PROMPT-CMI: IN-CONTEXT ESTIMATION OF CONDITIONAL MUTUAL INFORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Conditional mutual information (CMI) measures the dependence between two variables that remains after conditioning on others, and it is the quantity behind conditional independence testing, feature selection and dependence discovery. Statistical estimators lose the signal as dimension grows, and neural estimators must be trained anew on every dataset. We introduce Prompt-CMI, a network pretrained once on synthetic tasks that estimates CMI on unseen tables with frozen parameters. Prompt-CMI reads a context of observations from the table and assigns every observation an information score, trained to approximate the log-density ratio between the joint law and its conditionally independent reference; the average score is the CMI estimate. A context encoder summarizes the context into a fixed-size memory for any sample size and variable dimensions, and a synthetic task generator supplies exact reference samples and known CMI for pretraining. On synthetic benchmarks, Prompt-CMI attains the lowest error across seven dependence mechanisms, up to 60 dimensions and CMI levels from 0 to 6 nats, at over a hundredfold lower cost than a conventional neural estimator, and its frozen scores serve for weak-dependence detection, feature selection and conditional independence testing on real data.

1 INTRODUCTION

Conditional mutual information (CMI), $I(X;Y \mid Z)$, measures how much information X and Y share after accounting for background variables Z, linear or nonlinear. It drives conditional independence testing, feature selection and dependence analysis, from nonrigid image registration (Loeckx et al., 2010) to protein modulation discovery in gene-expression data (Giorgi et al., 2014). Such analyses estimate CMI many times over, for candidate features, conditioning sets or time lags, and the available estimators fail them in one of two ways. Estimators built on distributional assumptions or local sample geometry lose the signal as dimension grows: with 20-dimensional X and Y and a true CMI of 4 nats, the nearest-neighbor estimator reports 0.26 nats (Table 1) (Frenzel & Pompe, 2007). Neural estimators model the dependence but must be trained anew on every dataset (Mukherjee et al., 2020; Molavipour et al., 2020; 2021; Duong & Nguyen, 2023), which costs eight to forty-four seconds per table in the same setting and leaves estimates that vary by more than a nat across samples of the same law (Table 1). What is missing is a neural estimator that reuses what it has learned across datasets.

Training one neural estimator on many datasets does not supply it by itself: the log-density ratio it learns is specific to the law of each table, so a network that receives an observation alone cannot score it. It must also see the table, and tabular foundation models show how (Hollmann et al., 2025; Grinsztajn et al., 2026): a network pretrained on synthetic tasks reads a table as context and performs inference on it with frozen parameters. Applied to CMI, the classifier that separates joint from conditionally independent samples can be conditioned on a context of observations, and synthetic pretraining supplies what no real table has, namely exact reference samples and the true CMI of the generating law. The score of an observation is then trained toward the log-density ratio of the law that generated its context, the average score over the table is the CMI, and the context tells the scorer which law it faces, so a new table needs no gradient step (Figure 1).

We build this estimator as **Prompt-CMI**: an architecture that summarizes a context of any sample size and role dimensions into a fixed-size memory, up to 60 dimensions with $(d_x, d_y, d_z) =$

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

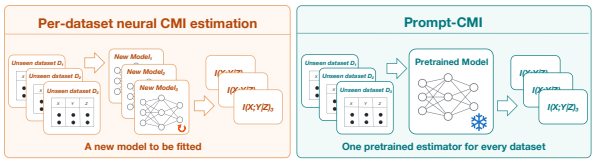

Figure 1: **CMI estimation through pretraining.** Conventional neural estimators train a separate model on each table; Prompt-CMI learns from synthetic tasks and applies one frozen network to every unseen table.

$(20, 20, 20)$, and a synthetic task generator that varies dependence mechanisms, conditional effects, variable types and observation maps while keeping the CMI known. On synthetic benchmarks, Prompt-CMI attains the lowest error across seven mechanisms and nine information levels and in five of six dimension settings, at **over a hundredfold lower cost** than a conventional neural estimator, and its frozen scores detect weak dependence, rank dependence strength, select features and test conditional independence without per-dataset training.

2 Problem Formulation

CMI estimation. We observe a table $\mathcal{D} = \{u_i\}_{i=1}^N$ of observations $u_i = (x_i, y_i, z_i)$ drawn independently from an unknown joint law P with density $p(x, y, z)$. We call X, Y and Z the three *roles* of the table; they have dimensions (d_x, d_y, d_z), so an observation has $D = d_x + d_y + d_z$ variables, each continuous or categorical. The conditional mutual information (CMI) between X and Y given Z compares P with the *reference law* Q, whose density $q(x, y, z) = p(z)p(x \mid z)p(y \mid z)$ keeps the conditionals of P but makes X and Y independent given Z:

$$I(X; Y \mid Z) = \mathrm{KL}(p(x, y, z) \,\|\, p(z)p(x \mid z)p(y \mid z)) = \mathbb{E}_{u\sim P}\left[\log \frac{p(u)}{q(u)}\right]. \tag{1}$$

Every estimator below approximates the second form, the average pointwise log-density ratio over joint observations. Statistical estimators such as the Frenzel–Pompe estimator replace the densities by nearest-neighbor counts (Frenzel & Pompe, 2007) (Appendix A) and lose accuracy as the dimension grows, because neighborhoods become sparse. Neural estimators learn the ratio instead (Mukherjee et al., 2020; Molavipour et al., 2020; 2021; Duong & Nguyen, 2023). CMINE, for example, trains a classifier to tell joint samples from reference samples with a balanced cross-entropy (Appendix A, Eq. 21); at the optimum its log odds equal $\log[p/q]$, so their average over joint samples estimates the CMI through (1). Because only the table is available, the reference samples are built from $\mathcal{D}$ itself by nearest-neighbor resampling in Z, which only approximates Q.

Challenge of real-time neural CMI estimation. Each such estimator is fitted anew to every table by T gradient updates $\phi^{t+1} \leftarrow \phi^t - \eta \nabla_\phi \mathcal{L}(\phi^t; \mathcal{D})$ on its objective $\mathcal{L}$, with learning rate η, before an estimate exists. Our goal is one estimator $f_{\theta^\star}$, learned once and then frozen, that processes a new table by inference alone:

$$\widehat{I} = f_{\theta^\star}(\mathcal{D}). \tag{2}$$

3 In-Context CMI Estimation

Prompt-CMI keeps this mechanism, a scorer whose average over joint observations is the CMI, but removes the per-table fit by making the scorer a function of the table itself. A subset $C \subset \mathcal{D}$ of n_C observations, the *context*, replaces the training set: a network with frozen parameters θ reads C and assigns to any observation u, the *query*, an *information score* $T_\theta(u; C) \in \mathbb{R}$, which pretraining (Section 4) trains to approximate $\log[p(u)/q(u)]$ under the law that generated C. We call this in-context estimation: the context is the prompt that adapts the frozen network to the table, and no gradient step is taken on a new table.

3.1 COPULA PREPROCESSING

The scorer must accept variables with arbitrary marginals and scales, whereas the CMI ignores them: for invertible maps f_X, f_Y and f_Z applied separately to the three roles,

$$I(f_X(X); f_Y(Y) \mid f_Z(Z)) = I(X; Y \mid Z), \tag{3}$$

so the CMI depends only on the *copula* of the law, the dependence left once every marginal is standardized. Copula preprocessing standardizes the marginals and decorrelates each role with two role-wise invertible maps, so that pretraining need not cover them; both maps are fitted on the context and applied to context and queries alike.

Rank-Gaussian transform. Each continuous variable j is mapped to a standard Gaussian scale through its empirical rank in the context,

$$g_{C,j}(v) = \Phi^{-1}\Big(\widetilde{F}_{C,j}(v)\Big), \tag{4}$$

where $\widetilde{F}_{C,j}$ is the empirical CDF of variable j on the context and Φ^{-1} is the standard Gaussian quantile function. The transformed variable follows the standard Gaussian quantiles over the context whatever its original marginal, and the map is monotone in each coordinate, so (3) applies.

Role-wise whitening. Coordinates within one role can be strongly correlated. By (3) this correlation carries no CMI, but it shapes every token the network sees, so the scorer would otherwise have to learn to disregard every within-role covariance. Regularized ZCA whitening removes it on the context, $w^r_C = W^r_C(g^r_C - \mu^r_C)$, where g^r_C collects the rank-Gaussian coordinates of role $r \in \{X, Y, Z\}$, μ^r_C is their context mean and W^r_C their whitening matrix with small eigenvalues floored (Appendix A). Since whitening mixes the coordinates of a role, the preprocessing map Ψ_C keeps both the rank-Gaussian and the whitened coordinates as network inputs. Categorical variables enter through category statistics computed on the context (Appendix A).

3.2 CMI ESTIMATOR ARCHITECTURE

Architecture overview. Every query is scored against the same context, so the context is encoded once. The network has two parts (Figure 2): a *context encoder* turns the preprocessed context into a *context memory*, a variable memory $\mathcal{A}_C$ summarizing each variable and a table memory M_C summarizing the joint distribution of all roles, and an *information decoder* scores each query by reading that memory.

