VARIABLE ELIMINATION NETWORKS: LEARNING GAUSSIAN POTENTIALS FOR GRAPH-WIDE CONDITIONAL INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

In many networked systems, measurements are available at only a subset of nodes, while the states of different nodes remain interdependent across the graph. Therefore, accurate recovery of the unobserved states requires sparse local evidence to be integrated with dependencies that may span long graph distances. Message-passing neural networks access distant nodes through repeated local updates, tying long-range conditioning to propagation depth. We introduce the Variable Elimination Network (VEN), a graph neural architecture that learns local Gaussian potentials and composes them through exact differentiable Gaussian elimination. Shared node and edge maps construct a positive-definite precision $Q_\theta(G, a)$, while a sample-dependent source map constructs $b_\theta(x, G)$; optional measurements enter as Gaussian likelihood factors, and Schur, Cholesky, or Woodbury backends return the joint posterior mean and covariance. We establish positive definiteness and permutation equivariance, prove all-pairs source sensitivity for connected scalar systems with positive couplings, and show equivalence of the three inference backends. Across six public water, power, gas, EEG, and air-quality state-recovery tasks, five-seed experiments show that VEN improves over the strongest evaluated baseline on every task. On the GasLib benchmark, VEN reduces NRMSE by approximately 12.1% relative to a pressure-conditioned hydraulic MAP estimator, by 85.6% relative to ridge, and by at least 89.5% relative to the learned graph baselines. Controlled size extrapolation and component ablations associate the gains with graph-wide conditioning, observations, and learned coupling. Cached inference supports repeated sparse conditional queries at throughput comparable to shallow GNNs in the evaluated moderate-size systems. Additional large sparse-field, citation-graph, and task-matched DGMRF comparisons delineate the regime in which reusable, low-fill Gaussian graph systems are effective. Code is available at https://anonymous.4open.science/r/VEN.

1 INTRODUCTION

Many physical and sensor networks expose only a sparse, changing subset of their state. Estimating the unobserved variables then requires more than local interpolation: evidence at one node must be reconciled with couplings and balance relations throughout the graph. This structure appears in power-system state estimation, gas and water pressure recovery, spatial sensing, and electrophysiology.

Message-passing neural networks (MPNNs) update each node from a bounded neighborhood (Kipf & Welling, 2017; Hamilton et al., 2017; Xu et al., 2019). An L-layer MPNN therefore transports information through at most L graph hops unless it augments the topology or uses a global operator. Deeper propagation, personalized diffusion, graph Transformers, and implicit GNNs broaden this range (Klicpera et al., 2019; Ying et al., 2021; Rampasek et al., 2022; Gu et al., 2020; Bamberger et al., 2025). For tasks whose target is the conditional mean of a coupled system, these architectures learn a representation-based approximation to the solution map rather than explicitly conditioning a joint state model.

Probabilistic graphical models provide the complementary computation. A Gaussian Markov random field (GMRF) represents local conditional dependencies by a sparse precision matrix and answers a

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.

conditional query through elimination or linear algebra (Rue & Held, 2005; Davis, 2006). Its global semantics and uncertainty are attractive, but a classical GMRF does not by itself specify how graph attributes and high-dimensional sample features should produce task-adapted local factors under end-to-end supervision.

We combine these ideas in a neural architecture. A *Variable Elimination Network* learns positive local node and edge potentials, assembles them into one joint Gaussian energy, inserts any observed node states as likelihood factors, and uses exact differentiable elimination as its graph-wide operator. The central computation is

$$E_\theta(z; x, G) = \tfrac{1}{2} z^\mathsf{T} Q_\theta(G, a) z - b_\theta(x, G)^\mathsf{T} z, \qquad p_\theta(z \mid x, G) = \mathcal{N}(Q^{-1} b, Q^{-1}), \tag{1}$$

where $z_i \in \mathbb{R}^d$ is a node state. Static graph attributes determine the precision, whereas sample features determine the information vector. Sparse measurements update the same joint model. Thus evidence can affect every compatible state in one inference operation rather than after a prescribed number of learned propagation layers.

The architecture separates the learned model from its numerical realization. Direct Schur elimination exposes the elimination program; Cholesky provides a stable factorization backend; and Woodbury reuses a fixed-topology factorization when observation sets change across queries. Given the same learned factors and observations, all three compute the same posterior. This separation lets the deployment backend follow graph structure without changing the network's semantics.

