CHAP: Causal Hierarchy-Aware Prediction for Tabular Data via Masked Attention

Anonymous submission

Methodology Details and Theoretical Proofs

The Detail of Optimization with Lazy Update

Algorithm 1 illustrates how the lazy update mechanism alternates training between the reconstruction and prediction branches. Let C denote the parameters related to the causal mask, P denote the predictor-related parameters (including a_i), and Θ represent all other model parameters. T_s is the number of mini-batches used for reconstruction branch updates, and T is the cycle length. Specifically, for every T mini-batches, the first T_s are used to update the reconstruction branch, while the remaining $T - T_s$ are used to update the prediction branch. A smaller T_s/T reduces training time but may lead to lower model performance. Empirically, we set $T_s = 2$ and $T = 10$.

Algorithm 1: Lazy Alternating Optimization of CHAP

Input: Training data $\mathcal{D}$, causal mask C, predictor P, other parameters Θ
Parameter: T_s, T
Output: Trained CHAP model

Initialize $t = 0$
for each mini-batch $B \in \mathcal{D}$ **do**
  if $t \bmod T < T_s$ **then**
    Randomly select a field in B, replace it with `[MSK]`
    Compute $\mathcal{L}_R = \alpha\mathcal{L}_{recon} + \beta(\mathcal{L}_{DAG} + \mathcal{L}_{sparse})$
    Update C and Θ via gradient descent
  else
    Replace target y in B with `[MSK]`
    Freeze C, compute $\mathcal{L}_P = \mathcal{L}_{pred} + \mathcal{L}_{dis}$
    Update P and Θ via gradient descent
  end if
  $t \leftarrow t + 1$
end for
return CHAP model =0

Details of Design Choices

In the *Effectiveness of Design Choices* subsection, we analyze the impact of three key design choices in CHAP: the method of applying the causal mask, the predictor aggregation strategy, and the approach to avoiding self-prediction during reconstruction. Below, we provide the specific formulations corresponding to each design.

Method of Applying Causal Mask To apply the learned causal graph $M \in \mathbb{R}^{n\times n}$ during attention computation, we explore three integration strategies. Let $A \in \mathbb{R}^{n\times n}$ denote the raw attention logits (before softmax), and $Attention$ be the normalized attention weights.

- **Multiplicative Masking (Used in CHAP)** This approach applies element-wise multiplication between the attention weights and the causal mask:

$$\tilde{Attention}_{i,j} = Attention_{i,j} \cdot M_{i,j} \tag{A-1}$$

- **Additive Masking** In this design, the causal scores are added to the normalized attention weights:

$$\tilde{Attention}_{i,j} = Attention_{i,j} + M_{i,j} \tag{A-2}$$

- **Softmax Modification** Instead of modifying post-softmax attention scores, this method incorporates the causal mask directly into the softmax computation:

$$Attention_{i,j} = \frac{\exp(A_{i,j}) \cdot M_{i,j}}{\sum_k \exp(A_{i,k}) \cdot M_{i,k}} \tag{A-3}$$

  This formulation ensures the attention remains normalized and grounded in causal structure.

Among the three, multiplicative masking empirically yields the best performance and integrates seamlessly with the transformer attention mechanism.

Predictor Aggregation Strategy Given the contextualized representations $H \in \mathbb{R}^{n\times d}$ for n fields, we examine several aggregation approaches to produce the final output $\hat{y}$:

- **Hierarchical-aware Aggregation (Used in CHAP)** This method assigns a learnable weight to each field, allowing the model to adaptively control their contribution:

$$\hat{y} = FNN\left(\sum_{i=1}^{n} w_i \cdot H_i\right) \tag{A-4}$$

  where w_i is a scalar parameter optimized during training.
- **Flattening** The full matrix H is flattened into a single vector before being passed to the prediction head:

$$\hat{y} = FNN(Flatten(H)) \tag{A-5}$$

This strategy treats all dimensions equally but ignores hierarchical structure.

- **Mean Pooling** A simpler alternative that averages representations across all fields:

$$\hat{y} = FNN\left(\frac{1}{n}\sum_{i=1}^{n} H_i\right) \tag{A-6}$$

It reduces variance but may lose important field-specific distinctions.

Hierarchical-aware aggregation achieves better generalization in both classification and regression tasks by learning the relative importance of each field.