3.2.1 CONTEXT ENCODER

Axial encoding. The encoder must accept any number of observations and any role dimensions with one set of parameters, which a fixed-width input vector, as used by the estimators of Section 2, cannot do. Axial encoding instead turns each preprocessed entry into a token of width d, computed from its value and embeddings of its type (continuous or categorical) and its role, where X and Y share one role embedding. The tokens then exchange information along both axes of the table. Within an observation, attention connects its D tokens. Within a variable, attention across the n_C observations would cost $O(n_C^2)$, so K_c learned inducing vectors attend to the variable's tokens and the tokens attend back to them (Lee et al., 2019), at cost $O(n_C K_c)$. Stacked blocks of the two attentions give

$$(V_C, \mathcal{A}_C) = \mathrm{AxialEnc}_\theta\big(\mathrm{Embed}_\theta(\Psi_C(C))\big), \qquad V_C \in \mathbb{R}^{n_C \times D \times d}, \tag{5}$$

where V_C holds the updated tokens and the variable memory $\mathcal{A}_C$ holds the K_c inducing vectors of every variable after the last block. We use $d = 128$ and $K_c = 128$.

Role pooling. The table memory and the score head need one vector of fixed width per observation, whatever (d_x, d_y, d_z) is, and that vector must respect the symmetry of the CMI in X and Y. For observation i, learned pooling vectors attend to its tokens of X, of Y, of Z and of all variables, giving a_i^X, a_i^Y, a_i^Z and a_i^{all} in $\mathbb{R}^d$, which are combined as

$$h_i = \big[a_i^X + a_i^Y \,;\, |a_i^X - a_i^Y| \,;\, a_i^Z \,;\, a_i^{\mathrm{all}}\big]. \tag{6}$$

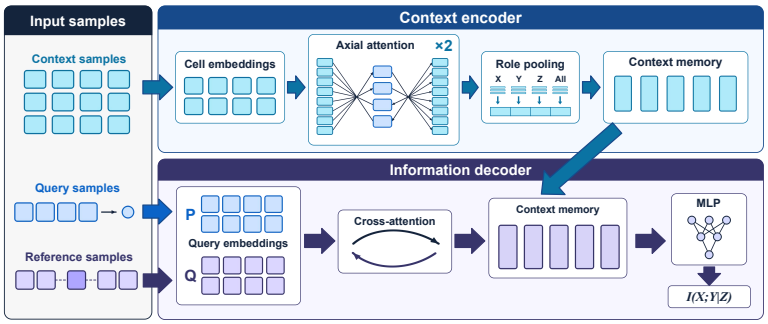

Figure 2: **Prompt-CMI estimator architecture.** The context encoder turns the context observations into a context memory (variable memory and table memory); the information decoder reads it to assign each query an information score, whose average over joint observations is the CMI estimate. In pretraining, joint and reference queries supervise the scores.

The sum and absolute difference are symmetric, and the pooling parameters are shared between X and Y, so exchanging the roles leaves h_i unchanged by construction; the all-variable summary retains cross-role interactions omitted by the three role summaries. The rows h_i form $H_C \in \mathbb{R}^{n_C \times 4d}$.

Context summarization. Attending from each query to all n_C rows of H_C would make the cost of a query grow with the table, and a single pooled vector could not represent a law whose dependence changes with Z, such as a correlation whose sign depends on Z. Context summarization condenses H_C into K_r learned memory vectors, which gather information from H_C by cross-attention and combine it by self-attention:

$$M_C = \mathrm{MemEnc}_\theta\Big(\mathrm{Pool}_\theta^{K_r}(H_C)\Big) \in \mathbb{R}^{K_r \times 4d}. \tag{7}$$

The table memory has the same size, $K_r = 512$ vectors, for any n_C; cached with $\mathcal{A}_C$, it forms the context memory.

3.2.2 Information Decoder

Scoring a query. A query u is preprocessed with the context-fitted Ψ_C, tokenized, and passed through the same axial encoder and role pooling, so that it is represented in the space of the context observations; in the cross-observation step its tokens read the cached $\mathcal{A}_C$ without updating it, so the context memory never depends on the query. Its pooled vector $h_u \in \mathbb{R}^{4d}$ then attends to M_C by cross-attention, and an MLP head outputs the information score $T_\theta(u; C)$. Because the context memory is cached, the cost of a query does not grow with n_C (Appendix A).

Cross-fitted readout. At inference every observation is a sample from the joint law of the table, so by (1) the estimate is the average score over observations. Scoring an observation against a context that contains it would let the scorer recognize it as joint by matching it to a context row, and inflate the estimate. We therefore split $\mathcal{D}$ into four folds $F_1, \ldots, F_4$, encode $C_k = \mathcal{D} \setminus F_k$ as context and score the held-out fold F_k against it. Each observation is scored once, with θ frozen:

$$\widehat{I}_\theta(\mathcal{D}) = \frac{1}{N} \sum_{k=1}^{4} \sum_{u_i \in F_k} T_\theta(u_i; C_k). \tag{8}$$

4 Large-Scale Synthetic Pretraining

No real table supplies the log-density ratio that the score must approximate; a law we construct does, because joint and reference samples can be drawn from it exactly and, for most constructions, its CMI is known in closed form. We therefore pretrain on a distribution Π of synthetic tasks.

4.1 Pretraining Objective

A task $\tau \sim \Pi$ consists of a generating law ℓ and a context size n_C; the law fixes a joint law P_ℓ with density $p_\ell(x, y, z)$ and its reference law Q_ℓ with density $q_\ell = p_\ell(z)p_\ell(x \mid z)p_\ell(y \mid z)$. From the task we draw, independently, the context C and *joint queries* U_P from P_ℓ, and *reference queries* U_Q from Q_ℓ (Figure 3).

Learning information scores. Classification needs only samples from the two laws, not their densities. The model predicts the probability $\sigma(T_\theta(u; C))$ that a query is joint rather than reference, where σ is the sigmoid function, and is trained by the balanced binary cross-entropy

$$\mathcal{L}_{\text{cls}}(\theta; \tau) = -\frac{\sum_{u \in U_P} \log \sigma(T_\theta(u; C))}{2|U_P|} - \frac{\sum_{u \in U_Q} \log \sigma(-T_\theta(u; C))}{2|U_Q|}. \tag{9}$$

For a fixed law, the minimizer is $T_\theta(u; C) = \log[p_\ell(u)/q_\ell(u)]$, whose average over joint queries is the CMI (Appendix A). Across tasks the model sees only C, so the loss drives $T_\theta(u; C)$ toward the log ratio of the joint and reference densities predicted from C (Appendix A, Eq. 13), the closest approximation to the pointwise ratio that the context supports.

Calibrating CMI estimates. The estimate is the mean joint score $\widehat{I}_\theta(U_P; C) = |U_P|^{-1} \sum_{u \in U_P} T_\theta(u; C)$, the average that (8) takes fold by fold. Classification constrains this mean only through the pointwise scores, and with a finite context the predicted ratio need not average to the CMI. For tasks whose CMI $I^\star_\ell$ is known (Section 4.2), a calibration loss $\mathcal{L}_{\text{cal}}$ therefore penalizes the difference between $\widehat{I}_\theta(U_P; C)$ and $I^\star_\ell$ on a compressive scale, so that tasks with large CMI do not dominate (Appendix A), and the shared parameters are trained on

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau \sim \Pi}[\mathcal{L}_{\text{cls}}(\theta; \tau) + \lambda \mathcal{L}_{\text{cal}}(\theta; \tau)], \tag{10}$$

where the weight λ is increased gradually during pretraining (Appendix A).

4.2 Pretraining Task Generation

Diverse task generation. The score is only as general as the laws seen in pretraining, so Π varies dependence structures rather than particular distributions. A law fixes a dependence core, which decides which latent signals X and Y share and how much conditional information they carry; conditional effects of Z on means, scales or signs; the variable types, continuous, categorical or mixed; and observation maps, invertible within each role at fixed z, which reshape the latent variables into the observed table without changing the CMI by (3). Role dimensions range from 1 to 20, context sizes from 48 to 10,000 and target CMI from 0 to 8 nats (Appendix B).

Conditional reference construction. On a real table, reference samples can only be approximated by resampling y among nearest neighbors in Z, and the approximation error enters the score as bias; a synthetic law is sampled exactly. Given $z \sim p_\ell(z)$, we draw two independent pairs $(x^{(a)}, y^{(a)})$ and $(x^{(b)}, y^{(b)})$ from $p_\ell(x, y \mid z)$ and form

$$\begin{aligned} u_P &= (x^{(a)}, y^{(a)}, z) \sim P_\ell, \\ u_Q &= (x^{(a)}, y^{(b)}, z) \sim Q_\ell. \end{aligned} \tag{11}$$

The complete y vector is exchanged, preserving within-Y dependence, so u_P and u_Q differ only in the conditional dependence between X and Y; exchanging individual coordinates could instead let the classifier exploit broken within-Y dependence. The exchange is applied after the observation maps so that reference queries follow the same input distribution as observed tables.

5 Experiments

The Method makes three claims: a scorer that reads the law from the context estimates CMI accurately with frozen parameters across mechanisms, dimensions, information levels and sample sizes at a fraction of the cost of per-dataset fitting (Section 5.1, CMI estimation); its scores separate conditional independence from weak dependence and follow the strength of dependence (Section 5.1,

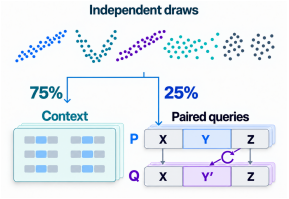

Figure 3: **Synthetic pretraining tasks.** Each task fixes a generating law and draws independent context observations and joint queries from it. A second conditional draw at the same Z supplies a complete Y vector, which replaces the original to form a reference query. The two labels supervise classification; tasks with known CMI calibrate the mean joint score.