We evaluate the architecture in continuous state recovery on reusable, low-fill systems. On controlled fixed-interaction chains and grids, its error remains low when test graphs are larger than training graphs. On six core public state-recovery tasks, it improves over the strongest evaluated baseline in every task, with the largest margin on the tree-structured GasLib-134 system; a 437-station measured extension provides a separate scale-candidate result under a finite-entry pilot mask. Ordinary citation tasks and high-fill factorization define the empirical and computational boundaries of this regime.

Our contributions are:

- We define VEN, a graph neural architecture whose layer learns local Gaussian potentials and composes them through exact variable elimination, returning a joint conditional distribution rather than only node embeddings.
- We establish well-posedness, permutation equivariance, global sensitivity for connected scalar systems, and equivalence of Schur, Cholesky, and Woodbury inference.
- We provide a five-seed evaluation spanning controlled size extrapolation and six mechanism-aligned public tasks, plus a 437-station measured scale extension, together with external physical residuals and component ablations of observations, coupling, and learned precision.
- We characterize the practical regime through a unified cold/hot/memory benchmark and explicit fill-in and caching limitations.

2 Background and Related Work

Local and global graph neural computation. GCN, GraphSAGE, GIN, and GINE represent the canonical local aggregation family (Kipf & Welling, 2017; Hamilton et al., 2017; Xu et al., 2019; Hu et al., 2020). APPNP separates prediction from a personalized PageRank propagation (Klicpera et al., 2019). Graph Transformers introduce global pairwise interaction, often with structural encodings (Ying et al., 2021; Rampasek et al., 2022; Deng et al., 2024), while diffusion and bundle-based models define graph-global evolution operators (Chamberlain et al., 2021; Bamberger et al., 2025). Implicit GNNs obtain representations as fixed points and use implicit differentiation (Gu et al., 2020). These methods enlarge receptive fields or change propagation dynamics. VEN differs at the output semantics and atomic operator: its state is a joint random variable, observations are likelihood factors, and the layer executes Gaussian marginalization or conditioning.

Graphical models and differentiable solvers. GMRFs encode conditional independence through sparse precision matrices, and sparse Cholesky or Schur complements support exact inference (Rue & Held, 2005; Davis, 2006). The Deep Gaussian Markov Random Field (DGMRF) learns multilayer

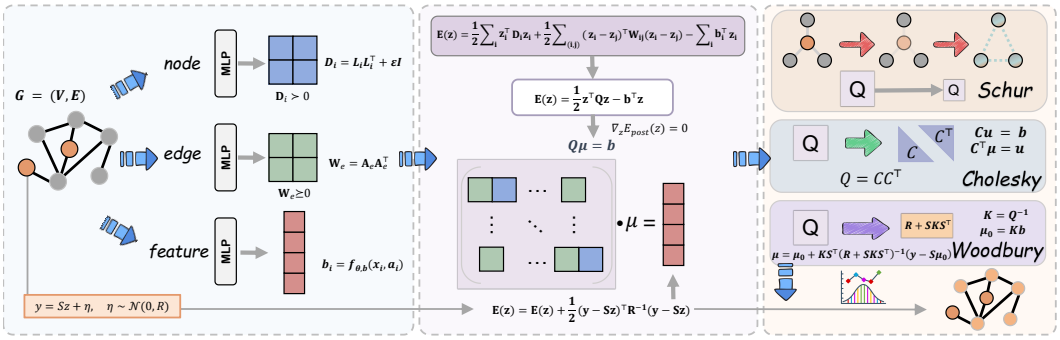


Figure 1: Core inference pipeline of VEN. MLPs construct local Gaussian potentials for nodes and edges, capturing node precision, pairwise coupling, and source terms. These potentials form a global Gaussian energy, which is updated with partial observations. Differentiating the energy yields a linear system solved by Schur elimination, Cholesky factorization, or Woodbury correction. Schur elimination reduces the graph by removing variables and inducing fill-in edges, enabling inference over all node states and their uncertainty.

