PETER: Post-Training Robustification of Probabilistic Circuits

Adrian Ciotinga[1] **Yeming Dai**[1] **YooJung Choi**[1]

[1]School of Computing and Augmented Intelligence, Arizona State University

Abstract

Probabilistic circuits (PCs) can model complex joint distributions while supporting exact and efficient computation of many inference queries. However, standard likelihood-based PC learning is vulnerable to overfitting and fragile generalization when confronted with data noise, small sample sizes, or distribution shifts. This can be mitigated using distributionally-robust optimization which consider worst-case distributions within a Wasserstein ball of the empirical distribution, but current methods are limited to training a model from scratch in this framework. Instead, we propose PETER: a novel, data-free post-training framework designed to robustify pre-trained PCs against distribution shifts without retraining from scratch. Empirical evaluations across multiple density estimation benchmarks demonstrate that PETER effectively robustifies baseline models against both random and adversarial perturbations, achieving competitive or superior performance to data-dependent robust learning baselines.

Code: `https://github.com/aciotinga/PeTeR`

1 INTRODUCTION

Distributionally robust optimization (DRO) has emerged as a principled framework for learning estimators that remain reliable under data noise, limited sample sizes, and distribution shift, by hedging against an uncertainty set of plausible distributions rather than committing to the empirical distribution alone [Delage and Ye, 2010, Bertsimas and Nohadani, 2019]. A common approach is to define this uncertainty set as a Wasserstein ball around the empirical distribution, yielding a minimax formulation in which the estimator is optimized against the worst-case distribution, according to the appropriate performance measure, within some radius ϵ. Although widely studied in the context of deep generative models [Liu et al., 2023, Kuhn et al., 2019, Bauso et al., 2017], its application to tractable probabilistic models (TPMs) remains scarce; to the best of our knowledge, Peddi et al. [2022] is the sole work addressing this intersection. While effective, existing approaches to robust learning typically focus on training a model from scratch on a dataset, tying robustness to the availability of training data and requiring that any existing models be completely replaced rather than improved.

We argue that TPMs, and probabilistic circuits (PCs) in particular, are uniquely positioned for an alternative, more flexible framework of *post-training robustification*. Rather than learning a new model from scratch or fine-tuning it with additional data as classic DRO settings, post-training robustification takes a pre-trained model and grants it the same distributionally-robust guarantees within an ϵ-Wasserstein ball *without the use of any training data or complete retraining of the model.* As we show in this paper, circuit structural properties enabling tractable inference allow us to not only compute the distance between two distributions, but also to explicitly represent and optimize against the worst-case adversarial distribution efficiently. Based on this insight, we propose PETER, a post-training robustification framework for PCs that hedges against worst-case perturbations within a Wasserstein ball, bounded using the recently introduced Circuit-Wasserstein distance [Ciotinga and Choi, 2025]. Our method reduces the optimization problem to an unconstrained problem and solves it via the gradient ascent-descent algorithm, leveraging tractable gradient computation for PCs. We empirically demonstrate the efficacy of our approach in comparison to an existing robust MLE baseline for PCs that requires a training dataset, as—to the best of our knowledge—no existing methods are fully data-free.

2 PRELIMINARIES

We use capital letters (X) to denote random variables and lowercase letters (x) to denote their assignments. Boldface

denotes a set of random variables and their assignments respectively (e.g., $\mathbf{X}$ and $\mathbf{x}$).

Probabilistic Circuits A probabilistic circuit (PC) [Choi et al., 2020] C over a set of variables $\mathbf{X}$ is a parameterized, rooted directed acyclic graph (DAG) with three types of nodes: sum, product, and input nodes. Each input node n is associated with function f_n that encodes a probability distribution over a variable $X_i \in \mathbf{X}$. The set of child nodes for an internal node (sum or product) n is denoted $\mathsf{ch}(n)$, and its scope is given by $\mathsf{sc}(n) = \bigcup_{c \in \mathsf{ch}(n)} \mathsf{sc}(c)$. Each sum node n has normalized parameters $\theta_{n,c}$ for each child node c. For an assignment $\mathbf{x}$, let $\mathbf{x}_n$ denote its projection onto the scope $\mathsf{sc}(n)$ of node n. Then a PC rooted at node n recursively defines a probability distribution $p_n(\mathbf{x})$ as follows:

$$p_n(\mathbf{x}) = \begin{cases} f_n(\mathbf{x}) & \text{if } n \text{ is an input node,} \\ \prod_{c \in \mathsf{ch}(n)} p_c(\mathbf{x}_c) & \text{if } n \text{ is a product node,} \\ \sum_{c \in \mathsf{ch}(n)} \theta_{n,c} p_c(\mathbf{x}_c) & \text{if } n \text{ is a sum node.} \end{cases}$$

Probabilistic circuits admit exact and efficient computation of many probabilistic inference queries, enabled by enforcing certain structural constraints. In particular, throughout this paper we assume two properties, *smoothness* and *decomposability*, which enable tractable (polytime) computation of marginal and conditional queries. We further state when PCs are *structured-decomposable* or *compatible*; see Appendix A for the formal definitions of these properties.

Robust Maximum-Likelihood Estimation In many machine learning paradigms, one assumes access to samples drawn from an underlying but unknown distribution $\mathcal{P}(\mathbf{X})$. In the limit, the empirical distribution $\hat{P}(\mathbf{X})$ approximates $\mathcal{P}(\mathbf{X})$; however, assuming that $\hat{P} \approx \mathcal{P}$ is not always principled. If the observed data is contaminated by noise, limited to a small sample size, or subject to distribution shifts, optimizing directly over the empirical distribution can lead to severe overfitting and fragile generalization. To mitigate this, one can adopt a distributionally robust optimization framework [Delage and Ye, 2010, Bertsimas and Nohadani, 2019] that considers an uncertainty set of candidate distributions around $\hat{P}$. A popular choice to define this neighborhood is the Wasserstein distance, which safeguards the learned estimator P by optimizing against the worst-case distribution Q within an ϵ-Wasserstein ball around $\hat{P}$, yielding the following minimax formulation:

$$\max_{P} \min_{Q} \mathbb{E}_Q[\log P(\mathbf{X})]$$
$$s.t.\, \mathsf{W}(\hat{P}, Q) \leq \epsilon$$

The Wasserstein Distance Let P and Q be probability measures on metric space $\mathbb{R}^n$ with distance metric $d(\cdot, \cdot)$. The *d-Wasserstein distance* between P and Q is defined as:

$$\mathsf{W}_d(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \mathbb{E}_{\gamma(\mathbf{x},\mathbf{y})}[d(\mathbf{x}, \mathbf{y})]$$

where $\Gamma(P, Q)$ denotes the set of all *couplings*, i.e. joint distributions whose marginal distributions match P and Q. That is, the following holds for all $\gamma \in \Gamma(P, Q)$:

$$P(\mathbf{x}) = \int_{\mathbb{R}^n} \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{y}, \quad Q(\mathbf{y}) = \int_{\mathbb{R}^n} \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{x}.$$

Intuitively, guaranteeing the performance of estimator P against distributions within Wasserstein distance ϵ of $\hat{P}$ also guarantees the performance of P against similarly bounded corruptions, adversarial perturbations, or sampling noise.