Table 1: **CMI estimation across distributions and dimensions.** Both settings use $N = 1024$. Cells show estimated CMI as mean $\pm$ sample SD over three sampling seeds for each fixed distribution. All methods use the same sampled tables; the seven mechanisms are described in Appendix C. Bold and underlined estimates mark the lowest and second-lowest MAE; t is the average runtime per dataset in seconds over three runs, including fitting when required.

Method	Gaussian	Student-t	Cauchy	Copula	Heterosk.	Switch	Half-cube	t (s)
				$(d_x, d_y, d_z) = (4, 4, 8)$, $I^\star = 0.8$ nats				
Prompt-CMI	**0.81±0.05**	**0.85±0.09**	**0.79±0.09**	0.62±0.05	**0.78±0.02**	**0.98±0.10**	**0.81±0.03**	**0.12**
kNN	0.30±0.01	0.29±0.02	0.14±0.02	0.18±0.02	0.24±0.02	0.15±0.005	0.24±0.03	0.19
CCMI	0.57±0.04	0.61±0.06	0.13±0.03	0.53±0.03	0.53±0.07	0.05±0.01	0.45±0.01	43.56
CMINE	0.60±0.35	0.76±0.16	0.08±0.04	0.66±0.20	0.66±0.10	0.14±0.03	0.78±0.07	9.84
DINE	0.76±0.06	0.66±0.07	0.22±0.03	**0.77±0.11**	0.73±0.007	0.01±0.007	0.60±0.04	8.08
				$(d_x, d_y, d_z) = (20, 20, 8)$, $I^\star = 4$ nats				
Prompt-CMI	**4.45±0.22**	**3.75±0.34**	**5.54±0.97**	4.81±0.70	**3.64±0.22**	**1.26±0.33**	**3.31±0.57**	**0.26**
kNN	0.26±0.005	0.48±0.02	0.21±0.01	0.18±0.02	0.17±0.02	0.10±0.01	0.19±0.02	0.42
CCMI	1.01±0.11	1.05±0.05	0.47±0.03	0.61±0.05	0.36±0.16	-0.17±0.03	0.38±0.10	44.19
CMINE	3.98±1.49	2.29±0.53	0.92±0.35	1.66±0.88	2.43±1.27	0.03±0.06	0.99±0.06	9.98
DINE	3.41±0.19	3.34±0.19	1.69±0.27	**3.22±0.24**	3.24±0.02	-0.003±0.03	2.28±0.08	8.22

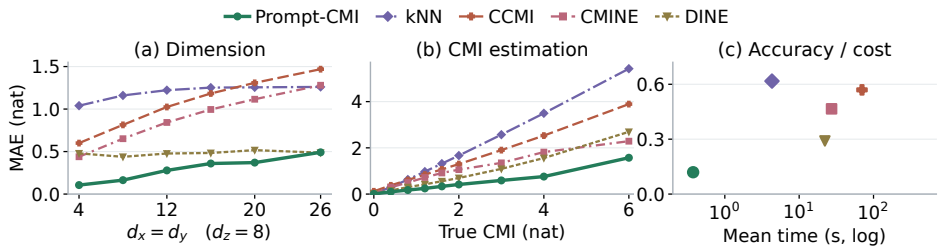

Figure 4: **Accuracy, scalability and cost across seven mechanisms.** MAE averaged over three seeds per mechanism and setting; lower is better. (a) $N = 1024$, $(d_x, d_y, d_z) = (m, m, 8)$ with $m \in \{4, 8, 12, 16, 20, 26\}$ and $I^\star \in \{0.8, 2\}$. (b) $N = 1024$, MAE over true CMI levels of 0–6 nats, averaged across $(4, 4, 8)$ and $(20, 20, 8)$. (c) $I^\star = 0.8$, both role configurations, six sample sizes from 256 to 8000. Appendix C gives the aggregation and timing protocol.

detection; Section 5.2, order preservation); and the frozen scores serve as a reusable statistic in downstream procedures on real data (Section 5.2, feature selection and conditional independence testing). We compare Prompt-CMI with nearest-neighbor estimation (kNN), neural CMI estimators (CCMI, CMINE and DINE) and task-specific baselines on shared tables and sampling repeats; Prompt-CMI stays frozen, while the neural baselines are fitted to each table (Appendix C).

5.1 Synthetic Data

CMI estimation. This experiment tests the first claim on seven mechanisms with known CMI (Appendix C), varying dimension, target CMI and sample size. Table 1 compares estimates at two dimensions and target CMI. Prompt-CMI achieves the lowest mean absolute error (MAE) across mechanisms in both settings, 0.080 nats at $(d_x, d_y, d_z) = (4, 4, 8)$ and 0.983 nats at $(20, 20, 8)$, where DINE, the strongest baseline, reaches 1.547 nats. It ranks first on six of seven mechanisms in each setting. Figure 4 extends the comparison: Prompt-CMI has the lowest MAE in five of six role

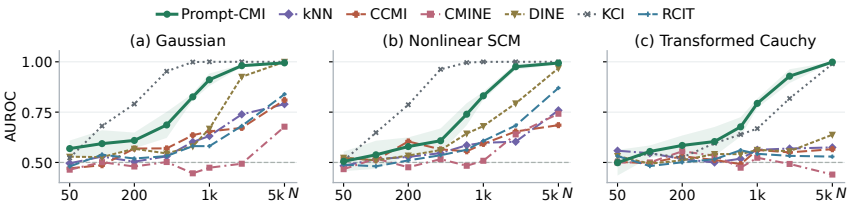

Figure 5: **Detecting weak conditional dependence.** Role dimensions $(20, 20, 20)$, target CMI 0 or 0.1 nat, sample sizes N from 50 to 5000. Curves show mean AUROC over five groups of 50 null and 50 alternative tables per N, shared across methods; higher is better. The dashed line marks chance and the shading ± 1 SD across groups for Prompt-CMI.

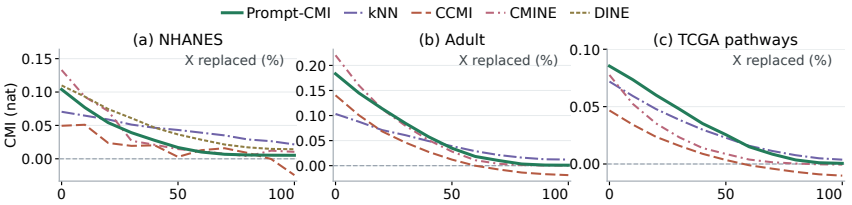

Figure 6: **Order preservation.** CMI estimates across 11 replacement levels from 0% to 100%, where X rows are replaced by Z-matched donors while Y and Z stay fixed. NHANES, Adult and TCGA pathways use $N = 4792, 5000, 5000$ with role dimensions $(32, 1, 3)$, $(32, 1, 1)$ and $(8, 1, 1)$; results average three seeds per task with equal task weighting within each dataset. DINE is reported only on NHANES, as Adult and TCGA have categorical targets. The dashed line marks zero CMI.

configurations and at all nine target CMI levels from 0 to 6 nats, and on the accuracy–cost benchmark its MAE is less than half DINE's at an average runtime of 0.37 seconds per dataset. In the timing comparison of Table 1 at $(20, 20, 8)$, it is approximately $39\times$ and $32\times$ as fast as CMINE and DINE. The context thus supplies what per-dataset fitting supplies for the baselines, and the cached context memory keeps the cost low; this supports the first claim.

Conditional dependence detection. This experiment tests the first half of the second claim: whether the scores separate conditional independence from weak dependence. We use the Gaussian, Nonlinear SCM and Transformed Cauchy mechanisms with $(d_x, d_y, d_z) = (20, 20, 20)$ and CMI of 0 or 0.1 nat, and measure by AUROC how well the scores separate conditionally independent from weakly dependent tables as the sample size grows.

At $N = 2000$, Prompt-CMI reaches AUROCs of 0.981, 0.976 and 0.929 on Gaussian, Nonlinear SCM and Transformed Cauchy (Figure 5). On Transformed Cauchy, its AUROC rises from 0.794 at $N = 1000$ to 0.999 at $N = 5000$, and at $N = 5000$ it exceeds 0.99 on all three mechanisms, while the three neural baselines reach 0.440–0.638 on Transformed Cauchy. A 0.1-nat dependence in 60 dimensions is thus visible in the scores once the context is large enough, as the classification objective asks of them.

5.2 Real-World Tasks

Order preservation. This experiment tests the second half of the second claim on real data: whether the scores follow the strength of dependence as it is weakened. We replace increasing fractions of X rows with Z-matched donor rows while keeping Y and Z fixed, and all methods score the same perturbed tables. The inversion rate is the fraction of replacement-level pairs whose scores violate the expected decreasing order. We apply this protocol to NHANES health survey data (CDC/NCHS, 2013–2014), Adult census data (Becker & Kohavi, 1996) and TCGA gene-expression pathways (The Cancer Genome Atlas Research Network et al., 2013).