precision operators on arbitrary graphs and uses variational training with conjugate Gaussian inference (Oskarsson et al., 2022). Its native protocol fits one scalar field under a fixed observation mask; VEN focuses on block-valued states and repeated queries with changing evidence. Gaussian belief propagation offers an iterative alternative under convergence conditions (Weiss & Freeman, 2001). SPDE constructions derive sparse GMRF precisions from discretized differential operators, while INLA and selected inversion provide complementary posterior-computation toolchains (Lindgren et al., 2011; Rue et al., 2009; Lin et al., 2011). Learned Gaussian CRFs predict quadratic energies for structured outputs such as denoising and segmentation (Vemulapalli et al., 2016a;b); sparse and low-rank conditional GGMs estimate related conditional precision structure through regularized statistical objectives (Frot et al., 2019). VEN combines shared local neural maps, likelihood-factor observations, and interchangeable exact Gaussian backends in one attributed-graph interface. Differentiable optimization layers provide the broader mechanism for training through such solves (Amos & Kolter, 2017; Agrawal et al., 2019).

Graph state estimation. Graph networks have been used to approximate physical simulators and power flow, including MeshGraphNets and PowerFlowNet (Pfaff et al., 2021; Lin et al., 2024). Their learned messages are effective when local updates approximate the target operator. VEN targets a complementary regime in which local couplings can be reused across many sparse conditional queries. We therefore compare against canonical MPNNs, diffusion, analytic estimators, and available task-specific models under shared data splits.

3 Variable Elimination Networks

3.1 Problem Setup

Let $G = (V, E)$ be an attributed graph with $n = |V|$ nodes, static node and edge attributes a, and sample features x. The model assigns a continuous state $z_i \in \mathbb{R}^d$ to each node and stacks all states as $z \in \mathbb{R}^D$, $D = nd$. A query may additionally contain observations $y \in \mathbb{R}^k$ of linear state coordinates selected by $S \in \mathbb{R}^{k \times D}$ with noise covariance $R \succ 0$. The goal is to produce a state estimate and, when requested, a conditional distribution over all coordinates.

One VEN forward pass has four stages: (i) learn positive node and edge factors from static graph attributes, (ii) learn the information vector from the current sample, (iii) add the observation likelihood, and (iv) eliminate the joint Gaussian to obtain its posterior. Figure 1 follows these stages.

3.2 LEARNING A POSITIVE GRAPH PRECISION

For node i, a shared map produces a positive-definite block

$$D_i = L_i L_i^\mathsf{T} + \varepsilon I_d, \qquad L_i = f_{\theta,D}(a_i), \tag{2}$$

where diagonal entries of L_i use a positive parameterization. For edge $e = (i, j)$, a second shared map produces a positive-semidefinite coupling

$$W_e = A_e A_e^\mathsf{T}, \qquad A_e = f_{\theta,W}(a_i, a_j, a_e). \tag{3}$$

Let $B_e \in \mathbb{R}^{d\times D}$ satisfy $B_e z = z_i - z_j$. The global precision is

$$Q_\theta(G, a) = \mathrm{blkdiag}(D_1, \ldots, D_n) + \sum_{e\in E} B_e^\mathsf{T} W_e B_e. \tag{4}$$

Equation (4) is a matrix-weighted grounded Laplacian: one edge contributes the block $\left[\begin{smallmatrix} W_e & -W_e \\ -W_e & W_e \end{smallmatrix}\right]$. This disagreement-potential bias guarantees positive-semidefinite edge contributions and enables the scalar sensitivity result below. The supported pairwise class is positive, symmetric difference coupling; signed and asymmetric interactions require a broader parameterization. Grounded Laplacian constructions are included by fixing a reference state and applying Eq. (4) to the remaining coordinates. Scalar state recovery is the special case $d = 1$; multi-channel power states use $d > 1$ with full coupling blocks.

A source network maps the current sample to local natural parameters,

$$b_\theta(x, G) = \mathrm{stack}_{i\in V}\, f_{\theta,b}(x_i, a_i), \tag{5}$$

possibly after a task-specific local encoder. The base distribution is therefore

$$p_\theta(z \mid x, G) = \mathcal{N}(\mu_0, K), \qquad \mu_0 = Q^{-1}b, \quad K = Q^{-1}. \tag{6}$$

For a fixed graph and static attributes, Q is shared across samples; current values and masks affect b and the observation factors. This separation is what makes factorization reuse possible.