3 ROBUST POST-TRAINING

We introduce *robust post-training*, a modification to robust MLE, which aims to take a pre-trained generative model encoding a baseline distribution and update it to be robust to distribution shifts. Specifically, the goal is to achieve strong performance across an uncertainty set containing all distributions within a Wasserstein distance of ϵ from the baseline distribution. Similar to robust MLE, this also safeguards our model against corrupted data or adversarial perturbations of bounded magnitude. Note that the framework does not assume any access to data—including the model's original training data, making the task generally harder than robust MLE but also more flexible and widely applicable.

To solve this for PCs, we propose PETER: a Post-Training Robustification framework that takes a pre-trained PC and updates its parameters to be robust to distribution shifts within a Wasserstein ball. Our approach is enabled by tractability of probabilistic circuits. Specifically, while computing the Wasserstein distance is #P-hard even for product distributions [Taşkesen et al., 2022], the recently introduced Circuit-Wasserstein distance CW [Ciotinga and Choi, 2025] can be computed exactly and efficiently for structured-decomposable PCs. We introduce a differentiable approach to computing CW, which we use to formulate the constrained optimization problem within an ϵ-CW ball as an unconstrained gradient descent-ascent problem.

3.1 THE OPTIMIZATION PROBLEM

In the post-training setting, we do not assume access to the original training data or its empirical distribution. Instead, the pretrained PC $\hat{P}$ serves as a proxy for the data-generating distribution, and distributions in a neighborhood around $\hat{P}$ represent plausible perturbations of this distribution. We therefore seek a robust circuit P_θ that assigns high likelihood even under the worst-case distribution in this neighborhood. Formally, we optimize

$$\max_{P_\theta} \min_{Q_\phi} \mathbb{E}_{Q_\phi}[\log P_\theta(\mathbf{X})] \quad (1)$$
$$\text{s.t. } \mathsf{CW}(\hat{P}, Q_\phi) \leq \epsilon$$

Definition 1 (Circuit-Wasserstein Distance [Ciotinga and Choi, 2025])**.** Let $P(\mathbf{X})$ and $Q(\mathbf{Y})$ be compatible PCs on a metric space with distance metric $d(\cdot,\cdot)$ and $C_\theta(\mathbf{X},\mathbf{Y})$ a coupling circuit parameterized by θ. The *d-Circuit-Wasserstein distance* is defined as:

$$\mathsf{CW}_d(P,Q) = \inf_\theta \mathbb{E}_{C_\theta(\mathbf{x},\mathbf{y})}[d(\mathbf{x},\mathbf{y})].$$

The Circuit-Wasserstein distance is a natural choice of a bounding metric because of its efficient and exact algorithm when applied to compatible circuits. Note that $\hat{P}$ being structured-decomposable implies that both P_θ and Q_ϕ are compatible with it, as all three circuits share the same structure. CW acts as an upper-bound on the true Wasserstein distance and allows us to explicitly parameterize the worst-case distribution $Q_\phi(\mathbf{X})$ using the same PC structure parameterized by ϕ.

3.2 GRADIENT-BASED ROBUST POST-TRAINING

To solve the constrained optimization problem for robust post-training, we first rewrite it as an unconstrained problem.

Proposition 1. *The optimal solution to the following unconstrained Lagrangian optimization problem is a feasible and optimal solution to the constrained problem in Equation 1.*

$$\max_{P_\theta}\min_{Q_\phi}\max_{\lambda\geq 0}\ \mathbb{E}_{Q_\phi}[\log P_\theta(\mathbf{X})] + \lambda(\mathsf{CW}(\hat{P},Q_\phi)-\epsilon).$$

Saddle-point optimization problems arise frequently in machine learning—e.g., GAN training, multi-agent reinforcement learning, etc. [Goodfellow et al., 2014, Albrecht et al., 2024]—and can be solved using the optimistic gradient ascent-descent algorithm [Daskalakis et al., 2018]. Our problem admits the following update rules with learning rates $\eta_\theta, \eta_\phi, \eta_\lambda$:

$$\begin{aligned}
\theta_{t+1} \leftarrow& \theta_t + \eta_\theta\Big(2\nabla_\theta\,\mathbb{E}_{Q_\phi}[\log P_{\theta_t}(\mathbf{X})] \\
&- \nabla_\theta\,\mathbb{E}_{Q_\phi}[\log P_{\theta_{t-1}}(\mathbf{X})]\Big) \\
\phi_{t+1} \leftarrow& \phi_t - \eta_\phi\Big(2\nabla_\phi\big(\mathbb{E}_{Q_{\phi_t}}[\log P_\theta(\mathbf{X})] + \lambda\mathsf{CW}(\hat{P},Q_{\phi_t})\big) \\
&- \nabla_\phi\big(\mathbb{E}_{Q_{\phi_{t-1}}}[\log P_\theta(\mathbf{X})] + \lambda\mathsf{CW}(\hat{P},Q_{\phi_{t-1}})\big)\Big) \\
\lambda_{t+1} \leftarrow& \max(0, \lambda_t + \eta_\lambda[\mathsf{CW}(\hat{P},Q_\phi)-\epsilon])
\end{aligned}$$

If P_θ and Q_ϕ are compatible and deterministic, the necessary gradients can be computed exactly and efficiently in a single forward and backward pass; however, without determinism, computing $\mathbb{E}_{Q_\phi}[\log P_\theta(\mathbf{X})]$ is #P-hard [Vergari et al., 2021]. To circumvent this, we estimate $\mathbb{E}_{Q_\phi}[\log P_\theta(\mathbf{X})]$ by sampling from Q_ϕ when computing gradients w.r.t. θ, and apply Jensen's inequality to bound $\mathbb{E}_{Q_\phi}[\log P_\theta(\mathbf{X})] \leq \log \mathbb{E}_{Q_\phi}[P_\theta(\mathbf{X})]$ and update ϕ to instead minimize this upper-bound—which we can compute exactly thanks to the compatibility of P_θ and Q_ϕ.

3.3 DIFFERENTIABLE COMPUTATION OF CW

While Ciotinga and Choi [2025] introduced a recursive algorithm computing CW between compatible PCs, computing its gradients is nontrivial because the standard forward pass relies on linear program solvers, which are not inherently compatible with backpropagation. To address this, we propose a differentiable framework for CW that allows efficient computation of $\nabla_\theta\mathsf{CW}(P_\theta,Q_\phi)$ and $\nabla_\phi\mathsf{CW}(P_\theta,Q_\phi)$. A rigorous derivation can be found in Appendix B.