Across all three datasets, Prompt-CMI's mean score decreases at every replacement level and approaches zero after full replacement (Figure 6). On NHANES it drops from 0.104 to 0.005 nat with no inversions in any seed. Its inversion rate, averaged equally across datasets, is 0.50%, versus 0.77% for kNN, 9.77% for CMINE and 8.88% for CCMI; on Adult and TCGA pathways, its rates

Table 2: **CMI feature selection for regression.** Test NRMSE (lower is better) of an identically configured MLP trained on the k selected feature groups together with Z. Hydraulic uses 24 pressure features conditioned on temperature variables (four temporal splits, three seeds each); SARCOS uses 14 velocity–acceleration features conditioned on joint positions (official split, three seeds). Bold and underlined values mark the best and second-best selectors; Appendix C gives the protocol.

	Hydraulic			SARCOS		
Method	$k=1$	$k=2$	$k=4$	$k=1$	$k=2$	$k=4$
Prompt-CMI	**0.1690**	<u>0.2625</u>	<u>0.2235</u>	0.2031	**0.1504**	**0.1130**
kNN	0.2844	0.3871	0.4055	<u>0.2013</u>	0.1649	**0.1130**
CCMI	0.3434	0.4298	0.3964	0.2155	0.1631	0.1207
CMINE	0.3715	0.9667	0.6567	0.2186	0.1704	0.1296
DINE	0.3279	0.3265	0.2606	0.2094	0.1648	0.1215
Marginal MI	<u>0.2377</u>	**0.2215**	**0.2117**	<u>0.2013</u>	0.1726	0.1311
mRMR	<u>0.2377</u>	0.5748	0.4655	<u>0.2013</u>	0.1565	0.1181
Val. selection	0.3330	0.3006	0.2652	**0.1980**	<u>0.1506</u>	<u>0.1145</u>
Z-only	0.4587	0.4587	0.4587	0.3154	0.3154	0.3154
All inputs	0.6336	0.6336	0.6336	0.0979	0.0979	0.0979

are approximately one tenth and one seventh of CMINE's. The mean score thus tracks the strength of dependence on real tables and returns to zero when it is gone, as calibration asks of it.

CMI feature selection. This experiment tests the third claim with a growing conditioning set: whether the frozen scores identify compact feature subsets that retain predictive information. Using training data only, each CMI method greedily selects the group with the largest estimated $I(G; Y \mid Z, S)$, where S denotes the previously selected groups, until $k \in \{1, 2, 4\}$ groups are chosen. Each selected subset, together with Z, is evaluated by an identically configured MLP using test-set NRMSE. We apply this protocol to flow prediction on Hydraulic (Helwig et al., 2015) and torque prediction on SARCOS (Vijayakumar et al., 2005).

Prompt-CMI achieves lower NRMSE than all three neural CMI baselines in all six dataset–budget combinations and ranks first or tied first among same-budget selectors in three (Table 2). Its NRMSE is 0.1690 with one Hydraulic group, versus 0.2377 for marginal MI and mRMR, and 0.1504 and 0.1130 with two and four SARCOS groups. The frozen scores thus keep identifying predictive groups as S grows, a setting in which every step changes the dimension of the conditioning role and the same network handles each.

Conditional independence testing. Finally, this experiment evaluates the third claim by using Prompt-CMI as a reusable test statistic on real-world data. Each variable pair is tested conditional on all remaining variables; CMI-based methods obtain p-values via shared conditional resampling, and AUROC measures how well the resulting scores $1 - p$ separate edges from nonedges in domain reference networks. The benchmarks include Sachs protein-signaling data (Sachs et al., 2005), Chamber and Camera-Beta physical measurements (Gamella et al., 2025), and financial returns from India NSE (qubiit, 2026) and European institutions (Hasse, 2020). Reference edges are defined by moralized biological or physical graphs for the scientific datasets and by sector or institution membership for the financial datasets. Prompt-CMI remains frozen throughout.

Prompt-CMI ranks among the top two methods on all five datasets and achieves the highest mean AUROC on Sachs and Camera-Beta (Table 3). It outperforms the neural CMI baselines on all five datasets: a frozen score inside a resampling test recovers conditional relationships without per-dataset training, which supports the third claim.

6 Related Work

MI and CMI estimation. Estimators differ in what supplies the density ratio. Statistical estimators read it from local sample neighborhoods (Gao et al., 2017; Frenzel & Pompe, 2007); neu-

Table 3: **Conditional independence testing on five real-world datasets.** AUROC of $1 - p$ (higher is better), using $N = 512$ observations per split or window. Entries report mean $\pm$ sample SD over five splits for Sachs, Chamber and Camera-Beta, and ten overlapping windows across two equally weighted periods for each financial dataset. CMI-based tests use 199 shared conditional resamples. Bold and underlined values mark the best and second-best mean AUROC in each column.

Method	Sachs	Chamber	Camera-Beta	India NSE	Europe
Prompt-CMI	**0.751±0.034**	0.653±0.029	**0.696±0.056**	0.950±0.021	0.788±0.036
kNN	0.733±0.064	0.536±0.038	0.575±0.078	0.884±0.018	**0.791±0.038**
CCMI	0.707±0.086	0.622±0.055	0.628±0.039	0.771±0.111	0.565±0.091
CMINE	0.646±0.095	0.603±0.009	0.535±0.027	0.642±0.126	0.564±0.067
KCI	0.705±0.043	0.592±0.035	0.593±0.079	0.758±0.043	0.565±0.096
RCIT	0.690±0.061	0.572±0.021	0.584±0.056	0.888±0.034	0.644±0.082
Fisher-Z	0.738±0.061	**0.679±0.052**	0.676±0.049	**0.956±0.033**	0.745±0.066

ral estimators fit it per dataset, through variational bounds as in MINE (Belghazi et al., 2018), classification-based density ratios as in CCMI and CMINE (Mukherjee et al., 2020; Molavipour et al., 2021), or invertible transformations that simplify the estimate as in DINE (Duong & Nguyen, 2023). All compute or fit per dataset; Prompt-CMI learns one information scorer across synthetic tasks and reuses it, frozen, on unseen tables.

Foundation models for statistical inference. Prior-data fitted networks and tabular foundation models learn reusable inference procedures by pretraining on synthetic tasks (Müller et al., 2022; Hollmann et al., 2025; Qu et al., 2026), and InfoNet and InfoAtlas bring this idea to mutual information without test-time optimization (Hu et al., 2024; 2026). Prompt-CMI extends the direction to conditional dependence with exact conditional reference generation for pretraining and a role-aware context memory of inducing and latent vectors (Lee et al., 2019; Jaegle et al., 2022) for any sample size and variable dimensions.

Conditional independence testing. Representative approaches include Gaussian partial-correlation tests (Kalisch & Bühlmann, 2007), kernel-based tests such as KCI (Zhang et al., 2011), random-feature approximations such as RCIT (Strobl et al., 2019), and information-based tests that combine a CMI statistic with conditional resampling to obtain a null reference (Runge, 2018). Prompt-CMI supplies the statistic for the last family without fitting it to each dataset; our experiments test it on weak dependence in synthetic data and on real-world reference networks.

7 Discussion

Taken together, the results establish that one network, pretrained once and frozen, estimates CMI on unseen tables with lower error than estimators fitted to each table in almost every setting across seven mechanisms, dimensions up to 60 and information levels from 0 to 6 nats, at 30–360$\times$ lower cost than the neural baselines, and that its scores carry over to detection, ranking, feature selection and conditional independence testing. It works because the information score is a log-density ratio read from the context rather than fitted to the table: the context adapts the scorer to the underlying task; copula preprocessing removes what carries no CMI, so pretraining covers dependence structures rather than distributions; and the synthetic generator supplies exact reference samples and known CMI, a supervision no real table provides. For the reader, CMI becomes a summary statistic rather than a training problem. The argument stops where the pretraining stops: generalization depends on the coverage of the task distribution; accuracy degrades at high information in high dimension (at $(20, 20, 8)$ and 4 nats, the Switch estimate misses the target by 2.7 nats, Table 1); and in conditional independence testing the error control rests on the resampling, not on the estimator. We plan two extensions that grow from what the method contains: pretraining tasks with lagged dependence, so that the same frozen scorer estimates transfer entropy in time series (Vicente et al., 2011; Runge et al., 2019), and a pretrained null distribution of the score, which the generator can supply, so that conditional independence tests need no resampling step.

Reproducibility statement

Appendices A and C describe the estimator and evaluation protocols, including frozen inference, shared inputs and result aggregation. Appendix D provides additional CMI estimates.

AI use statement

AI assistants helped with literature and code review, implementation, experiment coordination, figure preparation, translation and language editing. They also provided feedback on methodology and interpretation. The authors made the scientific decisions and remain responsible for the manuscript and its evidence.

References

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. URL https://doi.org/10.24432/C5XW20.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 531–540. PMLR, 2018. URL https://proceedings.mlr.press/v80/belghazi18a.html.

CDC/NCHS. National health and nutrition examination survey: 2013–2014 data, documentation, and codebooks. U.S. Department of Health and Human Services, Centers for Disease Control and Prevention, National Center for Health Statistics, 2013–2014. URL https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/default.aspx?BeginYear=2013.

Bao Duong and Thin Nguyen. Diffeomorphic information neural estimation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):7468–7475, 2023. doi: 10.1609/aaai.v37i6.25908. URL https://doi.org/10.1609/aaai.v37i6.25908.

Stefan Frenzel and Bernd Pompe. Partial mutual information for coupling analysis of multivariate time series. *Physical Review Letters*, 99(20):204101, 2007. doi: 10.1103/PhysRevLett.99.204101. URL https://doi.org/10.1103/PhysRevLett.99.204101.

