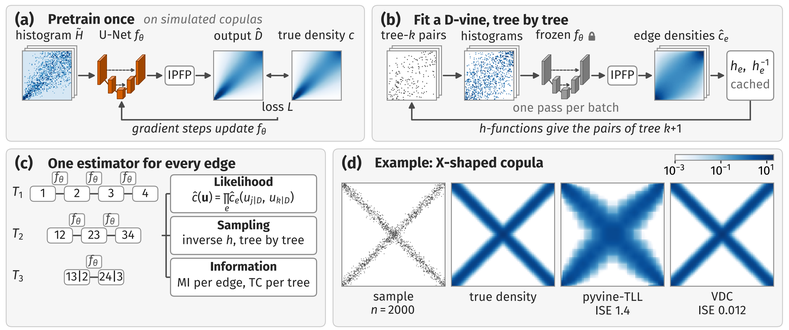
Amortized Vine Copulas for High-Dimensional Density and Information Estimation

Houman Safaai | Kempner Institute for the Study of Natural and Artificial Intelligence, Harvard University | houman_safaai@harvard.edu

1 VDC at a glance: one pretrained network estimates every pair-copula of a vine



(a, b) outputs of the deployed checkpoint on illustrative samples; (d) measured on a fresh sample (n = 2,000) from the X-shaped pattern of the training zoo. ISE: integrated squared error.

In brief

Problem. A vine on d variables needs $d(d-1)/2$ bivariate fits, one per edge; parametric fits risk misspecification and nonparametric ones are slow.

Idea. Vine Denoising Copula (VDC): train one denoising network on simulated copulas, then reuse it, frozen, at every edge.

Per edge. Histogram → one batched network pass → IPFP projection onto copula densities → cached h-functions for the next tree.

Result. An explicit vine with likelihood, sampling, and mutual information and total correlation by edge and tree, fitted faster at high dimension once pretrained.

Limits. A finite grid blurs near-deterministic pairs, and accuracy varies by dataset.

2 Vine copulas and the per-edge bottleneck

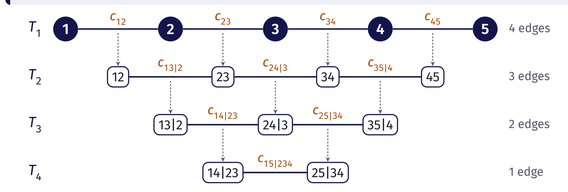
Sklar's theorem [1] splits a continuous random vector into its marginals and its copula C, a joint CDF with uniform marginals:

$$F(x_1,\dots,x_d) = C\big(F_1(x_1),\dots,F_d(x_d)\big)$$
$$f(\mathbf{x}) = c(u_1,\dots,u_d)\prod_{i=1}^{d} f_i(x_i), \qquad u_i = F_i(x_i)$$

A regular vine [2, 3] factorizes the copula density into $d(d-1)/2$ bivariate pair-copulas on trees $T_1,\dots,T_{d-1}$:

$$c(u_1,\dots,u_d) = \prod_{\ell=1}^{d-1}\prod_{(j,k|D)\in T_\ell} c_{j,k|D}\big(u_{j|D},\, u_{k|D}\big)$$

d = 5: $c = c_{12}\,c_{23}\,c_{34}\,c_{45}\cdot c_{13|2}\,c_{24|3}\,c_{35|4}\cdot c_{14|23}\,c_{25|34}\cdot c_{15|234}$



D-vine, d = 5: each edge is one pair-copula; dashed arrows are h-functions.

h-functions supply the conditional arguments tree by tree (simplified vine: $c_{j,k|D}$ does not depend on the value of u_D):

$$h_{U|V}(u \mid v) = \int_0^u c_{UV}(s,v)\,ds = \mathbb{P}(U \le u \mid V = v)$$
$$u_{j|D\cup\{k\}} = h_{j|k;D}\big(u_{j|D} \mid u_{k|D}\big)$$

The bottleneck: one bivariate fit per edge

d = 100 variables give 4,950 edges and d = 500 give 124,750; classical pipelines fit each separately [4, 5]:

Pair-copula estimator	Flexible	Time / edge
Parametric family selection	only in its library	591 ms
Transformation local likelihood (pyvine-TLL)	✓	16 ms
Neural copula, retrained per edge	✓	~ 10^3 s
VDC: one pretrained network	✓	**6 ms**

Per-fit times of the bivariate benchmark (VDC: one H100 GPU; pyvine: one CPU thread); neural copula: ACNet, retrained for each pair.