CW is computed recursively by traversing circuits P_θ and Q_ϕ, where the nature of the computation—and thus the gradient flow—at each step is dependent on the node type. Letting n and m be corresponding nodes in P_θ and Q_ϕ respectively, we have three cases:

- **Sum Nodes:** The distance is formulated as an optimal transport problem between the children of n and m, subject to the constraints imposed by the mixture weights θ and ϕ. By applying the Envelope Theorem to the dual formulation of this linear program, we can compute the partial derivatives of the distance with respect to both the children's distances and the mixture weights: $\frac{\partial\mathsf{CW}(n,m)}{\partial\theta_i} = x_i^*$ and $\frac{\partial\mathsf{CW}(n,m)}{\partial\phi_j} = y_j^*$.
- **Product Nodes:** The distance is defined as the unweighted sum of distances between corresponding children, leading to an identity gradient.
- **Input Nodes:** We assume the distance between base distributions is differentiable, allowing for efficient computation of $\nabla_\theta\mathsf{W}(n,m)$ and $\nabla_\phi\mathsf{W}(n,m)$. This holds for standard families such as Gaussian or categorical distributions.

4 EXPERIMENTS

In this section, we empirically evaluate PETER on the density estimation benchmark datasets [Lowd and Davis, 2010, Van Haaren and Davis, 2012, Bekker et al., 2015, Larochelle and Murray, 2011], a set of binary datasets ranging from 16 to 1000 variables. We use Hidden Chow-Liu Tree (HCLT) [Liu and Van den Broeck, 2021] implemented with PyJuice [Liu et al., 2024] to learn the PC structure for each dataset and estimate the parameters using PETER and two other baselines for comparison.

Baselines We compare against the robust MLE method (RL-TPM) proposed by Peddi et al. [2022] and standard maximum-likelihood estimation via expectation-maximization (MLE-PC) [Desana and Schnörr, 2016]. RL-TPM, designed for binary datasets, poses robust MLE as a minimax game between the PC estimator maximizing likelihood and an adversary corrupting bits with a fixed budget to minimize the estimator's likelihood. Its corruption budget ϵ corresponds to the maximum Hamming distance a data

		$\mathcal{T}$			$\mathcal{T}_a$			$\mathcal{T}_r$		
Dataset	ϵ	MLE-PC	RL-TPM	**PETER**	MLE-PC	RL-TPM	**PETER**	MLE-PC	RL-TPM	**PETER**
nltcs	1	**-6.09**	-9.35	-6.65	-11.26	-11.08	**-9.69**	-8.40±0.02	-10.12±0.02	**-8.03**±0.02
	3		-10.14	-8.13	-18.43	-12.90	**-11.55**	-11.45±0.03	-11.26±0.01	**-9.79**±0.01
	5		-10.88	-10.73	-23.02	-12.63	**-11.81**	-13.24±0.05	-11.45±0.02	**-11.00**±0.01
msnbc	1	**-6.43**	-9.31	-6.88	-12.01	-12.37	**-9.81**	-8.32±0.01	-10.11±0.01	**-8.06**±0.00
	3		-8.54	-8.26	-19.88	-11.99	**-11.86**	-11.28±0.01	-10.25±0.01	**-9.81**±0.00
	5		-9.98	-10.20	-25.39	-12.23	**-12.07**	-13.51±0.01	**-10.98**±0.00	-11.03±0.00
plants	1	**-14.46**	-21.16	-14.96	-23.39	-24.92	**-22.81**	-18.80±0.04	-23.80±0.03	**-18.71**±0.03
	3		-22.33	-15.95	-38.81	-31.55	**-31.38**	-26.25±0.06	-28.73±0.03	**-23.77**±0.04
	5		-23.49	-16.74	-52.40	-36.90	**-34.73**	-32.39±0.10	-32.48±0.06	**-27.60**±0.05
bnetflix	1	**-57.59**	-58.82	-57.84	-61.13	-60.93	**-60.91**	**-58.27**±0.02	-59.38±0.02	-58.48±0.02
	3		-59.29	-58.44	-67.12	**-63.25**	-65.62	**-59.54**±0.03	-60.57±0.02	-60.00±0.03
	5		-60.08	-59.21	-72.17	**-65.56**	-68.58	**-60.77**±0.05	-61.87±0.03	-61.38±0.04
dna	1	**-79.97**	-82.66	-80.09	-87.68	-86.85	**-86.78**	-83.48±0.08	-84.65±0.04	**-83.04**±0.06
	3		-84.31	-80.90	-102.88	**-93.76**	-96.24	-90.38±0.15	-88.81±0.07	**-87.13**±0.09
	5		-85.90	-84.10	-117.82	**-99.05**	-103.31	-97.13±0.19	-92.09±0.05	**-91.11**±0.07
tmovie	1	**-52.81**	-54.55	-53.56	-61.93	**-60.69**	-63.71	**-58.53**±0.07	-59.16±0.07	-58.97±0.08
	3		-56.94	-56.43	-79.51	**-72.46**	-81.09	-69.74±0.22	**-68.66**±0.11	-69.78±0.16
	5		-59.30	-58.35	-96.75	**-82.62**	-90.06	-80.77±0.24	-77.06±0.09	**-76.93**±0.13
bbc	1	**-250.20**	-250.72	-250.88	-258.72	**-255.37**	-258.01	**-253.05**±0.06	-253.43±0.06	-253.65±0.06
	3		-251.45	-253.62	-275.07	**-263.44**	-271.64	**-258.80**±0.11	-259.31±0.13	-261.51±0.11
	5		-252.51	-254.88	-290.24	**-271.09**	-283.80	**-264.30**±0.20	-265.01±0.19	-267.50±0.18

Table 1: Test-set log-likelihood. $\epsilon \in \{1, 3, 5\}$: corruption budget for test-set generation. $\mathcal{T}$: original test set; $\mathcal{T}_a$: adversarially-perturbed test set; $\mathcal{T}_r$: average over 10 randomly-perturbed test sets. Boldface indicates best approach, underline second best.

point may be moved, subsequently bounding the Wasserstein distance between the original and corrupted datasets by ϵ. Lastly, we use MLE-PC as our base distribution $\hat{P}$.

Evaluation Sets For each dataset, we construct three test sets to evaluate each method: (1) the original test set $\mathcal{T}$; (2) an adversarially-perturbed test set $\mathcal{T}_a$, where each data point receives ϵ corruptions applied greedily to maximize the drop in likelihood of the given PC; and (3) $\mathcal{T}_r$ generated by randomly corrupting each data point ϵ times.

Results Table 1 summarizes the results. As we expect, the distributionally robust methods achieve slightly worse likelihood than MLE-PC given the unperturbed test set $\mathcal{T}$; nevertheless, we observe that PETER consistently outperforms RL-TPM on $\mathcal{T}$. PETER also consistently outperforms RL-TPM on the randomly-perturbed test set $\mathcal{T}_r$, which aims to capture an "average-case" corruption rather than a single worst case w.r.t. MLE-PC. Lastly, PETER outperforms RL-TPM on the majority of the adversarially-perturbed datasets. We note that instances where PETER is outperformed tend to be at higher corruption levels, likely due to the CW-distance being a looser upper bound on the W-distance than the Hamming distance used by RL-TPM is; this results in the true Wasserstein ball that PETER optimizes for being smaller than the one RL-TPM does, so the true worst-case distribution—typically lying on the edge of the ball—may be outside of the ϵ-CW ball. We provide empirical evidence supporting this hypothesis in Appendix D. Lastly, despite the strict ϵ-CW constraint, PETER remains highly stable during training, as discussed further in Appendix C.