Juan L. Gamella, Jonas Peters, and Peter Bühlmann. Causal chambers as a real-world physical testbed for AI methodology. *Nature Machine Intelligence*, 7:107–118, 2025. doi: 10.1038/s42256-024-00964-x. URL https://www.nature.com/articles/s42256-024-00964-x.

Weihao Gao, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Estimating mutual information for discrete-continuous mixtures. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/ef72d53990bc4805684c9b61fa64a102-Abstract.html.

Federico M. Giorgi, Gonzalo Lopez, Jung H. Woo, Brygida Bisikirska, Andrea Califano, and Mukesh Bansal. Inferring protein modulation from gene expression data using conditional mutual information. *PLOS ONE*, 9(10):e109569, 2014. doi: 10.1371/journal.pone.0109569. URL https://doi.org/10.1371/journal.pone.0109569.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026. doi: 10.48550/arXiv.2605.13986. URL https://arxiv.org/abs/2605.13986.

Jean-Baptiste Hasse. *SystemicR: Monitoring Systemic Risk*, 2020. URL https://CRAN.R-project.org/package=SystemicR. R package version 0.1.0.

Nikolai Helwig, Eliseo Pignanelli, and Andreas Schütze. Condition monitoring of hydraulic systems. UCI Machine Learning Repository, 2015. URL https://doi.org/10.24432/C5CW21.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025. doi: 10.1038/s41586-024-08328-6. URL https://doi.org/10.1038/s41586-024-08328-6.

Zhengyang Hu, Song Kang, Qunsong Zeng, Kaibin Huang, and Yanchao Yang. InfoNet: Neural estimation of mutual information without test-time optimization. *arXiv preprint arXiv:2402.10158*, 2024. doi: 10.48550/arXiv.2402.10158. URL https://arxiv.org/abs/2402.10158.

Zhengyang Hu, Yanzhi Chen, Hanxiang Ren, Qunsong Zeng, Youyi Zheng, Adrian Weller, Kaibin Huang, and Yanchao Yang. InfoAtlas: A foundation model for zero-shot statistical dependence estimate. *arXiv preprint arXiv:2606.00241*, 2026. doi: 10.48550/arXiv.2606.00241. URL https://arxiv.org/abs/2606.00241.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs and outputs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=fILj7WpI-g.

Markus Kalisch and Peter Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007. URL https://www.jmlr.org/papers/v8/kalisch07a.html.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Dirk Loeckx, Pieter Slagmolen, Frederik Maes, Dirk Vandermeulen, and Paul Suetens. Nonrigid image registration using conditional mutual information. *IEEE Transactions on Medical Imaging*, 29(1):19–29, 2010. doi: 10.1109/TMI.2009.2021843. URL https://doi.org/10.1109/TMI.2009.2021843.

Sina Molavipour, German Bassi, and Mikael Skoglund. Conditional mutual information neural estimator. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5025–5029. IEEE, 2020. doi: 10.1109/ICASSP40776.2020.9053422. URL https://doi.org/10.1109/ICASSP40776.2020.9053422.

Sina Molavipour, German Bassi, and Mikael Skoglund. Neural estimators for conditional mutual information using nearest neighbors sampling. *IEEE Transactions on Signal Processing*, 69:766–780, 2021. doi: 10.1109/TSP.2021.3050564. URL https://doi.org/10.1109/TSP.2021.3050564.

Sudipto Mukherjee, Himanshu Asnani, and Sreeram Kannan. CCMI: Classifier based conditional mutual information estimation. In Ryan P. Adams and Vibhav Gogate (eds.), *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 1083–1093. PMLR, 2020. URL https://proceedings.mlr.press/v115/mukherjee20a.html.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026. doi: 10.48550/arXiv.2602.11139. URL https://arxiv.org/abs/2602.11139.

qubiit. NIFTY-50 stock market data. GitHub data repository, 2026. URL https://github.com/qubiit/NIFTY-50-Stock-Market-Data. Accessed September 25, 2026.

Jakob Runge. Conditional independence testing based on a nearest-neighbor estimator of conditional mutual information. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 938–947. PMLR, 2018. URL https://proceedings.mlr.press/v84/runge18a.html.

Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5 (11):eaau4996, 2019. doi: 10.1126/sciadv.aau4996. URL https://doi.org/10.1126/sciadv.aau4996.

Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721): 523–529, 2005. doi: 10.1126/science.1105809. URL https://doi.org/10.1126/science.1105809.

Eric V. Strobl, Kun Zhang, and Shyam Visweswaran. Approximate kernel-based conditional independence tests for fast non-parametric causal discovery. *Journal of Causal Inference*, 7(1): 20180017, 2019. doi: 10.1515/jci-2018-0017.

The Cancer Genome Atlas Research Network, John N. Weinstein, Eric A. Collisson, Gordon B. Mills, Kenna R. Mills Shaw, Brad A. Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, and Joshua M. Stuart. The Cancer Genome Atlas pan-cancer analysis project. *Nature Genetics*, 45(10):1113–1120, 2013. doi: 10.1038/ng.2764. URL https://www.nature.com/articles/ng.2764.

Raúl Vicente, Michael Wibral, Michael Lindner, and Gordon Pipa. Transfer entropy—a model-free measure of effective connectivity for the neurosciences. *Journal of Computational Neuroscience*, 30(1):45–67, 2011. doi: 10.1007/s10827-010-0262-3. URL https://doi.org/10.1007/s10827-010-0262-3.

Sethu Vijayakumar, Aaron D'Souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634, 2005. doi: 10.1162/089976605774320557. URL https://doi.org/10.1162/089976605774320557.

Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 804–813, 2011.

A Estimator Details

Model configuration. We use $d = 128$, $K_c = 128$ and $K_r = 512$, with two axial blocks, eight attention heads and twelve self-attention blocks for table-memory refinement. Observation representations have width $4d = 512$; the token-level and memory-level feed-forward widths are 256 and 1024, respectively. The information decoder uses one cross-attention block followed by a GELU MLP with widths $512 \to 512 \to 1$. Dropout is zero.

Why classification yields information scores. For a fixed generating law, a balanced classifier distinguishing $P = p(x, y, z)$ from $Q = p(z)p(x \mid z)p(y \mid z)$ has optimal class probability $\eta^*(u) = P(u)/(P(u) + Q(u))$. Its log odds and their expectation are therefore

$$T^*(u) = \log \frac{\eta^*(u)}{1 - \eta^*(u)} = \log \frac{P(u)}{Q(u)}, \qquad \mathbb{E}_{u \sim P}[T^*(u)] = I(X; Y \mid Z). \tag{12}$$

This identity connects binary classification to pointwise information scores and identifies their joint expectation as CMI. Prompt-CMI extends this construction across tasks by conditioning the scorer on the observed context.

Context-conditioned scoring across tasks. The context supplies the distributional information used to adapt the scorer to each task. Let $\pi(\mathrm{d}\ell \mid C)$ denote the posterior over generating laws induced by the task distribution and context C. In the observation space before context-dependent preprocessing, the optimal unrestricted classifier under balanced class sampling has logit

$$T^*_{\mathrm{cls}}(u; C) = \log \frac{\int p_\ell(u)\, \pi(\mathrm{d}\ell \mid C)}{\int q_\ell(u)\, \pi(\mathrm{d}\ell \mid C)}. \tag{13}$$

The numerator and denominator are the joint and reference predictive densities given C. Their log ratio defines the context-conditioned classification target. Prompt-CMI jointly learns sample-level discrimination and calibrates the mean joint score to task-level CMI.

Calibration during pretraining. Calibration supervises the same mean joint score used for CMI estimation. For a task with target $I^\star_\ell$, we use

$$\mathcal{L}_{\mathrm{cal}}(\theta; \tau) = \mathrm{Huber}_\delta \Big[\mathrm{asinh}(\widehat{I}_\theta(U_P; C)/s) - \mathrm{asinh}(I^\star_\ell/s) \Big]. \tag{14}$$

The scale $s = 0.05$ nat balances errors across information levels, and $\delta = 1.5$ sets the Huber transition. We gradually increase the calibration weight λ during pretraining. All tasks contribute to classification, and tasks with known population CMI that satisfy the applicable support criteria also contribute to calibration. We optimize with AdamW using $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.01 and global gradient-norm clipping at 1.0.

Preprocessing and held-out inference. We fit the marginal transforms and regularized whitening on context observations, then apply them to both context and queries. Whitening acts separately within X, Y and Z, and the original rank-Gaussian coordinates remain as additional input channels. For each role, the context sample covariance Σ^r_C of rank-Gaussian coordinates uses denominator $n_C - 1$. ZCA whitening floors its eigenvalues at $\max(10^{-3}\|\Sigma^r_C\|_2, 10^{-6})$; when $\|\Sigma^r_C\|_2 \le 10^{-6}$, the whitened coordinates are set to zero. Each continuous cell concatenates four piecewise-linear features of its rank-Gaussian coordinate and four of its whitened coordinate. Both use knots $(-3, \Phi^{-1}(0.25), 0, \Phi^{-1}(0.75), 3)$ with linear tail extrapolation.

Categorical cells use eight context-support features: seen and unseen indicators, relative frequency, normalized log count, normalized smoothed surprisal, normalized log category count, a singleton indicator and inverse-square-root support. Category identities are represented through within-column equality groups with context-derived prototypes. Categories absent from the context have zero support and use a shared learned unseen-category state.