3.3 OBSERVATIONS AS LIKELIHOOD FACTORS

We model sparse evidence by

$$y = Sz + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, R). \tag{7}$$

Multiplying Eq. (6) by this likelihood gives

$$Q_{\mathrm{post}} = Q + S^\mathsf{T} R^{-1} S, \tag{8}$$

$$\mu_{\mathrm{post}} = Q_{\mathrm{post}}^{-1}\big(b + S^\mathsf{T} R^{-1} y\big), \tag{9}$$

$$\Sigma_{\mathrm{post}} = Q_{\mathrm{post}}^{-1}. \tag{10}$$

When no evidence is available, S has zero rows and these equations reduce to Eq. (6). Observation noise is a functional part of the architecture: replacing R changes both how strongly measurements alter the state and the posterior uncertainty.

For fixed Q, the matrix inversion lemma yields the evidence-space form

$$\mu_{\mathrm{post}} = \mu_0 + KS^\mathsf{T}(R + SKS^\mathsf{T})^{-1}(y - S\mu_0), \tag{11}$$

$$\Sigma_{\mathrm{post}} = K - KS^\mathsf{T}(R + SKS^\mathsf{T})^{-1}SK. \tag{12}$$

Only the $k \times k$ evidence system changes with a sparse observation set.

3.4 EXACT ELIMINATION BACKENDS

Partition the state into eliminated coordinates E and retained coordinates K. Eliminating z_E produces the Schur update

$$Q' = Q_{KK} - Q_{KE}Q_{EE}^{-1}Q_{EK}, \tag{13}$$

$$b' = b_K - Q_{KE}Q_{EE}^{-1}b_E. \tag{14}$$

Repeated application followed by back substitution is direct variable elimination. Cholesky applies the algebra to a factorized global matrix, and Eqs. (11)–(12) reuse the base inverse for changing observations. All operations are differentiable; training losses may be placed on the posterior mean, marginal log likelihood, or a downstream task head.

Algorithm 1 A VEN conditional query

Require: Graph (G, a), sample features x, optional evidence (S, y, R)
1: Construct D_i and W_e with Eqs. (2)–(3)
2: Assemble Q by Eq. (4); compute $b = b_\theta(x, G)$
3: **if** a compatible base cache exists **then**
4: Apply Eqs. (11)–(12)
5: **else**
6: Form Eqs. (8)–(10) and solve by Schur or Cholesky
7: **return** posterior mean μ_{post} and requested covariance statistics

3.5 Complexity and cache regime

Let f_t be the scalar frontal dimension at elimination event t. A direct dense realization costs $O(\sum_t f_t^3)$ time and $O(\sum_t f_t^2)$ stored frontal entries; in the worst case these become $O(D^3)$ and $O(D^2)$. Thus fill-in, rather than node count alone, controls exact sparse elimination. For bounded induced width w and fixed block size d, the standard bound is $O(n(wd)^3)$ time and $O(n(wd)^2)$ factor storage.

The evaluated cached implementation materializes $K = Q^{-1}$, using $O(D^2)$ memory. A query with k observations costs a dense source multiplication plus cross-covariance and evidence solves, $O(D^2 + Dk + k^3)$ per sample before batching. This cache targets moderate D, $k \ll D$, and repeated use of one Q. Sparse triangular solves or selected inverse columns extend the same equations to larger systems with lower cache memory.

4 Properties

Well-posedness. **Proposition 1.** If every $D_i \succ 0$ and every $W_e \succeq 0$, then Q in Eq. (4) is positive definite. If $R \succ 0$, then $Q_{\text{post}} \succ 0$ for every observation selector S.

The result follows because $z^\mathsf{T} Q z = \sum_i z_i^\mathsf{T} D_i z_i + \sum_e (B_e z)^\mathsf{T} W_e (B_e z) > 0$ for nonzero z. The likelihood adds a positive-semidefinite term. The posterior is therefore unique and finite for every query.

Permutation equivariance. **Proposition 2.** Suppose the local maps are shared and edge aggregation is order invariant. For a node permutation P, consistently permuting graph attributes, state coordinates, and observations gives $Q' = PQP^\mathsf{T}$, $b' = Pb$, $S' = SP^\mathsf{T}$, and $\mu'_{\text{post}} = P\mu_{\text{post}}$, $\Sigma'_{\text{post}} = P\Sigma_{\text{post}}P^\mathsf{T}$.

This property holds independently of the exact backend because each backend computes the same posterior map.

Graph-wide dependence. **Proposition 3.** In the scalar connected case with $D_i = m_i > 0$ and positive edge conductances $W_e = w_e > 0$, Q^{-1} is elementwise positive. Consequently, every source coordinate has nonzero influence on every posterior-prior state coordinate before observation cancellation.