5 CONCLUSION

We introduced PETER, a data-free post-training framework that robustifies probabilistic circuits against distribution shifts within a Wasserstein ball without requiring access to any training data or retraining a circuit from scratch. By leveraging the Circuit-Wasserstein distance and deriving gradients for its computation, we formulated robust post-training within a Wasserstein ball as an unconstrained gradient descent-ascent problem that can be solved efficiently for structured-decomposable PCs. Empirically, PETER consistently outperforms the data-dependent baseline across both unperturbed and perturbed test sets, particularly at lower corruption budgets, while maintaining stable training despite the non-smooth nature of Circuit-Wasserstein gradients. These results suggest that post-training robustification offers a practical and flexible alternative to training robust models from scratch and motivate further development of optimal transport-based losses for PCs.

Acknowledgements

This work was supported in part by the AFOSR grant FA9550-25-1-0320. The authors would like to thank Stefan Wagner for helpful discussions on implementation and hyperparameter tuning.

References

Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. URL https://www.marl-book.com.

Dario Bauso, Jian Gao, and Hamidou Tembine. Distributionally robust games: f-divergence and learning. In *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*, pages 148–155, 2017.

Jessa Bekker, Jesse Davis, Arthur Choi, Adnan Darwiche, and Guy Van den Broeck. Tractable learning for complex probability queries. *Advances in Neural Information Processing Systems*, 28, 2015.

Dimitris Bertsimas and Omid Nohadani. Robust maximum likelihood estimation. *INFORMS Journal on Computing*, 31(3):445–458, 2019.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models, oct 2020. URL http://starai.cs.ucla.edu/papers/ProbCirc20.pdf.

Adrian Ciotinga and YooJung Choi. Optimal transport for probabilistic circuits. In *The 41st Conference on Uncertainty in Artificial Intelligence*, 2025. URL https://openreview.net/forum?id=lNau87jgPx.

Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training GANs with optimism. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJJySbbAZ.

Erick Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations research*, 58(3):595–612, 2010.

Mattia Desana and Christoph Schnörr. Expectation maximization for sum-product networks as exponential family mixture models, 04 2016.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/f033ed80deb0234979a61f95710dbe25-Paper.pdf.

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, July 2014. URL https://starai.cs.ucla.edu/papers/KisaKR14.pdf.

Daniel Kuhn, Peyman Mohajerin Esfahani, Viet Anh Nguyen, and Soroosh Shafieezadeh-Abadeh. Wasserstein distributionally robust optimization: Theory and applications in machine learning. In *Operations research & management science in the age of analytics*, pages 130–166. Informs, 2019.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286, December 2012. ISSN 1935-8245. doi: 10.1561/2200000044. URL http://dx.doi.org/10.1561/2200000044.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 29–37. JMLR Workshop and Conference Proceedings, 2011.

Anji Liu and Guy Van den Broeck. Tractable regularization of probabilistic circuits. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3558–3570. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/1d0832c4969f6a4cc8e8a8fffe083efb-Paper.pdf.

Anji Liu, Kareem Ahmed, and Guy Van den Broeck. Scaling tractable probabilistic circuits: A systems perspective, 2024. URL https://arxiv.org/abs/2406.00766.

Jiashuo Liu, Zheyan Shen, Peng Cui, Linjun Zhou, Kun Kuang, and Bo Li. Distributionally robust learning with stable adversarial training. *IEEE Trans. on Knowl. and Data Eng.*, 35(11):11288–11300, November 2023. ISSN 1041-4347. doi: 10.1109/TKDE.2022.3224056. URL https://doi.org/10.1109/TKDE.2022.3224056.

Daniel Lowd and Jesse Davis. Learning markov network structure with decision trees. In *2010 IEEE International Conference on Data Mining*, pages 334–343. IEEE, 2010.

Rohith Peddi, Tahrima Rahman, and Vibhav Gogate. Robust learning of tractable probabilistic models. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 1572–1581. PMLR, 01–05 Aug 2022. URL https://proceedings.mlr.press/v180/peddi22a.html.

Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 630–645, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-44851-9.

Bahar Taşkesen, Soroosh Shafieezadeh-Abadeh, Daniel Kuhn, and Karthik Natarajan. Discrete optimal transport with independent marginals is #p-hard, 2022. URL https://arxiv.org/abs/2203.01161.

Jan Van Haaren and Jesse Davis. Markov network structure learning: A randomized feature generation approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1148–1154, 2012.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference. In *Advances in Neural Information Processing Systems*, volume 34, pages 13189–13201, 2021.

PETER: Post-Training Robustification of Probabilistic Circuits (Supplementary Material)

Adrian Ciotinga[1] **Yeming Dai**[1] **YooJung Choi**[1]

[1]School of Computing and Augmented Intelligence, Arizona State University

A PROBABILISTIC CIRCUITS BACKGROUND

Modeling probability distributions in a way that enables tractable computation of queries is of great interest to the machine learning community. Thus, many classes of tractable probabilistic models have been proposed in the literature; probabilistic sentential decision diagrams [Kisa et al., 2014], cutset networks [Rahman et al., 2014], and determinental point processes [Kulesza and Taskar, 2012], represent just a few. Probabilistic circuits (PCs) [Choi et al., 2020] provide a unifying framework for representing many classes of tractable probabilistic models as computational graphs; within this framework, tractability of certain queries can be guaranteed by imposing structural properties on the computational graph of the circuit. This includes tractable marginal and conditional inference, as well as pairwise queries that compare two distributions such as Kullback-Leibler (KL) divergence and cross-entropy [Vergari et al., 2021].

Now, we formally establish the core structural properties of probabilistic circuits that dictate their tractability for various probabilistic inference queries.

Definition 2. A probabilistic circuit C is **smooth** if every sum node $n \in C$ has the same scope as its children: $\forall n_i \in \mathsf{ch}(n)$, $\mathsf{sc}(n_i) = \mathsf{sc}(n)$.

Definition 3. A probabilistic circuit C is **decomposable** if the children of every product node $n \in C$ have disjoint scope: $\forall\, n_i \neq n_j \in \mathsf{ch}(n), \mathsf{sc}(n_i) \bigcap \mathsf{sc}(n_j) = \emptyset$.

Definition 4. A probabilistic circuit C is **structured-decomposable** if it is decomposable and any two product nodes with the same scope partition their scope identically: $\forall\, n, m \in P$, $\mathsf{sc}(n) = \mathsf{sc}(m) \implies \{\mathsf{sc}(n_i) \mid n_i \in \mathsf{ch}(n)\} = \{\mathsf{sc}(m_i) \mid m_i \in \mathsf{ch}(m)\}$.

Definition 5. Two probabilistic circuits C_1 and C_2 are **compatible** if they are both structured-decomposable and any product node in C_1 partitions its scope identically to any product node in C_2 with the same scope: $\forall\, n \in C_1, m \in C_2$, $\mathsf{sc}(n) = \mathsf{sc}(m) \implies \{\mathsf{sc}(n_i) \mid n_i \in \mathsf{ch}(n)\} = \{\mathsf{sc}(m_i) \mid m_i \in \mathsf{ch}(m)\}$.

