Skepthical review: *Cosmological Parameter Inference from Merger Trees Using Hierarchical Quantum Tensor Networks*

Summary

This paper proposes inferring cosmological parameters (Ω_m and σ_8) directly from dark-matter halo merger trees using a Tree Tensor Network (TTN), termed a "Hierarchical Quantum Tensor Network (HQTT)". Each halo node is represented by four scalar features (log mass, log concentration, log $V_{\max}$, and scale factor), normalized and passed through a shared embedding MLP to produce latent node vectors (Sec. 2.1.1, Sec. 2.2.1). Tree structure is incorporated by selecting a learnable basis tensor conditioned on the node's branching factor (T_{leaf}, $T_{1\text{child}}, \ldots, T_{\max_c\text{hildren}}$), and contracting these tensors bottom-up to obtain a fixed-size root representation, which is mapped linearly to (Ω_m, σ_8) (Sec. 2.2.2–2.2.3). On a dataset of **1000** simulated trees (**70/15/15** split), the method reports test $R^2 \approx \mathbf{0.915}$ for Ω_m and **0.892** for σ_8 with MSEs $\sim \mathbf{10^{-3}}$ (Sec. 3.1), and provides qualitative interpretability analyses via embedding-weight inspection, basis-tensor norms, and perturbation/saliency examples (Sec. 3.2). The direction is timely and the inductive bias (hierarchical contraction) is a plausible match to merger-tree structure; however, the current manuscript does not yet support its strongest scientific and comparative claims due to missing dataset provenance/coverage details (including leakage-safe splitting by cosmology), lack of baselines and ablations, insufficient methodological specification for reproducibility (including TTN construction details and JAX/quimb differentiability pathway), and limited robustness/uncertainty analysis for a small dataset. Addressing these items would substantially strengthen both scientific interpretability and the "bigger-picture" impact of the work.

Strengths

- Well-motivated problem: using the full hierarchical information in merger trees for cosmological inference is timely and potentially impactful compared to relying solely on hand-crafted summaries (Sec. 1).
- Clear high-level pipeline: node features → shared embedding → arity-dependent basis tensors → bottom-up TTN contraction → fixed-size root vector → linear head to two parameters (Sec. 2.2.1–2.2.4).
- Conceptually suitable inductive bias: TTNs naturally match rooted hierarchical structures and can, in principle, separate "main-branch propagation" from multi-progenitor merger events via different arity tensors (Sec. 2.2.2–2.2.3, Sec. 3.2.2).
- Reported predictive performance is promising ($R^2 \sim \mathbf{0.9}$ on the provided split), suggesting nontrivial signal in the trees and that the architecture can exploit it (Sec. 3.1).

- Interpretability is treated as a first-class goal, with multiple qualitative probes (feature embedding inspection, basis-tensor norms, perturbation/saliency) that could become compelling if made more systematic and quantitatively validated (Sec. 3.2).

Major issues

1. **Dataset provenance, cosmological coverage, and target definition are insufficiently specified, preventing scientific interpretation of the reported R^2 and MSE (Sec. 2.1.1, Sec. 3.1, Sec. 4.1).** The manuscript does not clearly state the simulation suite(s) (e.g., AbacusSummit or otherwise), box size/resolution, halo finder and tree builder, whether baryonic physics is present, the selection of root halos (mass/redshift), node pruning/cuts, snapshot/redshift sampling, and—critically—the ranges/priors and sampling strategy for Ω_m and σ_8 and how many distinct cosmologies are represented. Without these, the difficulty and generality of the task (broad regression vs. narrow interpolation) cannot be assessed, and the performance metrics are hard to contextualize.

   Recommendation: Add a dedicated dataset subsection in Sec. 2.1 that includes: (i) simulation suite name(s), box size, mass resolution, number of boxes, and physics (DMO vs hydro); (ii) halo finder + merger-tree algorithm and any key settings; (iii) explicit ranges/priors for Ω_m and σ_8, number of unique cosmologies, and trees per cosmology; (iv) how root halos are selected (e.g., $z = 0$ roots, root-mass range), node-level cuts (minimum progenitor mass, treatment of disrupted halos), and snapshot times/redshifts used (linking "scale factor" to node time); and (v) summary statistics/histograms of target distributions and tree sizes across train/val/test.

