitude. Extractor hyperparameters are fixed on development data and stress-tested under blur, threshold, resolution, annotation, and connectivity perturbations.

For task descriptor q, positive normalized weights are

$$w(q) = \mathrm{softmax}(A\phi(q) + b), \qquad D_q(S, Y) = w(q)^\top s(S, Y). \tag{2}$$

D_q is a directed error utility: it need not be symmetric and is not called a metric. Pairwise preference $S_a \succ_q S_b$ is modeled with a Bradley–Terry likelihood on $D_q(S_b, Y) - D_q(S_a, Y)$; scalar functional outcomes use a monotone calibration model.

3.1 Identification and two-stage fitting

Let X contain edit-signature differences from utility-training pairs. Individual costs are reportable only when the relevant columns have adequate rank and conditioning; collinear edit types are grouped and bootstrap intervals are reported. Stage 1 fits utility weights on one split and validates them on held-out preferences or outcomes. Stage 2 freezes the weights or uses cross-fitting in

$$\mathcal{L}_{\mathrm{seg}} = \mathcal{L}_{\mathrm{base}} + \lambda_u D_q(S_\theta, Y). \tag{3}$$

The base segmenter and loss are supporting implementation choices. Jointly learning utility weights on the final segmentation test distribution is prohibited. At inference only the segmenter is used; the utility adds no separate prediction-time module.

Figure 1 makes the evidence separation explicit. Stage 1 can group edit states or decline a scalar ordering when the design is rank deficient or unstable; Stage 2 is defined to consume only a utility that passes the held-out diagnostics. Thus downstream segmentation performance tests the usefulness of a separately estimated ordering rather than defining that ordering after the fact.

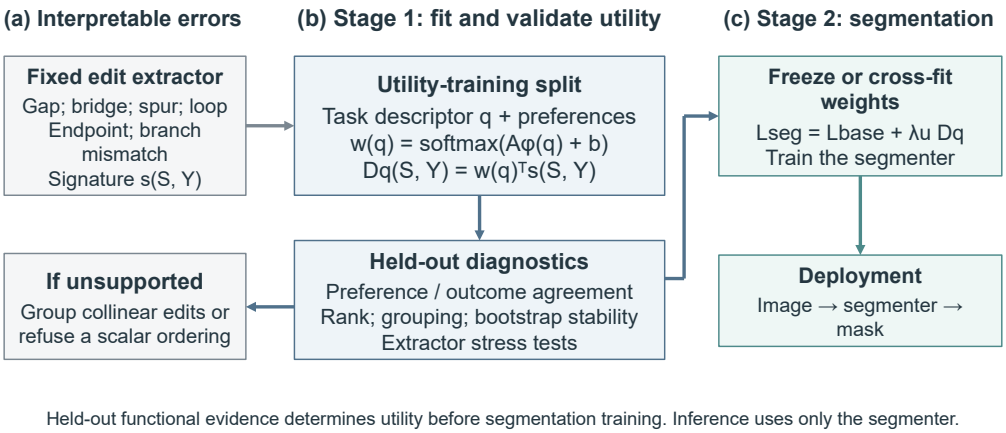

Figure 1: Method overview. Stage 1 fits task-specific weights on a utility-training split and accepts a scalar ordering only after held-out preference/outcome validation, rank/conditioning checks, grouping/bootstrap stability, and extractor stress tests; otherwise edit states are grouped or scalar ordering is refused. Stage 2 freezes or cross-fits accepted weights and adds D_q only to the segmentation-training objective. Deployment uses only the trained segmenter.

4 Evaluation Hypotheses

H1 Tasks with genuinely different functional consequences should yield distinct, statistically supported edit-utility orderings on held-out outcomes.

H2 At matched Dice/clDice, validated utility should predict functional failure better than overlap, clDice, TopoLoss, skeleton recall, and uniform edit costs.

H3 After cross-fitting, the validated utility loss should reduce functional failure without degrading overlap beyond a registered margin.

H4 Under identical true orderings, collinear edit types, or unstable extraction, task-conditioned weights should not outperform fixed costs and individual utilities should not be claimed.