Nodes n and m in compatible PCs C_1 and C_2 that share the same type (e.g., sum, product, or input) and same scope are called **corresponding** nodes.

B DIFFERENTIABLE COMPUTATION OF CW

Ciotinga and Choi [2025] detailed the recursive algorithm for computing the Circuit-Wasserstein distance, but the use of linear program solvers during the forward pass prevents the use of classic automatic differentiation techniques to compute gradients. Here, we derive the gradients for each step of the Circuit-Wasserstein algorithm. CW looks at pairs of corresponding nodes n and m in P_θ and Q_ϕ respectively; let n_i (ext. m_j) denote the i'th child of n (ext. m). There are three cases:

n and m are Sum Nodes Let θ_i and ϕ_j denote the weights of the i'th and j'th children of n and m respectively. The Circuit-Wasserstein distance between corresponding sum nodes is the solution to the discrete optimal transport problem

where the distance between child n_i and m_j is their recursively-computed Circuit-Wasserstein distance. This is formulated as a linear programming problem:

$$\mathsf{CW}(n,m) = \begin{cases} \min\limits_{w} & \sum_{i,j} \mathsf{CW}(n_i, m_j) w_{i,j} \\ \text{s.t.} & \forall i,\ \sum_j w_{i,j} = \theta_i \\ & \forall j,\ \sum_i w_{i,j} = \phi_j \\ & w_{i,j} \geq 0 \end{cases} \tag{2}$$

With solution w^* and applying the Envelope Theorem, we see that $\frac{\partial \mathsf{CW}(n,m)}{\partial \mathsf{CW}(n_i,m_j)} = w^*_{i,j}$. Now, denoting the dual variables for the θ_i and ϕ_j constraints as x_i and y_j respectively, the above linear programming problem has the equivalent dual form:

$$\mathsf{CW}(n,m) = \begin{cases} \max\limits_{x,y} & \sum_i \theta_i x_i + \sum_j \phi_j y_j \\ \text{s.t.} & \forall i,j,\ x_i + y_j \leq \mathsf{CW}(n_i, m_j) \end{cases} \tag{3}$$

Another application of the Envelope Theorem at the optimal point yields $\frac{\partial \mathsf{CW}(n,m)}{\partial \theta_i} = x^*_i$ and $\frac{\partial \mathsf{CW}(n,m)}{\partial \phi_j} = y^*_j$ respectively. Calculating $\nabla_\theta \mathsf{CW}(n,m)$ and $\nabla_\phi \mathsf{CW}(n,m)$ in this manner yields a *sub-gradient*—as the solution to a linear programming problem is non-smooth at its optimal point—leading to the computed gradient vectors being non-smooth step functions. This would typically necessitate the use of e.g., a quadratic regularizer to smooth out the gradients, but we forego this for differentiating Circuit-Wasserstein due to the unique optimization landscape of the problem effectively nullifying this discontinuity.

n and m are Product Nodes Since circuits P_θ and Q_ϕ are compatible, there is a bijective map between children n_i and m_i of n and m according to their scopes. The Circuit-Wasserstein distance between n amd m is simply an unweighted sum of the distances between corresponding children. Consequently, $\frac{\partial \mathsf{CW}(n,m)}{\partial \mathsf{CW}(n_i,m_i)} = 1$.

n and m are Input Nodes The Circuit-Wasserstein distance between input nodes is equal to the Wasserstein distance between their encoded distributions, which can be computed efficiently for many continuous and discrete families of distributions. By the assumption of tractability for input node distributions commonly made for probabilistic circuits, we assume that $\nabla_\theta \mathsf{W}(n,m)$ and $\nabla_\phi \mathsf{W}(n,m)$ can be computed efficiently. This assumption holds for many classes of distributions, included (but not limited to) categorical and Gaussian distributions.

C DETAILS OF PETER

Proposition 1. *The optimal solution to the following unconstrained Lagrangian optimization problem is a feasible and optimal solution to the constrained problem in Equation 1.*

$$\max_{P_\theta} \min_{Q_\phi} \max_{\lambda \geq 0} \mathbb{E}_{Q_\phi}[\log P_\theta(\mathbf{X})] + \lambda(\mathsf{CW}(\hat{P}, Q_\phi) - \epsilon). \tag{4}$$

Proof. Let $\theta^*, \phi^*, \lambda^*$ denote an optimal solution to 4. If $\mathsf{CW}(\hat{P}, Q_{\phi^*}) > \epsilon$, then $\lambda \to \infty$ results in the innermost maximization going to positive infinity ($+\infty$). Since the outer optimization with respect to Q_ϕ seeks to *minimize* this objective, it will strictly avoid any Q_ϕ that causes the inner maximum to diverge, provided there is at least one feasible solution (for instance, setting $Q_\phi = \hat{P}$ yields $\mathsf{CW}(\hat{P}, \hat{P}) = 0 \leq \epsilon$, which results in a finite value).

Consequently, to prevent the objective from diverging to $+\infty$, the optimal distribution must satisfy $\mathsf{CW}(\hat{P}, Q_{\phi^*}) \leq \epsilon$. This ensures that the optimal solution Q_{ϕ^*} strictly resides within the ϵ-CW ball. Thus, the optimal solution to the unconstrained Lagrangian optimization problem is guaranteed to be feasible with respect to the original constraint. □

C.1 SOLVING THE OPTIMIZATION PROBLEM

As mentioned in Section 3.2, our optimization problem admits the following update rules with learning rates $\eta_\theta, \eta_\phi, \eta_\lambda$:

$$\theta_{t+1} \leftarrow \theta_t + \eta_\theta \Big(2\nabla_\theta \, \mathbb{E}_{Q_\phi}[\log P_{\theta_t}(\mathbf{X})] - \nabla_\theta \, \mathbb{E}_{Q_\phi}[\log P_{\theta_{t-1}}(\mathbf{X})] \Big)$$

$$\phi_{t+1} \leftarrow \phi_t - \eta_\phi \Big(2\nabla_\phi \big(\mathbb{E}_{Q_{\phi_t}}[\log P_\theta(\mathbf{X})] + \lambda \mathsf{CW}(\hat{P}, Q_{\phi_t}) \big) - \nabla_\phi \big(\mathbb{E}_{Q_{\phi_{t-1}}}[\log P_\theta(\mathbf{X})] + \lambda \mathsf{CW}(\hat{P}, Q_{\phi_{t-1}}) \big) \Big)$$

$$\lambda_{t+1} \leftarrow \max(0, \lambda_t + \eta_\lambda [\mathsf{CW}(\hat{P}, Q_\phi) - \epsilon])$$