2. **Potential information leakage due to splitting "by tree" rather than "by cosmology/simulation" is not addressed, and could substantially inflate test performance (Sec. 2.3.2, Sec. 3.1).** If multiple trees share the same underlying cosmology (common in simulation suites), then random tree-level splits allow the model to implicitly learn cosmology-specific artifacts and generalize only within-cosmology rather than to unseen cosmologies.

   Recommendation: Define and implement leakage-safe splits. At minimum, report results for (i) a split by cosmology (all trees from a given cosmology assigned to a single split) and/or (ii) split by simulation box/realization if multiple realizations per cosmology exist. In Sec. 2.3.2 describe the split unit explicitly (tree vs cosmology vs box), and in Sec. 3.1 report performance for both the original and leakage-safe splits (with identical metrics), discussing any gap.

3. **Claims of improvement over "traditional summary-statistic methods" and the necessity of the TTN inductive bias are not supported because no baselines are evaluated (Abstract, Sec. 1, Sec. 3.1, Sec. 4.2–4.4).** Without comparisons, it is unclear whether the TTN is outperforming simpler alternatives (e.g., root-only features, summary-statistics regressors, DeepSets, or standard GNNs).

Recommendation: Add baseline experiments trained/evaluated on identical splits (especially the leakage-safe split) and report results in a table in Sec. 3.1: (i) root-only regressor (MLP on root node features); (ii) summary-statistics regressor (linear/RandomForest/MLP) using interpretable tree summaries (main-branch mass assembly, formation time, major/minor merger counts, progenitor-mass moments vs scale factor, node count); (iii) topology-aware baseline such as a message-passing GNN (GraphSAGE/GIN) using the same node features and edges; and optionally (iv) a DeepSets "bag-of-nodes" model to test whether topology matters. Update Abstract/Sec. 4 claims accordingly (quantify gains or soften claims if comparable).

4. **Method and implementation details are insufficient for reproducibility and for assessing capacity/overfitting risk (Sec. 2.2–2.3).** Key missing items include: exact embedding MLP and output head architecture (layer widths, activations, normalizations), final hyperparameters (d_{embed}, d_{bond}, max_children), optimizer settings, LR schedule, batch size, number of epochs, early stopping criteria, regularization, initialization of basis tensors, parameter count, and how batching works with variable tree shapes.

   Recommendation: Expand Sec. 2.2–2.3 with a reproducibility checklist: (i) explicit layer-by-layer definitions for NN_embed and the output head; (ii) final hyperparameter values used for Sec. 3.1 results (and what was tuned); (iii) optimizer (e.g., Adam/optax) with full hyperparameters, LR schedule, batch size, epochs, early stopping; (iv) initialization for each $\boldsymbol{T}_k$ and network weights; (v) total trainable parameter count (broken down into embedding, tensors, head) and training/validation curves; and (vi) a public code link or at least pseudocode for the full training loop and TTN construction.

5. **Tree definition and TTN construction contain ambiguities (rooting, edge direction, "children" meaning, ordering/permutation invariance, max_children handling) that are central to correctness (Sec. 2.1.1, Sec. 2.1.4, Sec. 2.2.2–2.2.3).** Merger trees are time-directed DAGs; depending on convention, "parent/child" can swap between progenitor/descendant. Also, if child ordering is deterministic (e.g., sorted by mass or time), it may leak additional information; if arbitrary, the model may not be permutation invariant.

   Recommendation: In Sec. 2.1.4 and Sec. 2.2.2–2.2.3: (i) define edge_index direction explicitly (progenitor→descendant or reverse) and map it to TTN parent/child roles; (ii) define the unique root (e.g., $z = 0$ descendant) and confirm all graphs are connected acyclic trees after preprocessing; (iii) specify how children are ordered and whether the model is intended to be permutation invariant—if invariance is desired, enforce it (e.g., symmetric tensor constraints, commutative pooling, or randomized child order during training); (iv) report how nodes with arity $> \mathbf{max_children}$ are handled (cap/merge/prune) and how arities $< \mathbf{max_children}$ select tensors (one tensor per exact arity vs masking/shared parameters); and (v) add a schematic figure showing a small merger tree mapped to tensors and contraction order, with index labels.