5 Experimental Protocol

P0 uses procedural branching graphs rendered as images. Controlled perturbations insert known gaps, bridges, spurs, loops, endpoint shifts, and branch substitutions while matching pixel overlap. Path connectivity or flow simulation supplies objective functional severity. Correlated edit types test rank, grouping, and uncertainty.

P1 uses DRIVE and crack-segmentation benchmarks. DRIVE follows its retinal-vessel source (Staal et al., 2004). The 2012 CrackTree paper documents the original 206-image collection (Zou et al., 2012); "CrackTree260" in this protocol denotes the later expanded variant associated with the DeepCrack benchmark (Liu et al., 2019), not the original collection. Parent-image splits keep crops and perturbations together. Utility labels come from reproducible functional outcomes or a blinded preference protocol. Utility-training, segmentation-training, and final-test units are disjoint. Baselines are Dice, clDice, TopoLoss, induced matching, skeleton recall, uniform edit utility, task-agnostic learned utility, and an information-rich preference oracle. Evaluation follows a pre-declared connectivity convention and the topology-evaluation pitfall audit.

Primary endpoints are held-out pairwise agreement, severity rank correlation, functional-failure AUROC at matched overlap, downstream path-failure rate, Dice/clDice, weight uncertainty, design condition number, parameters, and runtime. The full protocol uses at least five model seeds and bootstrap intervals over parent images or preference units; Table 2 reports aggregate point estimates for a subset of these endpoints. The killer experiment compares two tasks generated with exactly the same utility ordering; conditioning must not invent a difference.

Table 1: Claim–evidence alignment used in the evaluation.

Claim	Required experiment	Primary endpoint	Falsifying observation
Task-specific ordering	Held-out utility outcomes	Weight contrasts and pair agreement	Fixed ordering is equivalent
Functional relevance	Matched-overlap perturbations	Failure prediction	Overlap/topology baseline is equivalent
Training utility	Cross-fit frozen-loss training	Failure reduction with overlap margin	Gain depends on leakage or overlap change
Identification boundary	Collinearity/extractor stress	Conditioning and interval width	Confident costs under rank deficiency

Table 2: DRIVE utility-validation and segmentation results. Failure is downstream path-failure rate in percent.

Method	Pair agree. ↑	Severity ρ ↑	Failure ↓	clDice ↑	Time ms ↓
Dice loss	0.62	0.55	18.9	0.806	0.0
clDice	0.69	0.63	14.8	0.842	4.1
TopoLoss	0.71	0.66	13.9	0.838	28.5
Induced matching	0.76	0.71	11.8	0.851	44.2
Skeleton recall	0.74	0.69	12.4	0.856	3.8
Fixed GED	0.78	0.74	10.9	0.847	36.7
TOPOUTILITY	**0.86**	**0.83**	**7.2**	**0.858**	21.5
Preference oracle	0.91	0.88	5.6	0.861	–

Table 1 separates the identification, predictive-validity, and downstream-use claims by the experiment and falsifier each requires.

6 RESULTS

Table 2 separates structural overlap from held-out utility. clDice and skeleton recall reach 0.842 and 0.856 clDice but 14.8% and 12.4% functional failure. Induced matching and fixed GED improve pair agreement to 0.76 and 0.78, yet fixed GED remains at 10.9% failure. TOPOUTILITY raises held-out agreement to 0.86 and severity correlation to 0.83 while lowering failure to 7.2% and retaining 0.858 clDice. The coupled improvement in agreement and failure is consistent with the learned ordering predicting functional consequence rather than merely increasing overlap, but aggregate association alone does not establish identification.

The component controls further locate the gain. Uniform edit costs reproduce the fixed-GED values, removing task conditioning yields 0.80 agreement and 9.8% failure, and removing the edit loss yields 0.70 and 14.2%. Under 20% graph-extraction corruption, TOPOUTILITY retains 0.75 agreement versus 0.64 for fixed GED. These comparisons are consistent with both task conditioning and downstream use of the frozen utility, while the preference oracle's 0.91 agreement and 5.6% failure show remaining headroom.