While differentiable Circuit-Wasserstein computation is straightforward, exactly computing $\mathbb{E}_{Q_\phi}[\log P_\theta(\mathbf{X})]$ and differentiating it with respect to both θ and ϕ poses challenges. Without determinism admitting exact and efficient computation, we estimate this term by drawing samples from Q_ϕ and differentiating this estimation to estimate $\nabla_\theta \, \mathbb{E}_{Q_\phi}[\log P_\theta(\mathbf{X})]$. To avoid needing to differentiably sample from Q_ϕ, we apply Jensen's inequality to upper-bound the expectation by $\mathbb{E}_{Q_\phi}[\log P_\theta(\mathbf{X})] \leq \log \mathbb{E}_{Q_\phi}[P_\theta(\mathbf{X})]$, and utilize the tractability of expectation queries to exactly and differentiably optimize this upper-bound. Specifically, we define surrogate functions $f(\theta, \phi)$ and $g(\theta, \phi)$ to express these objectives:

$$f(\theta, \phi) = \frac{1}{n} \sum_i \log P_\theta(\mathbf{x}_i), \quad \mathbf{x}_1, ..., \mathbf{x}_n \overset{\text{i.i.d.}}{\sim} Q_\phi(\mathbf{X})$$

$$g(\theta, \phi) = \log \mathbb{E}_{Q_\phi}[P_\theta(\mathbf{X})] + \lambda \mathsf{CW}(\hat{P}, Q_\phi)$$

We finally present Algorithm 1, which updates the parameters θ and ϕ by applying Optimistic Gradient Descent-Ascent to these surrogate objectives.

Algorithm 1 PETER$(\hat{P}, \epsilon)$: returns parameters θ that make $\hat{P}$ robust to distributions within CW-ball of radius ϵ

$P_\theta, Q_\phi \leftarrow \hat{P}$ ▷ Initialize P_θ, Q_ϕ
repeat
  $\theta_{t+1} \leftarrow \theta_t + \eta_\theta (2\nabla_\theta f(\theta_t, \phi_t) - \nabla_\theta f(\theta_{t-1}, \phi_{t-1}))$
  $\phi_{t+1} \leftarrow \phi_t - \eta_\phi (2\nabla_\phi g(\theta_t, \phi_t) - \nabla_\phi g(\theta_{t-1}, \phi_{t-1}))$
  $\lambda_{t+1} \leftarrow \max(0, \lambda_t + \eta_\lambda [\mathsf{CW}(\hat{P}, Q_\phi) - \epsilon])$
until *convergence;*

Stability of CW Gradients As noted in Section 3.3, the gradients computed for the linear programs in CW computation are actually discontinuous sub-gradients that behave like step functions. This would typically make for highly instable training—manifesting as the optimizer's inability to maintain a feasible $\mathcal{Q}_\phi$—but we observed that training within a CW-ball is actually highly stable, with our adversarial distribution Q_ϕ maintaining negligible constraint violation with nearly zero hyperparameter tuning. We attribute this to the fact that a single sum node has multiple corresponding sum nodes in the other circuit. Thus, its parameter gradients are the sum of multiple sub-gradients with offset discontinuities, mitigating the instability.

D ADDITIONAL EXPERIMENTAL DETAILS

This section provides additional details on the experimental setup and results.

Hardware and Software Packages Unless otherwise specified, all PETER runs were conducted on a machine running Rocky Linux 8.10 with 2× AMD EPYC 7713 CPUs, 2 TB DDR4 RAM, and no GPU. Multiple runs of PETER were done simultaneously across all 128 CPU cores. Our code requires SparC 0.6.1 from PyPi to perform PC parameter updates. We implement our baselines RL-TPM and MLE-PC using PyJuice 2.4.3 from PyPi. Our baseline experiments were conducted on a machine with 2x L40S GPUs.

D.1 DEBD EXPERIMENTS

All code to reproduce our results is included in our GitHub repository: `https://github.com/aciotinga/PeTeR`.

For each dataset, all PCs use the same circuit structure learned using the Hidden Chow-Liu Tree algorithm [Liu and Van den Broeck, 2021].

For MLE-PC, we run 1000 iterations of Expectation-Maximization on the uncorrupted training dataset to learn the parameters of the PC. For RL-TPM, at each corruption level $\epsilon \in \{1, 3, 5\}$, we run 1000 iterations of RL-TPM on the training dataset using the chosen hyperparameters. For PETER, at each corruption level, we run 500 iterations of PETER using the chosen hyperparameters.

Hyperparameter Tuning Our hyperparameter tuning scheme uses the Tree-structured Parzen Estimator algorithm implementation in the Optuna package. For PETER we tune the learning rate $\eta_\theta \in [10^{-8}, 10^{-1}]$ and notably the *ratio* $\frac{\eta_\phi}{\eta_\theta} \in [10^{-1}, 10^{2}]$ per dataset and per corruption level. For corruption level ϵ, we adversarially corrupt the base model's validation dataset with budget ϵ and use likelihood on this dataset after 500 iterations of PETER as our hyperparameter tuning target. We keep all other hyperparameters the same across all datasets and corruption levels.

Results and Testing Procedure We provide the full version of Table 1 including all 28 datasets at the end of this supplement in Table 4. We evaluate each method against three versions of the dataset's test set: the original test set, a randomly perturbed test set, and an adversarially perturbed test set. Random perturbations are constructed by sampling ϵ bit indices with replacement and flipping those indices; if the same bit is sampled an even number of times, no change happens. We create 10 variations of the randomly perturbed dataset per corruption level, reporting the average over the 10. Adversarial perturbations are created by greedily flipping the bit that drops the data point's likelihood according to the model being evaluated by the largest amount, with ϵ total bit flips being done. Note that this means a different adversarial dataset per PC, as the adversarial attack is model-aware.

Circuit-Wasserstein Distance of Adversarial Datasets We hypothesize that PETER performs less strongly against adversarial corruptions of higher magnitude because the corrupted data lie outside of our Circuit-Wasserstein uncertainty set (but within a Wasserstein uncertainty set of the same size ϵ). This is supported empirically; for each of the 28 DEBD datasets, we learn the parameters for the same circuit structure as earlier using 1000 iterations of expectation-maximization for the uncorrupted test set, as well as the three $\epsilon = 1, 3, 5$ adversarially corrupted test sets created for the PETER-robustified PCs. We then compute the Circuit-Wasserstein distance between the PC learned on the uncorrupted test set and each of the corrupted test sets. The result are in Table 2.

Corruption Scale ϵ	**Actual CW Distance**
1	12.08
3	17.66
5	21.37

Table 2: Circuit-Wasserstein distance between PCs learned on original and corrupted data, compared to the corruption scale of the data.

D.2 RUNTIME OF PETER

PETER is computationally cheap. On a laptop with an Intel i9-13950HX CPU, we robustify the aforementioned PCs in between a few seconds to a few minutes in most cases. See Table 3 for the detailed results.