6. **Evaluation lacks robustness, uncertainty quantification, and cosmology-specific diagnostics; point-estimate MSE alone is not enough for cosmological inference, especially given known Ω_m–σ_8 degeneracies (Sec. 3.1, Sec. 4.2).** Only a single split/seed result is shown; residual structure (bias vs parameter value, heteroskedasticity vs tree size/root mass) and prediction covariance are not characterized.

   Recommendation: Augment Sec. 3.1 with: (i) multiple random seeds and (if feasible) multiple splits, reporting mean±std for MSE, MAE, and R^2; (ii) bootstrap confidence intervals on metrics; (iii) residual plots binned by true Ω_m and σ_8 (quantify regression-to-the-mean) and by tree size/root mass; (iv) report the 2D error covariance (or correlation) of prediction errors to assess degeneracy directions; and (v) consider a simple uncertainty-aware head (e.g., heteroscedastic Gaussian regression) or ensembling to provide predictive uncertainties and basic calibration checks.

7. **Interpretability claims are currently qualitative and in places methodologically weak: embedding weight magnitudes are not reliable feature-importance measures for MLPs, and tensor norms can scale with tensor order/initialization rather than learned physical meaning (Sec. 2.4.2, Sec. 3.2.1–3.2.3, Sec. 4.3).** The analysis also does not clearly state sample sizes or selection criteria for case studies.

   Recommendation: Strengthen Sec. 3.2 with quantitative, dataset-level attribution: (i) permutation importance and/or integrated gradients across the full model for the four node features; (ii) leave-one-feature-out or retrain ablations (drop scale factor, drop mass, etc.) reporting performance deltas; (iii) topology vs features tests (randomize child order; shuffle topology while keeping node features; swap subtrees between trees) to isolate what information the TTN uses; (iv) controlled merger masking by mass ratio to quantify the impact of major vs minor mergers on predictions (report distributions of $|\Delta\Omega_m|$, $|\Delta\sigma_8|$); and (v) clearly state how many trees are analyzed in each interpretability plot and how they are selected (random vs high-error vs representative).

Minor issues

1. Computational cost, scalability, and the practical JAX/quimb differentiability pathway are not sufficiently documented (Sec. 2.3.3, Sec. 4.4). Variable tree shapes can cause JIT recompilation overhead; quimb is often NumPy-based unless a JAX backend/autoray route is used, so it is unclear how gradients/JIT are handled and what runtime is.

   Recommendation: In Sec. 2.3.3 (or Sec. 3.1), report hardware (CPU/GPU), wall-clock training time, memory use, average nodes per tree, and how runtime scales with tree size and d_{bond}. Explicitly state the quimb backend and how contractions are per-

formed to remain JAX-differentiable and JIT-friendly (e.g., autoray/JAX arrays, jax.numpy einsum). Note whether you pad/bucket trees to reduce recompilations.

2. Branching-factor statistics and the chosen value of max_children are not reported, limiting understanding of model complexity and how often higher-arity tensors are used (Sec. 2.1.4, Sec. 2.2.2, Sec. 3.2.2).

   Recommendation: Add in Sec. 2.1.4 a histogram/table of node arities $(\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots)$ and maximum observed arity; state the resulting max_children used in experiments. In Sec. 3.2.2, contextualize basis-tensor analyses by reporting usage frequency of each $\boldsymbol{T}_k$ during contractions.

3. Evaluation metrics are not normalized to the target scale; MSE values are difficult to interpret without target ranges/variance (Sec. 3.1).

   Recommendation: In Sec. 3.1, report the mean/std and min/max of $\boldsymbol{\Omega}_m$ and $\boldsymbol{\sigma}_8$ (per split), and add MAE and normalized RMSE (e.g., RMSE divided by target std). Include residual histograms and binned errors to show where in parameter space the model succeeds/fails.

4. Hyperparameter specification is inconsistent between "typical" values in Methods and the fixed values used in Results, obscuring what configuration produced the main numbers (Sec. 2.2.1–2.2.2, Sec. 2.3.4, Sec. 3.1).

   Recommendation: Add a compact hyperparameter table for the main model ($\boldsymbol{d}_{\text{embed}}$, $\boldsymbol{d}_{\text{bond}}$, max_children, MLP layers/activations, optimizer/LR, batch size, epochs, early stopping). Clearly separate "searched ranges" from "final chosen values."