In four-fold inference, each observation serves once as a query, while the remaining folds form the context. All folds use the same scoring network and context-fitted preprocessing.

Memory and computational cost. The context encoder builds the reusable variable and table memories. Let n_C be the context size and $D = d_x + d_y + d_z$. At fixed depth and embedding width, its attention cost is

$$O\big(n_C D^2 + n_C D K_c + n_C K_r + K_r^2\big)\,. \tag{15}$$

These terms account for within-observation attention, variable summaries, table pooling and table-memory refinement, respectively. The variable and table memories contain $O(DK_c + K_r)$ vectors at fixed depth. Once these caches have been constructed, scoring n_Q further observations requires

$$O\big(n_Q(D^2 + DK_c + K_r)\big) \tag{16}$$

attention operations. Thus, repeated scoring reuses the context representation, with attention work linear in the number of additional observations. End-to-end timings include preprocessing and all network operations under the evaluation protocol.

Representative dataset-specific estimators. For continuous variables, the Frenzel–Pompe estimator finds the kth-neighbor radius in joint space, then counts neighbors within that radius in each subspace (Frenzel & Pompe, 2007):

$$\widehat{I}_{\mathrm{kNN}} = \psi(k) + \frac{1}{N}\sum_{i=1}^{N}[\psi(n_{Z,i}+1) - \psi(n_{XZ,i}+1) - \psi(n_{YZ,i}+1)]\,. \tag{17}$$

Here $n_{A,i}$ counts other observations strictly inside that radius in subspace A, using the maximum norm, and ψ is the digamma function.

The MI-difference variant of CCMI estimates conditional information through two classifier-based MI estimates (Mukherjee et al., 2020). For variables A, B, a classifier ω_ϕ distinguishes samples from P_{AB} and $P_A P_B$ using balanced binary cross-entropy:

$$\mathcal{L}_{\mathrm{BCE}}(\phi) = -\tfrac{1}{2}\mathbb{E}_{P_{AB}} \log \omega_\phi(A,B) - \tfrac{1}{2}\mathbb{E}_{P_A P_B} \log(1-\omega_\phi(A,B)). \tag{18}$$

With $g_\phi = \log[\omega_\phi/(1-\omega_\phi)]$, the Donsker–Varadhan readout on evaluation samples is

$$\widehat{I}_{\mathrm{clf}}(A;B) = \frac{1}{|\mathcal{J}|}\sum_{u\in\mathcal{J}} g_\phi(u) - \log\left[\frac{1}{|\mathcal{M}|}\sum_{v\in\mathcal{M}} e^{g_\phi(v)}\right], \tag{19}$$

where $\mathcal{J}$ contains joint pairs and $\mathcal{M}$ contains pairs drawn from the product of marginals. CCMI then uses

$$\widehat{I}_{\mathrm{CCMI}}(X;Y \mid Z) = \widehat{I}_{\mathrm{clf}}(X;(Y,Z)) - \widehat{I}_{\mathrm{clf}}(X;Z). \tag{20}$$

The reported estimate averages the repeated classifier fits specified by the evaluation protocol.

CMINE directly estimates the conditional density ratio by training a classifier ω_ϕ on joint samples B_P and approximate conditional-product samples B_Q:

$$\mathcal{L}_{\mathrm{CMINE}} = -\frac{\sum_{u\in B_P}\log\omega_\phi(u)}{2|B_P|} - \frac{\sum_{u\in B_Q}\log(1-\omega_\phi(u))}{2|B_Q|}. \tag{21}$$

After training, averaging the classifier's log odds over joint samples gives the CMI estimate. CMINE constructs its reference samples from the observed table. Prompt-CMI learns from reference queries supplied by the synthetic generator during pretraining.

B Synthetic Pretraining Tasks

Sampling a generating law. The generator constructs each task by specifying how dependence is created, conditioned, and expressed in the observations. It selects a compatible mechanism, role dimensions, variable types, and target information level. The dependence core defines the shared signal, while conditional effects and observation maps determine how it appears in the table. A law ℓ fixes these choices, including dependence parameters and signal coordinates or subspaces, before row-level sampling. The sampler draws $Z \sim p_\ell(z)$, generates latent variables $(U,V) \sim p_\ell(u,v \mid Z)$, and maps them to the observed roles X and Y. Context observations, joint queries, and reference queries all follow this law. Across tasks, the generator varies both the law and sample size, exposing Prompt-CMI to different distributions and amounts of statistical evidence. Context sizes and information targets are drawn from discrete grids spanning 48–10,000 observations and 0–8 nats, respectively. For each mechanism, role dimensions are drawn from supported configurations in the range 1–20.

Constructing latent dependence. Dependence cores control which signals are shared between the two roles and how much conditional information they carry. For a Gaussian core, let U_j and E_j be independent standard Gaussian variables, independent of Z, and define r active channels by

$$V_j = s(Z)\rho U_j + \sqrt{1-\rho^2}\, E_j, \qquad j = 1, \dots, r, \tag{22}$$

where $s(Z) \in \{-1, 1\}$ sets the dependence direction. The target information determines the channel correlation:

$$I(U; V \mid Z) = -\frac{r}{2}\log(1-\rho^2), \qquad \rho = \sqrt{1-\exp(-2I_\ell^\star/r)}. \tag{23}$$

For Gaussian and Student-t cores, we use $r = \min(8, d_x, d_y)$ active channels. Sparse signals occupy randomly selected coordinates; dense signal bases are obtained by orthonormalizing standard Gaussian matrices. Additional residual coordinates fill the remaining continuous dimensions. The Student-t core uses $\nu = 60$. Other cores use shared random scales, stable additive noise or finite-alphabet channels. These provide heavy-tailed dependence, noisy categorical interactions and rare-event signals within the same task construction.

Conditioning and observation maps. Conditional effects determine how a shared signal appears at different values of Z. Mechanisms use Z-dependent means, positive scales, dependence directions or regime-specific offsets. Observation maps then reshape the variables through monotone marginal transforms, triangular within-role maps or invertible geometric maps. Each mechanism uses its compatible combination of these operations. For maps applied separately to the two roles and invertible at each fixed z,

$$X = g_{\ell,X}(U, Z), \quad Y = g_{\ell,Y}(V, Z) \quad \Longrightarrow \quad I(X; Y \mid Z) = I(U; V \mid Z). \tag{24}$$

Here U and V include the residual coordinates. This separates the information carried by the latent dependence from the observed marginal shapes and geometry. For example, monotone transforms of a conditional Gaussian core produce non-Gaussian marginals while retaining its conditional copula and information target. In the Gaussian-core construction, conditional-mean coefficients are drawn from $\mathcal{N}(0, 0.35^2/m_z)$, where m_z is the number of active conditioning coordinates. For triangular maps, adjacent-coordinate coupling coefficients are sampled from $\mathrm{Uniform}(0.06, 0.18)$.

Categorical and mixed observations. Typed constructions extend the same process to tables containing categorical and continuous variables. Modular sums or XOR operations combine categorical inputs, with channel noise controlling their information and Z-dependent offsets supplying conditional structure. Binary signals can also be embedded in disjoint continuous intervals, so their values remain recoverable from the observations. Mixed tasks combine continuous and categorical signal blocks that are independent conditional on Z; their CMI contributions add. Rare-binary sources use prevalence $p = 0.05$. The additive mixed construction uses $q = 16$ categories and allocates $\min(0.25I_\ell^\star, 1)$ nats to its categorical component, with the remainder assigned to continuous blocks. Channel noise is determined by the corresponding entropy identities below. Additional coordinates and varied assignments of variable types expand the observed table while preserving the specified signal construction.

Joint and reference sampling. Each fixed law supplies independent context observations and joint observations for supervision. For a task with n_C context observations, we generate $\lceil n_C/3 \rceil$ joint queries and the same number of reference queries, giving balanced supervision of the two classes. To construct a reference query at a given z, we draw two independent conditional pairs $(x^{(a)}, y^{(a)})$ and $(x^{(b)}, y^{(b)})$ from $p_\ell(x, y \mid z)$ and form

$$u_P = (x^{(a)}, y^{(a)}, z), \qquad u_Q = (x^{(a)}, y^{(b)}, z). \tag{25}$$

Then $u_P \sim P_\ell$ and $u_Q \sim p_\ell(z)p_\ell(x \mid z)p_\ell(y \mid z)$. The symmetric construction exchanges X instead. The exchange occurs after all observation maps and replaces a complete role, preserving its internal dependence. Conditional-null tasks draw the roles independently given Z while retaining shared effects of Z; fully independent tasks also remove these shared effects. For a Student-t conditional null, the two roles receive independent conditional random scales.

Population information targets. Analytic cores supply the calibration target, and information-preserving observation maps carry it to the final table. In addition to the Gaussian identity above, the construction uses

$$\begin{aligned} I^\star_{\text{Student}} &= b_\nu(d_x, d_y, d_z) - \tfrac{r}{2}\log(1-\rho^2), \\ I^\star_{\text{Cauchy}} &= r\log(1+|a|), \\ I^\star_{q\text{-ary}} &= \log q - h_2(e) - e\log(q-1). \end{aligned} \tag{26}$$

For the shared-scale Student-t core, $b_\nu = h_\nu(d_x + d_z) + h_\nu(d_y + d_z) - h_\nu(d_z) - h_\nu(d_x + d_y + d_z)$, where $h_\nu(m)$ is the entropy of the m-dimensional Student-t distribution with ν degrees of freedom and identity scale. The correlation is chosen for a feasible target $I^\star_\ell \geq b_\nu$. The Cauchy expression follows from independent standard Cauchy variables with $V_j = aU_j + E_j$. The finite-alphabet expression uses a uniform q-ary source and a symmetric channel with error probability e, with $h_2(e) = -e\log e - (1-e)\log(1-e)$. For a binary source with prevalence p, the target is $h_2(p(1-e) + (1-p)e) - h_2(e)$.