Training Design to Avoid Self-Predicting In the reconstruction branch, we aim to prevent the model from directly using a field's own value for self-prediction. Without proper constraints, the hidden representation used for reconstructing s_i can unintentionally incorporate information from s_i itself, leading to information leakage.

The reconstruction process typically follows the form:

$$\hat{s}_i = Decoder(H_i), \quad H_i = \sum_{j=1}^{n} V_j \cdot Attention_{i,j} \tag{A-7}$$

Here, V_j is the value vector of field s_j, and $Attention_{i,j}$ is the attention weight from position i to j. If no constraint is applied, then $j = i$ is allowed, and the term $V_i \cdot Attention_{i,i}$ appears in the sum, meaning H_i directly contains information from s_i, violating the reconstruction objective.

To address this issue, we explore two masking strategies:

- **`MSK` Token-based Masking (Used in CHAP).** We randomly select a field s_i and replace its embedding with a special `MSK` token, ensuring no information about s_i is present in the input. This breaks the leakage path at the source. The reconstruction is still computed as:

$$\hat{s}_i = Decoder(H_i), \quad H_i = \sum_{j=1}^{n} V_j \cdot Attention_{i,j} \tag{A-8}$$

However, since V_i corresponds to the `MSK` token, it carries no semantic information of s_i. Therefore, even if $Attention_{i,i} \neq 0$, no leakage occurs. This strategy also allows the use of skip connections, benefiting training stability.

- **Diagonal-zero Masking.** As an alternative, we eliminate the self-attention term by explicitly masking out the diagonal of the causal mask M:

$$M_{i,i} = 0, \quad \forall i \tag{A-9}$$

This forces $Attention_{i,i} = 0$, preventing V_i from influencing H_i in Eq. A-1. This method allows for fully parallelized reconstruction within a batch. However, in this design, skip connections must be disabled to fully block access to s_i, which may lead to training instability or reduced performance.

Empirically, we find that the `MSK`-based masking method yields better reconstruction accuracy and more stable training dynamics.

Theoretical Basis of Causal DAG Assumption

The assumption of a causal Directed Acyclic Graph (DAG) $G = (V, E)$ is a foundational concept in modern causal inference. It encodes the data-generating process via a set of structural causal models (SCMs), where each variable is a deterministic function of its parents and some exogenous noise:

$$v_i = f_i(\mathrm{Pa}(v_i), \epsilon_i), \quad \epsilon_i \perp\!\!\!\perp \epsilon_j \ (i \neq j) \tag{A-10}$$

Here, $\mathrm{Pa}(v_i)$ denotes the set of parent nodes (i.e., direct causes) of v_i in G, and the noise terms ϵ_i are mutually independent.

Under this assumption, the joint distribution $P(V)$ satisfies the *Causal Markov Condition* (Pearl 2009), i.e.,

$$P(V) = \prod_{i=1}^{d+1} P(v_i \mid \mathrm{Pa}(v_i)) \tag{A-11}$$

This factorization allows the DAG to represent conditional independencies via the concept of d-separation (Peters, Janzing, and Schlkopf 2017). When G is a true causal DAG, removing non-parent edges avoids spurious correlations caused by either confounding paths or anti-causal directions, thus improving generalization across different environments.

Datasets

All datasets are publicly available on Kaggle or the UCI Machine Learning Repository. Data needs to be processed into CSV format, with some datasets requiring special processing. The specific download links and preprocessing details for each dataset are as follows:

Adult Dataset

Source: https://archive.ics.uci.edu/dataset/2/adult
Task: Binary classification (income prediction)
Preprocessing: None

Cardio Dataset

Source:https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset
Task: Multi-class classification (cardiotocography classification)
Preprocessing: Four features: height, weight, ap_hi, ap_lo are discretized into categorical variables with intervals of 10

CreditCard Dataset

Source:https://www.kaggle.com/datasets/arockiaselciaa/creditcardcsv
Task: Binary classification (credit card default prediction)
Preprocessing: None

Diamonds Dataset

Source:https://www.kaggle.com/datasets/shivam2503/diamonds
Task: Regression (diamond price prediction)
Preprocessing: The price column is moved to the last column as the prediction target

Elevator Dataset

Source:https://www.kaggle.com/datasets/shivamb/elevator-predictive-maintenance-dataset
Task: Regression (elevator behavior prediction)
Preprocessing: The vibration column is moved to the last column as the prediction target

Housesale Dataset

Source:https://www.kaggle.com/datasets/ruchi798/housing-prices-in-metropolitan-areas-of-india
Task: Regression (house price prediction)
Preprocessing: All 6 CSV files in the dataset are concatenated together, and the data is shuffled with random seed 42

Implementation Details

For all models, we use the Adam optimizer with a weight decay of 1×10^{-5}. The dropout rate is set to 0.1. For models based on multi-head self-attention, the number of heads is set to the hidden size divided by 8. Other hyperparameters used in the test datasets are selected according to the results on the validation dataset, and the selected values are then adopted to report the test performance. We tune the hyperparameters via grid search.