5. Terminology "Hierarchical Quantum Tensor Network (HQTT)" is potentially misleading because the method is classical; this affects audience expectations and positioning (Sec. 1, Sec. 4.4).

   Recommendation: Clarify early (Sec. 1–2) that this is a classical TTN inspired by tensor-network methods from quantum many-body physics, with no quantum hardware/algorithmic speedup claimed. Consider renaming to "Hierarchical Tensor Network" or "Tree Tensor Network" (or explicitly justify "quantum" as historical).

6. Related work positioning is incomplete with respect to graph/tree models used in cosmological inference; as written, readers cannot easily place TTNs relative to GNNs, tree-LSTMs, DeepSets, or simulation-emulator approaches (Sec. 1).

   Recommendation: Expand Sec. 1 to explicitly compare against common alternatives (summary-statistics emulators, GNNs on halo catalogs/graphs, tree sequence models along main branches). State what TTNs add (e.g., explicit low-rank factorization, hierarchical contraction, potentially different inductive bias/interpretability).

7. The paper lacks an explicit "Limitations and Future Work" section synthesizing constraints on dataset scope, generalization (e.g., to hydro/baryons or observational noise), and scaling to more parameters (Sec. 4).

   Recommendation: Add a short subsection in Sec. 4 covering: dataset scope/size, leakage/generalization risks, sensitivity to tree-building conventions, prospects for including baryonic effects/noise/selection functions, and extension to higher-dimensional cosmological parameter spaces.

8. Loss equation notation is internally inconsistent and unclear (Sec. 2.3.2).

   Recommendation: Rewrite Eq. (1) cleanly, e.g. $\mathcal{L} = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \left[(\Omega_{m,i}^{\text{pred}} - \Omega_{m,i}^{\text{true}})^2 + (\sigma_{8,i}^{\text{pred}} - \sigma_{8,i}^{\text{true}})^2 \right]$, and explicitly define indices and any additional averaging (over parameters).

Very minor issues

1. Index-order and tensor-shape conventions for basis tensors are described qualitatively but not fixed (e.g., which axis is parent vs child-1/child-2), and naming is inconsistent (T_leaf vs Tleaf; T_1child vs T1child) (Sec. 2.2.2–2.2.3).

   Recommendation: Standardize tensor names and provide a small table listing each $\boldsymbol{T}_k$ with its exact index order and shape (e.g., $\boldsymbol{T}_k[\mathbf{embed}, c_1, \ldots, c_k, p]$). Add one explicit Einstein-summation line showing contraction at a node and how the root open index is selected (Sec. 2.2.3).

2. Presentation/typos: inconsistent quotation marks for code identifiers, stray bullets/hyphens, and unusual title/affiliation text reduce professionalism (Sec. 2.2.2, Sec. 2.3.2, Sec. 4.4).

   Recommendation: Proofread and standardize formatting (use backticks for code, consistent heading capitalization). Remove stray bullets and replace nonstandard affiliation/title lines with venue-appropriate formatting.

3. Keywords in Abstract/Conclusion are generic and omit key technical terms (Abstract, Sec. 4.4).

   Recommendation: Revise keywords to emphasize "merger trees", "tree tensor networks", "tensor networks", and "cosmological parameter inference"; keep generic terms only if required by the venue.

4. References/citations show formatting inconsistencies and some seemingly tangential citations (Sec. 5/References, Sec. 1).

   Recommendation: Audit citation relevance and standardize reference formatting (author names, venues/arXiv IDs). Ensure each in-text citation supports the specific statement it is attached to.

5. Some long multi-clause sentences reduce readability (Sec. 1, Sec. 4.4).

   Recommendation: Split the longest sentences in Sec. 1 and Sec. 4.4 into shorter statements, especially where mixing motivation, method summary, and claims in one sentence.