Dataset	Iters/s	Time (min)	Dataset	Iters/s	Time (min)
ad	2.25	3.71	rcv1	26.05	0.32
voting	3.38	2.46	kosarek	27.62	0.30
bbc	4.36	1.91	plants	34.96	0.24
moviereview	4.54	1.84	tretail	36.15	0.23
cr52	4.80	1.74	baudio	42.02	0.20
c20ng	5.48	1.52	connect4	42.13	0.20
cwebkb	5.72	1.46	adult	43.75	0.19
nips	7.88	1.06	mushrooms	44.63	0.19
tmovie	8.86	0.94	bnetflix	44.68	0.19
book	9.72	0.86	jester	44.94	0.19
ocr_letters	16.06	0.52	accidents	45.48	0.18
pumsb_star	18.44	0.45	kdd	66.08	0.13
msweb	19.04	0.44	nltcs	219.12	0.04
dna	21.79	0.38	msnbc	302.16	0.03

Table 3: Runtime of PETER for PCs trained on density estimation benchmark datasets. Full PETER runs are 500 iterations.

		$\mathcal{T}$			$\mathcal{T}_a$			$\mathcal{T}_r$		
Dataset	ϵ	MLE-PC	RL-TPM	**PETER**	MLE-PC	RL-TPM	**PETER**	MLE-PC	RL-TPM	**PETER**
accidents	1		-35.71	-32.57	**-38.82**	-39.98	-42.89	**-34.51**±0.06	-38.49±0.03	-35.61±0.04
	3	**-29.91**	-37.16	-32.65	-55.91	**-47.59**	-51.31	-43.22±0.11	-43.98±0.04	**-40.08**±0.06
	5		-38.72	-33.79	-72.33	**-53.76**	-55.79	-51.19±0.11	-48.48±0.07	**-44.06**±0.06
ad	1		-18.88	-16.31	**-25.50**	-27.41	-35.05	**-24.28**±0.05	-26.41±0.08	-24.29±0.05
	3	**-16.26**	-20.64	-33.10	**-43.95**	-44.94	-85.89	**-40.26**±0.09	-41.49±0.15	-56.65±0.11
	5		-22.89	**-16.26**	**-62.41**	-63.06	-62.53	-56.14±0.14	**-55.56**±0.13	-56.14±0.14
adult	1		-21.24	-19.15	**-25.45**	-26.36	-31.76	**-23.12**±0.02	-25.01±0.01	-24.07±0.02
	3	**-16.29**	-22.50	-21.07	-43.41	**-34.77**	-37.41	-35.86±0.04	-31.67±0.01	**-30.47**±0.02
	5		-24.21	-22.42	-60.64	**-41.55**	-45.58	-47.40±0.03	-37.06±0.02	**-35.25**±0.01
baudio	1		-43.17	-40.79	-44.50	-45.60	**-44.18**	**-42.29**±0.02	-44.50±0.02	-42.30±0.02
	3	**-40.68**	-43.67	-41.19	-51.05	-49.70	**-49.39**	-45.31±0.04	-47.22±0.03	**-45.14**±0.03
	5		-44.27	-41.91	-56.72	**-53.27**	-53.64	-48.10±0.06	-49.65±0.05	**-47.72**±0.05
bbc	1		-250.72	-250.88	-258.72	**-255.37**	-258.01	**-253.05**±0.06	-253.43±0.06	-253.65±0.06
	3	**-250.20**	-251.45	-253.62	-275.07	**-263.44**	-271.64	**-258.80**±0.11	-259.31±0.13	-261.51±0.11
	5		-252.51	-254.88	-290.24	**-271.09**	-283.80	**-264.30**±0.20	-265.01±0.19	-267.50±0.18
bnetflix	1		-58.82	-57.84	-61.13	-60.93	**-60.91**	**-58.27**±0.02	-59.38±0.02	-58.48±0.02
	3	**-57.59**	-59.29	-58.44	-67.12	**-63.25**	-65.62	**-59.54**±0.03	-60.57±0.02	-60.00±0.03
	5		-60.08	-59.21	-72.17	**-65.56**	-68.58	**-60.77**±0.05	-61.87±0.03	-61.38±0.04
book	1		-39.50	-33.83	-40.31	-44.72	**-40.01**	**-38.25**±0.03	-43.88±0.04	-38.39±0.03
	3	**-33.44**	-40.28	-34.32	-53.56	-54.86	**-50.75**	-47.75±0.06	-52.64±0.04	**-47.16**±0.05
	5		-41.73	-35.64	-66.02	-64.84	**-61.08**	-56.81±0.06	-61.27±0.04	**-55.23**±0.03
c20ng	1		-156.09	-155.28	-161.03	**-160.67**	-161.92	**-157.85**±0.03	-159.29±0.03	-158.55±0.03
	3	**-154.53**	-156.51	-155.74	-172.34	**-168.94**	-173.11	**-164.45**±0.02	-165.77±0.02	-165.23±0.02
	5		-157.58	-155.98	-182.88	**-177.08**	-180.47	**-170.94**±0.06	-172.50±0.05	-171.31±0.06
connect4	1		-20.30	-15.61	-24.24	-26.27	**-22.89**	-23.25±0.01	-24.92±0.00	**-21.33**±0.00
	3	**-15.06**	-23.01	-18.76	-42.60	**-35.33**	-36.62	-38.67±0.02	-32.79±0.01	**-31.46**±0.01
	5		-24.02	-20.16	-60.91	**-42.02**	-47.24	-52.85±0.03	**-38.24**±0.01	-38.99±0.02
cr52	1		-94.40	**-87.12**	-95.68	-100.45	**-95.54**	-91.96±0.05	-98.85±0.04	**-91.94**±0.05
	3	**-87.12**	-94.93	-92.49	-112.41	**-110.93**	-118.58	**-101.53**±0.08	-107.43±0.06	-104.92±0.05
	5		-95.66	-100.32	-128.54	**-121.04**	-138.53	**-110.94**±0.10	-115.67±0.05	-118.88±0.08
cwebkb	1		-155.68	-152.92	**-160.32**	-160.63	-160.47	**-155.63**±0.05	-158.97±0.03	-156.27±0.05
	3	**-152.10**	-156.35	-154.56	-175.94	**-169.32**	-174.95	**-162.59**±0.10	-165.64±0.06	-164.14±0.08
	5		-157.73	-158.00	-190.51	**-177.86**	-192.51	**-169.45**±0.10	-172.52±0.10	-173.41±0.10
dna	1		-82.66	-80.09	-87.68	-86.85	**-86.78**	-83.48±0.08	-84.65±0.04	**-83.04**±0.06
	3	**-79.97**	-84.31	-80.90	-102.88	**-93.76**	-96.24	-90.38±0.15	-88.81±0.07	**-87.13**±0.09
	5		-85.90	-84.10	-117.82	**-99.05**	-103.31	-97.13±0.19	-92.09±0.05	**-91.11**±0.07
jester	1		-54.18	-53.25	-56.23	**-56.09**	**-56.09**	**-54.04**±0.02	-54.93±0.01	-54.11±0.02
	3	**-53.15**	-55.08	-53.57	-61.34	**-59.32**	-60.61	**-55.70**±0.03	-56.91±0.03	-55.82±0.03