The generator checks compatibility between the chosen mechanism, dimensions and target information. For categorical and mixed tasks, support checks cover the context, joint queries and independently sampled reference-source observations. Binary columns must contain both categories, and other categorical columns at least $\min(q, 8)$ distinct categories, where q is the column cardinality. All these tasks contribute to joint/reference classification, and those satisfying the category-coverage and local Z-support criteria also receive CMI calibration. The model receives observations, variable roles and types, while the population information target is supplied separately to the loss.

C Evaluation Protocols

Frozen inference and common inputs. Methods share the same input tables and evaluation repeats in each comparison. Prompt-CMI parameters remain frozen on test tables, while downstream predictors and dataset-specific baselines are trained according to their respective protocols.

Synthetic CMI mechanisms. Gaussian uses correlated Gaussian residuals with Z-dependent means; Student-t uses a shared-scale multivariate t core. Cauchy applies $\arctan$ transforms to additive Cauchy channels. Copula combines a conditional Gaussian copula with non-Gaussian marginals; Heterosk. introduces Z-dependent residual scales; Switch reverses correlation signs according to Z. Half-cube uses invertible, coupled $\tanh$ maps to produce bounded observations.

Synthetic CMI and aggregation. We evaluate CMI estimation on seven controlled synthetic mechanisms with known population CMI, varying sample size, role dimensions and information level. All methods receive the same sampled tables. Table 1 evaluates one fixed law per mechanism in each of two settings. For each of these 14 laws, we draw three independent tables with sampling seeds 0, 1 and 2, while keeping the estimator training seed fixed. Entries show the mean estimate and sample standard deviation (denominator $3-1$). Rankings use the mean absolute error across the three sampling seeds, computed before rounding. Tables 4–5 use the same fixed-distribution sampling protocol: each mechanism and setting has one fixed generating distribution and three independently sampled tables shared across methods. Their entries report mean and sample standard deviation across these sampling replicates, and rankings use the corresponding mean absolute errors before rounding. For Figure 4, we first average errors across three sampling seeds, then across configurations at each node, and finally give each mechanism equal weight. The dimension panel uses $N = 1024$, role dimensions $(d_x, d_y, d_z) = (m, m, 8)$ with $m \in \{4, 8, 12, 16, 20, 26\}$, and $I^\star \in \{0.8, 2\}$. The information panel uses role dimensions $(4, 4, 8)$ and $(20, 20, 8)$ and $I^\star \in \{0, 0.4, 0.8, 1.2, 1.6, 2, 3, 4, 6\}$. The cost panel uses the same 252 tables for every method: seven mechanisms, three sampling seeds, both role configurations, $I^\star = 0.8$, and $N \in \{256, 512, 1024, 2048, 5000, 8000\}$.

Neural baseline configurations. For Table 1, Figure 4, and Tables 4–5, neural baselines use a 50/25/25 train/validation/scoring split, with validation-based early stopping and restoration of the best weights. Baseline hyperparameters are set using independent development inputs. The kNN estimator uses $k = 5$.

CCMI and CMINE use three 500-unit ReLU hidden layers. CCMI estimates CMI through two MI terms, with refreshed training negatives and 100 classifier fits per estimate. CMINE uses conditional resampling with $k = 2$. DINE uses 16 mixture components and conditioning width 64, with a finite-sample correction to its latent Gaussian covariance readout.

The epoch caps for CCMI, CMINE, and DINE are 2,000, 4,000, and 2,000, with early-stopping patience 100, 600, and 100, respectively. For synthetic CI, DINE uses a joint shrinkage-covariance readout. CMI feature selection and order preservation use fixed baseline configurations.

Synthetic conditional dependence. For each of the three mechanisms, we use eight sample sizes, $N \in \{50, 100, 200, 400, 700, 1000, 2000, 5000\}$, and five independent groups of tables. At each sample size, every group contains 50 null tables with CMI 0 and 50 alternative tables with CMI 0.1 nat, giving 12,000 tables in total. Methods share the inputs within each group, including the prescribed pairing across sample sizes. We compute AUROC separately in each group and then average across the five groups. The green band in Figure 5 shows one between-group sample standard deviation, clipped to $[0, 1]$ for display.

We also compare with two dedicated tests, KCI (Zhang et al., 2011) and RCIT (Strobl et al., 2019). Our KCI variant uses median bandwidths and generalized cross-validation for regularization, retaining the full spectrum for the Gamma null moments. RCIT uses a sample-size-dependent number of conditional random features and stabilizes the residual covariance. For CMINE, we retain the Gaussian and SCM configurations and use an independently selected regularized configuration for the entire Cauchy curve. The CCMI and DINE configurations are fixed across groups and sample sizes. AUROC uses signed CMI scores and negative test p-values to summarize separation between null and alternative tables across score thresholds.

Order Preservation. For each setting and seed, each eligible observation receives a donor matched on Z alone: exact matching for categorical variables or standardized nearest-neighbor matching with a distance constraint for continuous variables. We fix a random observation order and replace a growing prefix with complete donor X vectors, keeping Y and Z fixed. We evaluate one clinical setting for NHANES, four representation–target combinations for Adult and 64 pathways for TCGA, each at 11 replacement fractions with three seeds. Within each dataset, curves average seeds within each setting and then weight settings equally. The inversion rate is the fraction of ordered replacement-level pairs for which the score at the higher replacement level is strictly greater. We compute this per setting and seed, then aggregate in the same order. This measures how consistently scores track the progression of dependence across replacement levels.

CMI Feature Selection. On Hydraulic, we predict flow from pressure-sensor groups conditional on temperature, using four temporal splits and three seeds per split. On SARCOS, we predict torque from velocity–acceleration groups conditional on joint position, using the official split and three seeds. Greedy forward selection scores each remaining group by its CMI with the target, conditional on Z and all previously selected groups. Selection uses training observations only. For each selector and budget, we train an identically configured MLP on the selected feature values together with Z. We divide each target's test RMSE by its training standard deviation, then average over targets, seeds and splits. The Z-only and all-input models use the same training and normalization rules.

Real-world conditional independence. Sachs, Chamber and Camera-Beta contain 11, 18 and 12 variables, giving 55, 153 and 66 unordered pairs, respectively. For each variable pair (X, Y), we test $H_0 : X \perp Y \mid Z$, where Z contains all remaining variables. The biological and physical reference DAGs are moralized by connecting parents that share a child and dropping edge directions, yielding 17, 72 and 35 reference edges, respectively. Each dataset uses five fixed splits, each with 512 scoring observations. India NSE and European financial institutions each contain 12 variables and 66 pairs per period. We evaluate both periods: 2019 and 2023 for India, and 2008 and 2018 for Europe. Their reference networks encode sector or institution membership, with 12 and 18 edges, respectively. We use five overlapping windows in each financial period, with 512 scoring observations per window. All splits, periods and windows contribute to the reported results.

For each variable pair, CMI methods share 199 resampled null tables constructed by replacing X using five-neighbor matching in Z under Chebyshev distance, with training-side scale normalization. The protocol permits self-matches and donor reuse. Prompt-CMI uses four-fold inference on the

512 scoring observations. Neural baselines fit on separate training and validation data and hold their weights fixed during scoring. For signed score T, the resampling p-value is

$$p = \frac{1 + \sum_{b=1}^{199} \mathbf{1}\{T^{(b)} \geq T^{\mathrm{obs}}\}}{200}. \tag{27}$$

KCI, RCIT and Fisher-Z (Kalisch & Bühlmann, 2007) use their native tests. Within each split or window, AUROC summarizes how well $1 - p$ separates reference edges from nonedges across score thresholds. Reference edges are positive labels, nonedges are negative labels, and tied positive–negative scores receive half credit. Biological and physical results average over five splits; financial results first average the five windows in each period, then give the two periods equal weight. Table 3 reports these means and the sample standard deviation across the five splits or ten financial windows.

Timing. Table 1 timings use the 14 seed-0 tables, separate from the three-seed accuracy summaries. Figure 4's cost panel evaluates accuracy on 252 tables and runtime on 84 representative tables spanning seven mechanisms, two role configurations, and six sample sizes. Each timed input is run three times. All methods use H100 80GB nodes with four CPU threads. Each call processes one table, synchronizes the GPU before and after timing, and refits trainable estimators. Timings include preprocessing, transfers, fitting, validation, weight restoration, and scoring, but exclude file I/O, startup, cold model loading, and pretraining. Order-preservation timings use 63 tables: seven settings, three seeds, and replacement fractions 0, 0.5, and 1, with three runs per table.

D Extended Synthetic CMI Results

Tables 4–5 extend Table 1 to eight settings at $N = 1024$, spanning role dimensions from $(4, 4, 8)$ to $(20, 20, 8)$ and true CMI from 0 to 4 nats. Each table varies one of these factors while holding the others fixed. Every setting retains all seven mechanisms and five methods. Each mechanism uses one fixed generating distribution per setting, with three independently sampled tables shared across methods, giving 280 estimates summarized from 840 individual evaluations on 168 input tables.