Key statements and references

- ✔ **Dark matter halos assemble hierarchically over cosmic time through continuous accretion and mergers, with this evolutionary history encoded in merger trees that have been modeled and characterized in detail using semi-analytic and simulation-based approaches to halo assembly and merger tree generation (e.g., Benson et al. 2012; Jiang and van den Bosch 2013; Parkinson et al. 2007; Yung et al. 2024).**
- *Reference(s):* Benson et al., 2012, Jiang and van den Bosch, 2013, Parkinson et al., 2007
- *Justification:* Directly supported. All three papers state that halos build up hierarchically via mergers and accretion and that this history is encoded in merger trees, which are modeled with semi-analytic/EPS methods and compared to or calibrated against N-body simulations. Parkinson et al. (2007) introduce a Monte Carlo merger-tree algorithm tuned to the Millennium simulation; Jiang and van den Bosch (2013) explicitly describe merger trees as encoding hierarchical mass assembly, compare multiple EPS-based algorithms to simulation benchmarks, and include smooth accretion; Benson et al. (2012) construct merger trees (using the Parkinson et al. algorithm), model merger rates and progenitor functions, and explicitly include smooth (non-halo) accretion, with detailed comparisons to simulations.

- ✔ **Tensor networks, and in particular hierarchical variants such as Tree Tensor Networks (TTNs), have been developed as efficient representations of high-dimensional correlated systems and are especially well-suited to data with intrinsic tree-like structure, as demonstrated in prior work on TTN algorithms and applications in physics and chemistry (Milsted et al. 2019; Gunst et al. 2019; Dowling et al. 2024).**

- *Reference(s):* Milsted et al., 2019, Gunst et al., 2019, Dowling et al., 2024
- *Justification:* Milsted et al., 2019 describe and implement a hierarchical Tree Tensor Network (TTN) algorithm for quantum spin systems, noting tensor networks were developed to efficiently simulate complex quantum systems and that the TTN organizes tensors in a tree structure with favorable computational scaling (e.g., $O(\chi^4)$) (Milsted et al., 2019). Gunst et al., 2019 extend a tree-based ansatz (T3NS) for quantum chemistry, arguing the tree-shaped network better captures molecular entanglement than linear MPS and providing resource scalings and accurate calculations on Cu_2O_2 isomers, demonstrating chemistry applications (Gunst et al., 2019). Dowling et al., 2024 introduce 'process trees'—tree-geometry tensor networks for multi-time quantum pro-

cesses—explicitly motivated by the fact that TTNs/MERA naturally accommodate polynomially decaying correlations, and show efficient computation and successful application to the spin-boson model (Dowling et al., 2024). Collectively, these works support that hierarchical TTN-like tensor networks are efficient representations for high-dimensional correlated systems and are particularly suitable when the problem exhibits a hierarchical/tree-like structure.

- ✔ **The merger-tree dataset used in this work builds on recent simulation and modeling efforts in which dark matter halo merger trees are generated and represented as graphs with node-level halo properties and parent–child relations, enabling machine-learning applications to emulate or analyze merger histories (Hearin et al. 2022; Lucie-Smith et al. 2024; Nguyen et al. 2025).**

- *Reference(s):* Hearin et al., 2022, Lucie-Smith et al., 2024, Nguyen et al., 2025
- *Justification:* Nguyen et al. (2025) explicitly represent full halo merger trees as graph structures with node-level properties (e.g., mass and concentration) and parent–child relations, and use a graph generative ML model to emulate them (abstract; Secs. 3–4; Fig. 2). Hearin et al. (2022) base their differentiable assembly model on large samples of merger trees extracted from simulations with Rockstar/ConsistentTrees and SUBLINK, i.e., simulation-generated parent–progenitor trees (Sec. 2; §3). Lucie-Smith et al. (2024) construct merger trees with TANGOS and use machine learning (IVE) to analyze how profiles relate to mass accretion/formation histories derived from those trees (Methods/Supp. B; Figs. 2–3). Collectively, these works support that recent simulation/modeling efforts generate merger trees and represent them as node–edge graphs enabling ML to emulate or analyze merger histories.