3 Method: denoise, project, recurse

1 Histogram of each edge's pseudo-observations

$$u_i = \frac{r_i^{(x)}}{n+1}, \quad v_i = \frac{r_i^{(y)}}{n+1}, \qquad H_{ab} = \frac{1}{n\Delta^2}\sum_{i=1}^{n}\mathbf{1}\big[(u_i,v_i)\in B_{ab}\big]$$

ranks r_i; $m \times m$ cells B_{ab} of width $\Delta = 1/m$, m = 64

2 One network pass predicts a density grid

$$Z = f_\theta\big(\tilde{H},\ \text{coords},\ \log n;\ t\big), \quad D_{\text{raw}} = e^{\text{clip}(Z,-20,20)}, \quad \hat{D} = \Pi_{\text{IPFP}}(D_{\text{raw}})$$
$$\tilde{H} = (1-\alpha)H + \alpha H_{\text{unif}}, \quad \alpha = 0.65\,t, \quad t \sim \text{Unif}(0,1) \quad \text{(training; test: } \tilde{H} = H,\ t = 0)$$

f_θ: a 2-D U-Net, FiLM-conditioned on t, with probit edge channels and a log n channel, trained once on simulated copulas (Gaussian, Student-t, Clayton, Gumbel, Frank, Joe, BB1, BB7, rotations, mixtures and complex shapes).

3 IPFP projects onto copula densities

$$Q = \frac{D_{\text{raw}}}{\Delta^2\sum_{a,b} D_{ab}^{\text{raw}}}, \qquad D_{ab} \leftarrow D_{ab}\,\frac{1/m}{\mu_a^{\text{row}}}, \quad \mu_a^{\text{row}} = \Delta^2\sum_b D_{ab}$$

iterative proportional fitting (IPFP): rows as above, then columns; K = 30 sweeps (15 in the scaling and S&P100 runs)

Proposition [6, 7]. For strictly positive Q, IPFP converges to the unique $\hat{D}_\infty = \arg\min_{P\in\Pi} \mathrm{KL}(P \,\|\, Q)$ over the grids Π with row and column masses $1/m$, a valid grid copula. For the cell-averaged true copula $c^* \in \Pi$,

$$\mathrm{KL}(c^* \,\|\, Q) = \mathrm{KL}(c^* \,\|\, \hat{D}_\infty) + \mathrm{KL}(\hat{D}_\infty \,\|\, Q),$$

so the exact projection never increases the KL error on the grid.

4 Training loss on the projected output

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{ISE}}\mathcal{L}_{\text{ISE}} + \lambda_{\text{tail}}\mathcal{L}_{\text{tail}} + \lambda_{\text{marg}}\mathcal{L}_{\text{marg}}, \qquad \mathcal{L}_{\text{CE}} = -\sum_{i,j} T_{ij}\,\Delta^2\log\big(\hat{D}_{ij}\Delta^2\big)$$

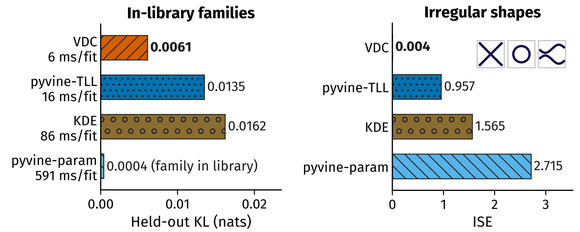
T: true density grid; $(\lambda_{\text{ISE}}, \lambda_{\text{tail}}, \lambda_{\text{marg}}) = (0,\ 0.2,\ 0.02)$

5 Batched, cached vine assembly

Tree by tree: histogram every edge, run the frozen f_θ in batches of up to 256 edges (one network pass per batch), apply IPFP, and cache the h-tables (cumulative sums of $\hat{D}$), which give the next tree's pseudo-observations $u_{j|D\cup\{k\}} = \hat{h}(u_{j|D} \mid u_{k|D})$. Histograms cost $O(n)$; network and projection costs do not depend on n.

4 Density estimation with one frozen network

Bivariate copulas (n = 2,000 per fit)



Left: 9 in-library cases × 5 fresh samples; time per fit (VDC: one H100; pyvine: one CPU thread). Right: X, ring and double-banana patterns, used for model selection: a diagnostic, not an independent test.

- Best flexible estimator in-library: 2–3× lower ISE, KL, τ and tail errors than pyvine-TLL, at a few milliseconds per fit.
- Parametric selection wins only when the true family is in its library, and does worst on irregular shapes.
- Unseen families (BB6, BB8, Tawn, Plackett): same likelihood as TLL refits and about half the MI error.

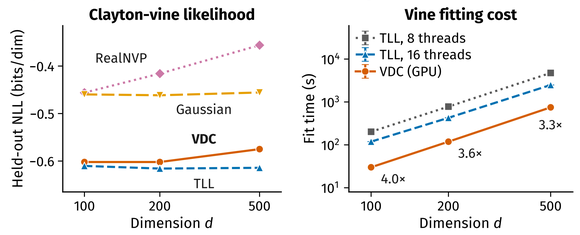
Full vines on UCI (MAF splits [8], 5,000 training rows)

Data (d)	VDC	TLL	param	RealNVP	Gauss.	Fit (s)
POWER (6)	−0.1413	**−0.1822**	−0.1078	−0.1221	−0.0685	0.56 / 0.97
GAS (8)	−1.4773	**−1.7050**	−1.4040	−1.0991	−1.2472	0.21 / 1.97
HEPMASS (21)	**−0.2372**	−0.2009	−0.1628	0.2538	−0.1039	1.17 / 9.66
MINIBOONE (43)	**−0.9689**	−0.7444	−0.9334	−0.4145	−0.8853	4.49 / 35.72

Held-out copula NLL (bits/dim); fit time VDC / TLL; parametric selection: 10–503 s; RealNVP: fixed training budget. With 50,000 rows TLL also beats VDC on HEPMASS, and RealNVP is best on all four.