	5		-55.67	-53.87	-65.67	**-61.83**	-64.46	**-57.26**±0.04	-58.45±0.03	-57.36±0.03
kdd	1		-9.09	-2.72	-10.73	-13.68	**-7.04**	-7.99±0.01	-12.61±0.01	**-6.61**±0.00
	3	**-2.07**	-9.29	-4.41	-26.87	-20.82	**-15.89**	-17.63±0.01	-18.29±0.01	**-13.04**±0.01
	5		-10.20	-6.59	-37.82	-26.88	**-20.16**	-24.57±0.02	-22.96±0.01	**-17.86**±0.01
kosarek	1		-12.42	-11.31	-18.80	-19.08	**-17.61**	-16.43±0.02	-17.69±0.01	**-15.83**±0.01
	3	**-10.59**	-15.13	-12.40	-34.05	-28.18	**-27.33**	-27.30±0.04	-25.75±0.02	**-23.78**±0.02
	5		-16.80	-15.14	-48.31	**-35.57**	-49.45	-36.73±0.05	**-32.23**±0.03	-33.27±0.05
moviereview	1		-340.12	-337.16	-343.47	-344.04	**-343.46**	**-336.97**±0.08	-341.96±0.09	-339.02±0.09
	3	**-335.02**	-338.27	-336.20	-359.88	**-348.00**	-354.65	**-340.92**±0.20	-343.67±0.17	-341.83±0.17
	5		-339.04	-337.24	-375.89	**-354.05**	-371.65	**-345.05**±0.30	-347.90±0.24	-346.76±0.27
msnbc	1		-9.31	-6.88	-12.01	-12.37	**-9.81**	-8.32±0.01	-10.11±0.01	**-8.06**±0.00
	3	**-6.43**	-8.54	-8.26	-19.88	-11.99	**-11.86**	-11.28±0.01	-10.25±0.01	**-9.81**±0.00
	5		-9.98	-10.20	-25.39	-12.23	**-12.07**	-13.51±0.01	**-10.98**±0.00	-11.03±0.00
msweb	1		-17.49	-11.92	-19.17	-23.83	**-18.97**	**-16.75**±0.02	-22.55±0.02	-17.15±0.01
	3	**-10.02**	-18.42	-12.96	-37.31	-34.69	**-31.01**	-29.78±0.04	-31.67±0.04	**-26.13**±0.03
	5		-19.53	-14.56	-55.25	-44.65	**-41.08**	-42.27±0.05	-39.83±0.04	**-34.36**±0.03
mushrooms	1		-26.08	-20.67	**-25.90**	-31.22	-27.73	**-23.70**±0.03	-29.77±0.03	-25.15±0.03
	3	**-17.27**	-27.45	-21.20	-42.93	-39.89	**-38.72**	-35.94±0.07	-36.34±0.03	**-30.95**±0.03
	5		-27.16	-23.67	-59.79	**-44.73**	-46.06	-47.50±0.09	-39.92±0.04	**-36.37**±0.05
nips	1		-280.53	**-277.83**	-285.55	**-284.22**	-284.51	-278.89±0.06	-281.37±0.04	**-278.76**±0.05
	3	-277.94	-280.10	**-277.61**	-300.27	**-288.44**	-294.91	-280.77±0.10	-282.39±0.08	**-280.28**±0.10
	5		-280.17	**-277.94**	-313.37	**-292.48**	-313.30	-282.64±0.12	-283.77±0.08	**-282.63**±0.12
nltcs	1		-9.35	-6.65	-11.26	-11.08	**-9.69**	-8.40±0.02	-10.12±0.02	**-8.03**±0.02
	3	**-6.09**	-10.14	-8.13	-18.43	-12.90	**-11.55**	-11.45±0.03	-11.26±0.01	**-9.79**±0.01
	5		-10.88	-10.73	-23.02	-12.63	**-11.81**	-13.24±0.05	-11.45±0.02	**-11.00**±0.01
ocr_letters	1		-50.37	-44.53	-51.74	-54.17	**-51.59**	-47.28±0.02	-52.52±0.02	**-47.21**±0.02
	3	**-44.13**	-50.56	-44.75	-65.48	**-60.08**	-61.74	-53.24±0.06	-56.16±0.02	**-52.14**±0.05
	5		-51.21	-46.50	-77.64	**-65.26**	-72.26	-58.66±0.06	-59.41±0.04	**-56.66**±0.04
plants	1		-21.16	-14.96	-23.39	-24.92	**-22.81**	-18.80±0.04	-23.80±0.03	**-18.71**±0.03
	3	**-14.46**	-22.33	-15.95	-38.81	-31.55	**-31.38**	-26.25±0.06	-28.73±0.03	**-23.77**±0.04
	5		-23.49	-16.74	-52.40	-36.90	**-34.73**	-32.39±0.10	-32.48±0.06	**-27.60**±0.05
pumsb_star	1		-31.15	-27.24	**-36.10**	-36.85	-36.80	-33.43±0.06	-35.10±0.04	**-33.11**±0.06
	3	**-27.17**	-32.42	-37.27	-53.71	**-44.97**	-61.35	-45.53±0.13	**-41.58**±0.04	-47.32±0.09
	5		-34.32	-34.88	-71.06	**-52.27**	-75.65	-57.08±0.13	**-47.48**±0.05	-53.17±0.07
rcv1	1		-54.42	-52.27	-56.78	-57.52	**-56.52**	**-53.92**±0.01	-56.12±0.00	-54.09±0.00
	3	**-51.96**	-54.24	-52.55	-64.88	**-61.28**	-62.78	-57.64±0.01	-58.82±0.01	**-57.54**±0.01
	5		-55.39	-52.93	-71.70	**-65.90**	-67.99	-61.13±0.01	-62.28±0.00	**-60.73**±0.01
tmovie	1		-54.55	-53.56	-61.93	**-60.69**	-63.71	**-58.53**±0.07	-59.16±0.07	-58.97±0.08
	3	**-52.81**	-56.94	-56.43	-79.51	**-72.46**	-81.09	-69.74±0.22	**-68.66**±0.11	-69.78±0.16
	5		-59.30	-58.35	-96.75	**-82.62**	-90.06	-80.77±0.24	-77.06±0.09	**-76.93**±0.13
tretail	1		-13.67	-11.86	-18.80	-18.81	**-18.55**	**-15.39**±0.02	-17.49±0.02	-15.63±0.02
	3	**-10.82**	-13.10	-12.98	-31.40	**-24.62**	-28.57	-23.82±0.04	**-22.48**±0.04	-22.88±0.05
	5		-14.61	-13.98	-41.64	**-31.18**	-32.35	-31.26±0.07	-28.14±0.04	**-27.89**±0.05
voting	1		-250.36	**-241.97**	-251.19	-257.87	**-250.62**	-246.57±0.20	-254.49±0.15	**-246.41**±0.20
	3	-242.05	-260.90	-243.80	-269.26	-280.01	**-267.40**	**-255.36**±0.32	-272.14±0.24	-256.51±0.30
	5		-269.43	-244.65	-286.97	-298.15	**-280.72**	**-264.04**±0.49	-287.12±0.31	-264.38±0.39

Table 4: Full comparison between PETER, RL-TPM, and MLE-PC on the DEBD datasets.