- ⚠ **The training setup—standardizing node features using statistics computed on the training set and optimizing a Mean Squared Error loss with Adam in JAX/optax—follows established best practices from recent studies showing that feature scaling and adaptive gradient-based optimization significantly affect regression performance and cosmological machine-learning pipelines (de Amorim et al. 2022; Pinheiro et al. 2025; Gangloff and Jouvin 2024).**

- *Reference(s):* de Amorim et al., 2022, Pinheiro et al., 2025, Gangloff and Jouvin, 2024
- *Justification:* Pinheiro et al., 2025 explicitly recommend fitting scalers on the training set and applying them to train/test to avoid leakage (Section IV.B; Algorithm 1 step 7) and demonstrate that feature scaling significantly affects performance on regression tasks (e.g., Wilcoxon/Friedman tests in Tables 6–8). However, the paper does not discuss JAX/optax, does not analyze the impact of Adam or adaptive optimizers, and does not cover cosmological pipelines. Thus only the feature-scaling/best-practice part is supported.

Mathematical consistency audit

This section audits **symbolic/analytic** mathematical consistency (algebra, derivations, dimensional/unit checks, definition consistency).

Maths relevance: light

The paper is primarily methodological and descriptive, with limited formal mathematics. The main symbolic content consists of tensor shape specifications for a tree tensor network over merger trees and a mean-squared-error loss definition for predicting two cosmological parameters. There are no detailed derivations, proofs, or multi-step algebraic manipulations to audit; the main checks are shape/definition consistency and correctness/clarity of the one explicit formula.

Checked items

1. ✔ **Node feature standardization definition** (Sec. 2.1.2, p.3)
    - **Claim:** Each of the 4 node features is standardized using training-set mean and standard deviation, then applied consistently to all splits.
    - **Checks:** definition consistency, dimensional/symbol sanity
    - **Verdict:** PASS; confidence: high; impact: minor
    - **Assumptions/inputs:** Feature vector x has shape [num_nodes, 4]., Means/standard deviations are computed on training set only.
    - **Notes:** No symbolic inconsistencies found; the described standardization is well-defined and consistent with later use of normalized features as input to NN_embed.
2. ✔ **Embedding network mapping** (Sec. 2.2.1, p.3)
    - **Claim:** NN_embed maps a 4D normalized node feature vector to a d_{embed}-dimensional embedding vector v_j.
    - **Checks:** shape consistency, notation consistency
    - **Verdict:** PASS; confidence: high; impact: minor
    - **Assumptions/inputs:** Input feature dimension is exactly 4., Output embedding dimension is d_{embed}.
    - **Notes:** The mapping $4 \to d_{\text{embed}}$ is consistent with later basis tensor shapes that take a d_{embed} index.
3. ✔ **Leaf basis tensor shape and contraction** (Sec. 2.2.2–2.2.3, pp.3–4)
    - **Claim:** For a leaf node, T_{leaf} has shape $(d_{\text{embed}}, d_{\text{bond}})$ and contracting it with v_j along d_{embed} yields an effective leaf tensor carrying a single d_{bond} index to connect upward.
    - **Checks:** shape/algebra check
    - **Verdict:** PASS; confidence: high; impact: moderate

- **Assumptions/inputs:** v_j has shape $(d_{\text{embed}},)$. Contraction is along the d_{embed} index.
- **Notes:** Contracting $(d_{\text{embed}}, d_{\text{bond}})$ with $(d_{\text{embed}},)$ over d_{embed} yields $(d_{\text{bond}},)$, matching the intended single bond to the parent.

4. ✔ **One-child basis tensor shape and resulting bonds** (Sec. 2.2.2–2.2.3, pp.3–4)
   - **Claim:** For a node with one child, $T_{1\text{child}}$ has shape $(d_{\text{embed}}, d_{\text{bond}}, d_{\text{bond}})$, and after contracting with v_j the node tensor has one bond to the child and one to the parent.
   - **Checks:** shape/algebra check, interface consistency between parent/child tensors
   - **Verdict:** PASS; confidence: medium; impact: moderate
   - **Assumptions/inputs:** Child effective tensor exports a d_{bond} index upward (to be contracted). All child/parent bond dimensions are d_{bond}.
   - **Notes:** After contracting with v_j, the tensor becomes a $(d_{\text{bond}}, d_{\text{bond}})$ object. Contracting one axis with the child's $(d_{\text{bond}},)$ vector leaves a $(d_{\text{bond}},)$ vector passed upward. Axis order is not fixed in text, but a consistent choice exists.