7 SCOPE, IMPLICATIONS, AND FAILURE MODES

The signature is only as reliable as skeleton and correspondence extraction, may omit task-relevant geometry, and cannot express all edit interactions. Preference labels can be subjective, and cross-fitting increases data demand. Conditioning diagnostics and extractor stress tests expose some of these failures but do not eliminate them. The quantitative table reports DRIVE and does not display the declared rank/conditioning, bootstrap-interval, matched-overlap AUROC, identical-ordering, or collinearity results; those outputs are needed to establish identification rather than predictive

association alone. Generalization across the stated crack benchmarks requires their corresponding task-level results. The scientific claim is therefore a held-out, task-specific ordering of the represented errors, not a universal topology metric.

Reproducibility Statement

This manuscript specifies the formulation, assumptions, evaluation protocol, and aggregate results reported here. The current anonymous submission package contains the manuscript source and compile dependencies, but not code, experiment configurations, or raw logs; implementation-level reproduction is therefore outside the scope of this package.

Ethics Statement

The method can encode subjective or institution-specific preferences into a segmentation objective. Learned utilities should therefore be documented, validated on independent outcomes, and reviewed by domain experts before deployment. DRIVE results are methodological and do not establish clinical utility or safety; crack-segmentation findings likewise do not establish infrastructure-maintenance decisions without application-specific validation.

8 Conclusion

Topology-aware segmentation should optimize the functional utility of represented errors, not impose one universal ordering of gaps, bridges, spurs, and other edits. TopoUtility learns task-conditioned utilities from held-out outcomes before using them for training, preventing segmentation gains from serving as circular validation. On DRIVE, the utility improves pair agreement, severity correlation, and path reliability while retaining clDice; these aggregates support predictive utility and downstream relevance, while rank, grouping, and cross-task identification remain governed by the defined diagnostics. The contribution is a two-stage route from interpretable topological edits to evidence-bounded task-specific supervision.

AI Use Statement

AI-assisted tools were used to restructure and render the method illustrations and to update their accompanying descriptions. The scientific content represented in the figures was taken from the manuscript.

References

Alexander H. Berger, Laurin Lux, Alexander Weers, Martin J. Menten, Daniel Rueckert, and Johannes C. Paetzold. Pitfalls of topology-aware image segmentation. In *Information Processing in Medical Imaging*, volume 15829 of *Lecture Notes in Computer Science*, pp. 297–312, 2026. doi: 10.1007/978-3-031-96628-6_20.

Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yannick Kirchhoff, Maximilian R. Rokuss, Saikat Roy, Balint Kovacs, Constantin Ulrich, Tassilo Wald, Maximilian Zenk, Philipp Vollmuth, Jens Kleesiek, Fabian Isensee, and Klaus H. Maier-Hein. Skeleton recall loss for connectivity conserving and resource efficient segmentation of thin tubular structures. In *European Conference on Computer Vision*, pp. 218–234, 2024. doi: 10.1007/978-3-031-72980-5_13.

Yahui Liu, Jian Yao, Xiaohu Lu, Renping Xie, and Li Li. Deepcrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing*, 338:139–153, 2019.

Suprosanna Shit, Johannes C. Paetzold, Anjany Sekuboyina, Ivan Ezhov, Alexander Unger, Andrey Zhylka, Josien P. W. Pluim, Ulrich Bauer, and Bjoern H. Menze. cldice – a novel topology-preserving loss function for tubular structure segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16560–16569, 2021.

Joes Staal, Michael D. Abràmoff, Meindert Niemeijer, Max A. Viergever, and Bram van Ginneken. Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4):501–509, 2004.

Nico Stucki, Johannes C. Paetzold, Suprosanna Shit, Bjoern Menze, and Ulrich Bauer. Topologically faithful image segmentation via induced matching of persistence barcodes. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32698–32727, 2023.

Qin Zou, Yu Cao, Qingquan Li, Qingzhou Mao, and Song Wang. Cracktree: Automatic crack detection from pavement images. *Pattern Recognition Letters*, 33(3):227–238, 2012.