Table 4: **CMI estimates across dimensions.** Sample size is $N = 1024$ and true CMI is $I^\star = 0.8$ nat. Block headers give the sample size, role dimensions and true CMI. Cells show mean $\pm$ sample SD over three independent sampling replicates for each fixed generating distribution. Bold and underlined entries mark the lowest and second-lowest mean absolute errors across replicates, computed before rounding.

Method	Gaussian	Student-t	Cauchy	Copula	Heterosk.	Switch	Half-cube	t (s)
$N = 1024,\ (d_x, d_y, d_z) = (8, 8, 8),\ I^\star = 0.8$ nats								
Prompt-CMI	0.97±0.12	0.85±0.13	**1.06±0.01**	1.14±0.09	**0.89±0.02**	**0.88±0.07**	**0.86±0.07**	**0.13**
kNN	0.21±0.02	0.27±0.02	0.09±0.01	0.16±0.01	0.12±0.02	0.10±0.01	0.15±0.010	0.26
CCMI	0.51±0.04	0.47±0.05	0.10±0.02	0.39±0.04	0.32±0.05	-0.01±0.008	0.28±0.04	42.29
CMINE	0.62±0.48	**0.78±0.06**	0.06±0.07	0.69±0.28	0.52±0.20	0.03±0.01	0.22±0.22	8.91
DINE	**0.81±0.04**	0.80±0.08	0.27±0.04	**0.82±0.05**	0.71±0.10	0.02±0.03	0.51±0.05	8.05
$N = 1024,\ (d_x, d_y, d_z) = (12, 12, 8),\ I^\star = 0.8$ nats								
Prompt-CMI	1.06±0.17	1.03±0.05	**1.16±0.08**	1.05±0.05	0.62±0.05	**0.57±0.17**	**0.83±0.08**	**0.16**
kNN	0.15±0.006	0.28±0.03	0.06±0.03	0.13±0.01	0.06±0.002	0.05±0.03	0.13±0.03	0.32
CCMI	0.36±0.03	0.29±0.03	0.06±0.01	0.22±0.03	0.18±0.03	-0.08±0.02	0.15±0.03	42.19
CMINE	0.33±0.16	0.56±0.30	0.02±0.03	0.28±0.33	0.29±0.28	-0.02±0.03	0.26±0.19	8.91
DINE	**0.76±0.03**	**0.74±0.07**	0.28±0.03	**0.71±0.12**	**0.66±0.02**	0.02±0.03	0.61±0.15	8.24
$N = 1024,\ (d_x, d_y, d_z) = (16, 16, 8),\ I^\star = 0.8$ nats								
Prompt-CMI	0.66±0.06	**0.82±0.04**	**1.17±0.27**	0.94±0.06	0.58±0.07	0.01±0.0006	1.17±0.09	**0.20**
kNN	0.12±0.004	0.32±0.02	0.04±0.01	0.09±0.02	0.06±0.02	**0.03±0.02**	0.10±0.02	0.37
CCMI	0.20±0.02	0.16±0.03	0.03±0.01	0.07±0.03	0.00±0.03	-0.12±0.02	0.01±0.05	42.81
CMINE	0.36±0.16	0.17±0.12	0.03±0.04	0.35±0.26	0.17±0.10	-0.04±0.06	0.22±0.11	8.93
DINE	**0.79±0.05**	0.75±0.08	0.25±0.04	**0.73±0.11**	**0.71±0.02**	-0.02±0.05	**0.60±0.08**	8.25
$N = 1024,\ (d_x, d_y, d_z) = (20, 20, 8),\ I^\star = 0.8$ nats								
Prompt-CMI	**0.71±0.07**	**0.81±0.07**	**1.28±0.19**	1.21±0.14	0.96±0.14	0.03±0.03	1.17±0.27	**0.23**
kNN	0.10±0.02	0.36±0.006	0.03±0.04	0.08±0.006	0.10±0.03	**0.04±0.03**	0.11±0.01	0.42
CCMI	0.06±0.03	0.04±0.01	0.01±0.006	-0.07±0.02	-0.08±0.10	-0.17±0.02	-0.18±0.06	44.24
CMINE	0.06±0.07	0.24±0.09	0.01±0.03	0.34±0.14	0.02±0.07	-0.01±0.02	0.04±0.15	9.46
DINE	0.72±0.11	0.77±0.11	0.23±0.05	**0.74±0.09**	**0.67±0.09**	-0.02±0.03	**0.64±0.11**	8.46

t is mean runtime per dataset in seconds, averaged over seven mechanisms with three runs per input. Timing uses the first sampled table for each mechanism on H100 GPUs with four CPU threads and includes fitting when required.

Table 5: **CMI estimates across information levels.** Sample size is $N = 1024$ and role dimensions are $(d_x, d_y, d_z) = (4, 4, 8)$. Block headers give the sample size, role dimensions and true CMI. Cells show mean ± sample SD over three independent sampling replicates for each fixed generating distribution. Bold and underlined entries mark the lowest and second-lowest mean absolute errors across replicates, computed before rounding.

Method	Gaussian	Student-t	Cauchy	Copula	Heterosk.	Switch	Half-cube	t (s)
$N = 1024,\ (d_x, d_y, d_z) = (4, 4, 8),\ I^\star = 0$ nats								
Prompt-CMI	**0.00±0.0003**	**0.00±0.003**	**0.00±0.003**	**0.00±0.0010**	**0.00±0.001**	**0.00±0.003**	**0.00±0.001**	**0.13**
kNN	0.01±0.01	0.00±0.006	-0.02±0.009	0.00±0.009	0.01±0.006	-0.01±0.01	0.00±0.004	0.20
CCMI	-0.02±0.004	-0.01±0.009	0.00±0.003	-0.01±0.01	-0.04±0.03	-0.01±0.003	-0.03±0.006	41.28
CMINE	-0.02±0.002	0.01±0.04	-0.02±0.03	-0.02±0.02	0.01±0.03	0.03±0.01	0.01±0.04	8.95
DINE	0.00±0.005	0.00±0.008	-0.01±0.003	0.00±0.005	0.00±0.007	0.00±0.007	0.00±0.003	8.04
$N = 1024,\ (d_x, d_y, d_z) = (4, 4, 8),\ I^\star = 1.2$ nats								
Prompt-CMI	**1.23±0.04**	**1.20±0.07**	**1.29±0.09**	0.93±0.03	**1.23±0.07**	**1.59±0.05**	**1.10±0.05**	**0.10**
kNN	0.43±0.01	0.41±0.03	0.21±0.02	0.27±0.02	0.37±0.01	0.25±0.03	0.34±0.02	0.19
CCMI	0.87±0.04	0.89±0.12	0.26±0.010	0.83±0.008	0.91±0.02	0.22±0.06	0.67±0.03	43.34
CMINE	1.04±0.29	1.18±0.20	0.17±0.05	1.07±0.25	1.22±0.21	0.43±0.23	0.72±0.35	9.00
DINE	1.07±0.06	1.11±0.12	0.37±0.02	**1.08±0.21**	1.16±0.06	0.01±0.008	0.86±0.04	8.01
$N = 1024,\ (d_x, d_y, d_z) = (4, 4, 8),\ I^\star = 2$ nats								
Prompt-CMI	**2.04±0.06**	**2.00±0.12**	**2.12±0.07**	1.53±0.03	**1.86±0.10**	**2.49±0.15**	**1.89±0.06**	**0.10**
kNN	0.66±0.01	0.59±0.02	0.40±0.02	0.40±0.02	0.51±0.01	0.35±0.02	0.48±0.01	0.19
CCMI	1.47±0.07	1.54±0.11	0.55±0.04	1.43±0.07	1.44±0.11	0.62±0.01	1.07±0.04	44.24
CMINE	1.91±0.05	2.06±0.42	0.70±0.13	2.20±0.22	1.55±0.43	1.51±0.19	1.43±0.39	9.06
DINE	1.86±0.09	1.75±0.10	0.68±0.04	**1.84±0.19**	1.79±0.06	0.02±0.03	1.40±0.07	8.03
$N = 1024,\ (d_x, d_y, d_z) = (4, 4, 8),\ I^\star = 4$ nats								
Prompt-CMI	**3.87±0.11**	3.48±0.10	**3.85±0.23**	3.00±0.08	3.28±0.04	4.76±0.03	**3.45±0.06**	**0.10**
kNN	1.00±0.01	0.91±0.010	0.78±0.02	0.57±0.01	0.74±0.02	0.62±0.009	0.74±0.03	0.18
CCMI	3.18±0.05	3.22±0.009	1.11±0.009	2.70±0.16	2.59±0.24	1.90±0.21	1.92±0.06	47.52
CMINE	4.27±0.89	**4.43±0.22**	1.22±0.22	**3.30±0.29**	3.51±0.70	**3.41±0.18**	2.24±0.38	9.08
DINE	3.75±0.26	3.35±0.16	1.33±0.17	3.23±0.16	**3.52±0.06**	0.02±0.03	2.54±0.15	8.22

t is mean runtime per dataset in seconds, averaged over seven mechanisms with three runs per input. Timing uses the first sampled table for each mechanism on H100 GPUs with four CPU threads and includes fitting when required.