5. ✔ **Two-child (and k-child) basis tensor shape generalization** (Sec. 2.2.2, p.4)
   - **Claim:** For a node with two children, $T_{2\text{child}}$ has shape $(d_{\text{embed}}, d_{\text{bond}}, d_{\text{bond}}, d_{\text{bond}})$, with indices for two children and one parent; this pattern extends up to max_children.
   - **Checks:** shape/algebra check, definition consistency
   - **Verdict:** PASS; confidence: medium; impact: moderate
   - **Assumptions/inputs:** Each child contributes one d_{bond} index to be contracted at its parent. Parent output remains a single d_{bond} index after contracting all children.
   - **Notes:** Contracting v_j reduces rank by one, giving a rank-3 tensor with three d_{bond} indices. Contracting with two child vectors leaves one d_{bond} index to pass upward. The text does not specify axis ordering, but the stated dimensionality works.

6. ⚠ **Root contraction output dimension** (Sec. 2.2.3, p.4)
   - **Claim:** Global contraction of the TTN yields, at the root, a fixed-dimension vector of size d_{bond} summarizing the merger tree.
   - **Checks:** shape consistency, missing-definition check
   - **Verdict:** UNCERTAIN; confidence: low; impact: moderate
   - **Assumptions/inputs:** Every non-root node passes exactly one d_{bond} index to its parent (its 'parent bond'). The root leaves one index uncontracted and treats it as the representation vector.

- **Notes:** This is plausible given the basis tensor shapes (each node appears to have exactly one 'parent' bond). However, the paper does not explicitly define how the root is handled when it has no parent (e.g., whether the 'parent bond' is repurposed as the output index, or whether a special root tensor without a parent index is used). An explicit index-level formula for the contraction would resolve this.

7. ✔ **Prediction head dimensionality** (Sec. 2.2.4, p.4)
   - **Claim:** A linear OutputLayer maps the d_{bond}-dimensional root vector to a 2D output $(\Omega_{m,\text{pred}}, \sigma_{8,\text{pred}})$.
   - **Checks:** shape consistency
   - **Verdict:** PASS; confidence: high; impact: minor
   - **Assumptions/inputs:** Root representation has shape $(d_{\text{bond}},)$.
   - **Notes:** A linear map $\mathbb{R}^{d_{\text{bond}}} \to \mathbb{R}^2$ is well-defined and consistent with the stated two-target regression.
8. ✖ **MSE loss formula (summation and normalization)** (Sec. 2.3.2, p.4)
   - **Claim:** Loss is the batch mean of the sum of squared errors for Ω_m and σ_8.
   - **Checks:** algebra/notation consistency, normalization check
   - **Verdict:** FAIL; confidence: high; impact: moderate
   - **Assumptions/inputs:** Batch size is N_{batch}. Each sample i has a 2D target and 2D prediction.
   - **Notes:** The written expression includes an extraneous/undefined 'N' and a malformed summation '∑batch' (as transcribed: 'Loss = 1/Nbatch N ∑batch i=1 [...]'). The intended loss is clear from context, but as written it is not internally consistent. The first incorrect element is the extra 'N' placed between $1/N_{\text{batch}}$ and the summation.

Limitations

- The paper contains very few explicit equations and no multi-step derivations; most of the method is described in words, so many potential consistency checks (e.g., explicit index contractions, exact tensor definitions) are not possible from the provided text alone.
- The TTN construction relies on implicit index-labeling and contraction conventions (via quimb/einsum) that are not mathematically specified in the manuscript; without explicit index notation, only plausibility checks on tensor shapes can be performed.
- No unit/dimensional analysis for physical quantities (mass, concentration, V_{max}, scale factor) is possible because the model uses standardized/log-transformed features and the paper does not define physical units beyond brief descriptions.

Numerical results audit

This section audits **numerical/empirical** consistency: reported metrics, experimental design, baseline comparisons, statistical evidence, leakage risks, and reproducibility.

All executed internal arithmetic and consistency checks passed: the 1000-tree dataset split is self-consistent in both counts and percentages; repeated dataset/hyperparameter constants match across sections; R^2 values are correctly translated to percentage-of-variance language (including a heuristic 'nearly 90%' statement); the 2D output matches the two stated target parameters; and the loss function's averaging/indexing and two-term structure are symbolically consistent.