For **CHAP**, the grid for hidden size is [16, 32, 64, 128], and 32 is selected as optimal. The batch size is chosen from [64, 128, 256, 512], with 128 being best. The learning rate is selected from [1e-5, 1e-4, 1e-3, 1e-2], and 1e-3 is found optimal. The number of encoder layers is chosen from [1, 2, 3, 4, 5, 6], with 2 selected. The number of epochs is set to 20.

For **FT-Transformer** (Dai et al. 2025), the hidden size is selected from [16, 32, 64, 128], and 64 is chosen. The batch size is selected from [64, 128, 256, 512], with 128 being best. The learning rate is chosen from [1e-5, 1e-4, 1e-3, 1e-2], and 1e-3 is selected. The number of encoder layers is chosen from [1, 2, 3, 4, 5, 6], and 2 is selected. The number of epochs is 30.

For **Tab-Transformer** (Huang et al. 2020), the hidden size is selected from [16, 32, 64, 128], with 32 being optimal. The batch size is chosen from [64, 128, 256, 512], and 128 is used. The learning rate is selected from [1e-5, 1e-4, 1e-3, 1e-2], and 1e-3 is best. The encoder layers are chosen from [1, 2, 3, 4, 5, 6], and 6 is selected. The number of epochs is 30.

For **SAINT** (Somepalli et al. 2021), the hidden size is selected from [16, 32, 64, 128], and 64 is best. The batch size is selected from [64, 128, 256, 512], with 128 chosen. The learning rate is selected from [1e-5, 1e-4, 1e-3, 1e-2], and 1e-4 is found optimal. The number of encoder layers is selected from [2, 4, 6, 8], and 4 is chosen (as each SAINT block contains both row and column attention). The number of epochs is 50.

For **CASTLE** (Kyono, Zhang, and Schaar 2020), the hidden size is selected from [16, 32, 64, 128], and 32 is best. The batch size is chosen from [64, 128, 256, 512], and 64 is optimal. The learning rate is selected from [1e-5, 1e-4, 1e-3, 1e-2], and 1e-4 is found optimal. The number of MLP layers is selected from [1, 2, 3, 4], and 2 is chosen. The special hyperparameters α and β are set according to the paper's recommendation as 1 and 5, respectively. The number of epochs is 50.

For **LogCause** (Janzing 2019), the hidden size is selected from [16, 32, 64, 128], and 32 is used. The batch size is selected from [64, 128, 256, 512], with 64 chosen. The learning rate is selected from [1e-5, 1e-4, 1e-3, 1e-2], and 1e-5 is optimal. The number of epochs is 200.

References

Dai, H.; Wu, S.; Zhao, H.; Huang, J.; Jian, Z.; Zhu, Y.; Hu, H.; and Chen, Z. 2025. FT-Transformer: Resilient and Reliable Transformer with End-to-End Fault Tolerant Attention. arXiv:2504.02211.

Huang, X.; Khetan, A.; Cvitkovic, M.; and Karnin, Z. 2020. TabTransformer: Tabular Data Modeling Using Contextual Embeddings. arXiv:2012.06678.

Janzing, D. 2019. Causal Regularization. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Kyono, T.; Zhang, Y.; and Schaar, M. 2020. CASTLE: Regularization via Auxiliary Causal Graph Discovery. *Neural Information Processing Systems,Neural Information Processing Systems*.

Pearl, J. 2009. *Causality: Models, Reasoning and Inference*. USA: Cambridge University Press, 2nd edition. ISBN 052189560X.

Peters, J.; Janzing, D.; and Schlkopf, B. 2017. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press. ISBN 0262037319.

Somepalli, G.; Goldblum, M.; Schwarzschild, A.; Bruss, C. B.; and Goldstein, T. 2021. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. *CoRR*, abs/2106.01342.