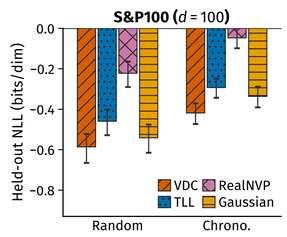
5 Scaling to hundreds of dimensions

Clayton D-vines, d = 100–500 (20,000 training rows)



On one GPU, VDC fits these vines **3.3–4× faster** than pyvine-TLL on 16 CPU threads, saving **about 29 minutes per fit** at d = 500, for a small likelihood cost (0.039 bits/dim). The 9.6 h pretraining is paid once.

S&P100 daily returns (d = 100, 1,356 training days)



VDC has the best held-out likelihood on both splits (bars: 95% CIs; RealNVP: small training budget).

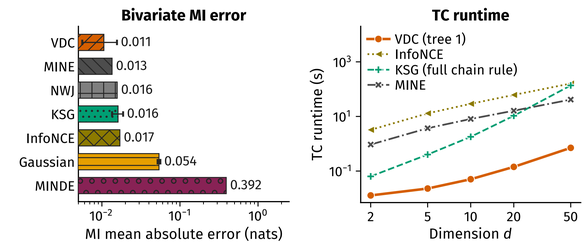
Chronological split: 0.126 bits/dim better than TLL on average, and better on 87% of test days.

Sampling: the fitted vine reproduces the held-out Kendall-τ matrix far better than a TLL refit (RMSE 0.052 vs. 0.192).

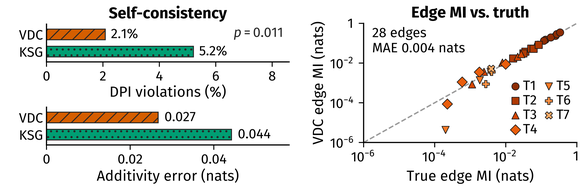
6 Information estimation from the vine

The fitted vine gives mutual information (MI) and total correlation (TC):

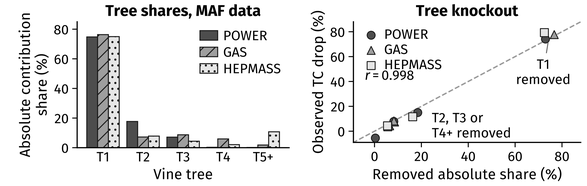
$$I(X;Y) = \mathbb{E}\big[\log c(U,V)\big] = -H(c), \qquad \hat{I}(X;Y) = \Delta^2\sum_{a,b}\hat{D}_{ab}\log\hat{D}_{ab}$$
$$\mathrm{TC}(X_1,\dots,X_d) = \mathbb{E}\big[\log c(U_1,\dots,U_d)\big] = \sum_{e\in\mathcal{E}}\mathbb{E}\big[\log c_e(U_e,V_e)\big]$$



Lowest bivariate MI error of 7 estimators. Total correlation of a 50-dimensional Gaussian AR(1) (first vine tree): **0.71 s** against 41–161 s for KSG [9] and neural estimators. **With the same first-tree structure, KSG is about as accurate but 4.7× slower** (one CPU thread vs. one GPU).



Fewer data-processing-inequality violations than KSG (**10 vs. 25** of 160 datasets, p = 0.011; the panel shows per-trial rates) and lower additivity error. Edge MI matches the analytic values of a Gaussian-factor vine (MAE 0.004 nats).



MAF data: tree 1 carries about three quarters of the absolute edge contributions; the TC drop after knocking out a tree group tracks its share (Pearson r = 0.998).

7 Best uses, limitations and resources

Where VDC helps most

- Many pairs, little data per pair (S&P100: best held-out likelihood).
- Repeated fits: rolling windows, bootstraps, many datasets.
- Interpretable MI and TC by edge and tree, plus sampling.
- Irregular shapes that parametric families cannot express.

Limitations

- Finite grid: near-deterministic pairs and fine tails are hard.
- With ample data per pair, TLL or flows can be more accurate.
- Simplified vines; imputation and tail risk need calibration.
- Depends on the training data; retrained VDC-64/128 improve most benchmarks.

Paper arXiv:2604.20568 | **Code and docs** GitHub repository | **Main checkpoint** Hugging Face

[1] Sklar 1959. [2] Bedford & Cooke 2002. [3] Aas et al. 2009. [4] Dißmann et al. 2013. [5] Nagler & Vatter 2025. [6] Sinkhorn & Knopp 1967. [7] Csiszár 1975. [8] Papamakarios et al. 2017. [9] Kraskov et al. 2004.

This work has been made possible in part by a gift from the Chan Zuckerberg Initiative Foundation to establish the Kempner Institute for the Study of Natural and Artificial Intelligence at Harvard University.