Checked items

1. ✔ **C1_dataset_split_counts** (Page 3, Sec. 2.1.3 "Dataset Splitting")
    - **Claim:** "The full dataset of 1000 merger trees is partitioned … 70% (700 trees) … 15% (150 trees) … remaining 15% (150 trees)"
    - **Checks:** parts_vs_total_and_percentages
    - **Verdict:** PASS
    - **Notes:** Counts sum to 1000; percentages sum to 1.0; each count equals percent×total exactly.
2. ✔ **C2_test_set_size_repeated** (Page 6, Sec. 3.1 "Quantitative Performance Evaluation")
    - **Claim:** "held-out test set comprising 150 merger trees"
    - **Checks:** repeated_constant_match
    - **Verdict:** PASS
    - **Notes:** Test set size (150) matches the dataset split statement.
3. ✔ **C3_conclusion_split_counts_repeated** (Page 7, Sec. 4.1 "Methods and Dataset Summary")
    - **Claim:** "trained end-to-end on 700 trees, with 150 trees each for validation and testing"
    - **Checks:** repeated_constant_match_and_parts_vs_total
    - **Verdict:** PASS
    - **Notes:** $700 + 150 + 150$ equals the stated total of 1000.
4. ✔ **C4_r2_to_percent_omega_m** (Page 6, Sec. 3.1)
    - **Claim:** "R-squared (R2) value of 0.915 … indicates that approximately 91.5% of the variance … is explained"
    - **Checks:** percentage_conversion
    - **Verdict:** PASS

- **Notes:** $100 \times 0.915 = 91.5\%$, matching the claim exactly (within stated rounding tolerance).

5. ✔ **C5_r2_to_percent_sigma8_nearly_90** (Page 6, Sec. 3.1)
    - **Claim:** "R2 value of 0.892 … capturing nearly 90% of its variance"
    - **Checks:** approximate_percentage_claim
    - **Verdict:** PASS
    - **Notes:** Heuristic check: $100 \times 0.892 = 89.2\%$, which is within ± 2 percentage points of 90% for the phrase 'nearly 90%'.
6. ✔ **C6_hyperparam_embed_dimension_consistency** (Page 6, Sec. 3.1 and Page 8, Sec. 4.2)
    - **Claim:** Optimal hyperparameters included embedding dimension d_{embed} of 16 (reported in Results and reiterated in Conclusions).
    - **Checks:** repeated_constant_match
    - **Verdict:** PASS
    - **Notes:** Embedding dimension is consistently reported as 16.
7. ✔ **C7_hyperparam_bond_dimension_consistency** (Page 6, Sec. 3.1 and Page 8, Sec. 4.2)
    - **Claim:** Optimal hyperparameters included bond dimension d_{bond} of 8 (reported in Results and reiterated in Conclusions).
    - **Checks:** repeated_constant_match
    - **Verdict:** PASS
    - **Notes:** Bond dimension is consistently reported as 8.
8. ✔ **C8_output_dimension_matches_two_targets** (Page 4, Sec. 2.2.4 "Prediction Head")
    - **Claim:** "maps the d_{bond}-dimensional input to a 2-dimensional output vector, representing … $(\Omega_{m,\text{pred}}, \sigma_{8,\text{pred}})$"
    - **Checks:** dimensional_consistency_from_explicit_targets
    - **Verdict:** PASS
    - **Notes:** The stated output dimension (2) matches the two explicit targets (Ω_m and σ_8).
9. ✔ **C9_loss_normalization_symbolic** (Page 4, Sec. 2.3.2 "Loss Function")
    - **Claim:** Loss is defined with a prefactor $1/N_{\text{batch}}$ multiplying a sum over $i = 1..N_{\text{batch}}$ of two squared-error terms.
    - **Checks:** symbolic_summation_index_consistency
    - **Verdict:** PASS
    - **Notes:** Symbolic consistency: denominator and summation limit both reference N_{batch}; per-item loss includes exactly two squared terms.

Limitations

- Only parsed text was available; no tables/figures with numeric entries were provided beyond text statements.
- All performance metrics (MSE, R^2) and interpretability claims depend on unavailable underlying data (predictions, labels, learned weights/tensors), so only internal arithmetic/consistency checks are feasible.
- No batch size, learning rate, or other numeric hyperparameters beyond d_{embed} and d_{bond} are specified, limiting the number of fast numeric checks.