Here $Q = \text{diag}(m) + L_w$ is an irreducible nonsingular M-matrix, whose inverse is strictly positive. This formalizes a direct graph-wide path for the scalar architecture; the block model retains global algebraic dependence but not an elementwise sign guarantee.

Strict positivity is an existence statement, not a distance-independent lower bound. The magnitude can decay rapidly with graph distance and depends on coupling, diagonal grounding, topology, and observation noise. We measure both the controlled sensitivity and task-level consequence in Figure 4; the detailed sweep is in Appendix J.

Backend equivalence. **Proposition 4.** Direct Schur elimination, Cholesky solution of Eqs. (8)–(9), and the Woodbury identities in Eqs. (11)–(12) return identical posterior moments in exact arithmetic.

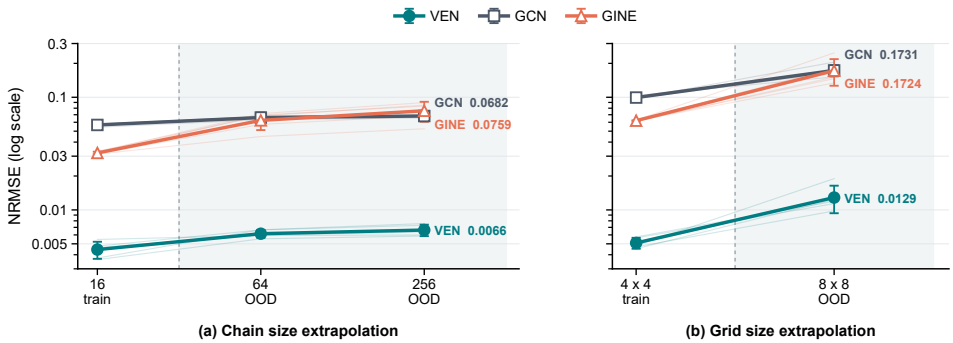


Figure 2: Fixed-interaction size extrapolation. **(a)** Models train on 16-node chains and are evaluated up to 256 nodes. **(b)** Models train on 4×4 grids and are evaluated on 8×8 grids. Thin paths show the five training runs; solid markers and error bars show mean NRMSE and sample standard deviation. The shaded region marks graph sizes unseen during training. VEN preserves substantially lower error as graph diameter and size increase. Complete two- and three-layer GCN/GINE and APPNP results appear in Appendix C.

Appendix A provides proofs. Across the implemented float32 regression tests, the largest absolute output difference among backends is 9.54×10^{-7}.

5 Experiments

We ask five questions. **Q1:** Does graph-wide elimination extrapolate beyond the graph sizes seen in training? **Q2:** Does VEN improve real or standard-simulation state recovery when evidence is sparse? **Q3:** Which gains depend on observations, graph coupling, and learned precision? **Q4:** Does graph-wide dependence retain useful magnitude as evidence becomes distant? **Q5:** What preparation, hot-query, and memory costs accompany exact inference?

5.1 Protocol

All trainable models use the same five run seeds, $\{7, 42, 123, 2002, 2027\}$. Dataset generation and fixed splits use seed 7; repeated runs change initialization, minibatch order, and training randomness. We report mean $\pm$ sample standard deviation. Baselines receive the same node features, observations, masks, topology, splits, and target normalization. Neural baselines include two- or three-layer GCN/GINE, GraphSAGE, GATv2, and APPNP where appropriate; task baselines include ridge, spatial or hydraulic interpolation, PowerFlowNet, and Polynormer. DGMRF is the closest probabilistic graph baseline: it retains the official three-layer DGMRF precision operator and uses one shared field model across training snapshots, with parameters frozen before validation/test conditioning. Each trainable model is selected by its own validation metric. Dataset construction, search spaces, and official-code adaptations are detailed in Appendix B.

5.2 Controlled size extrapolation

We generate targets by solving a fixed local SPD system while allowing only the information vector $b_\theta(x, G)$ to vary across samples. Models train on length-16 chains and 4×4 grids, then test on chains up to length 256 and 8×8 grids. All methods use hidden width 48 and the same optimization limit; GCN and GINE are evaluated at two and three layers.

Figure 2 answers Q1 within the controlled regime. At 256 chain nodes, VEN reaches 0.00662 ± 0.00077 NRMSE, compared with 0.06817 ± 0.00153 for GCN-3 and 0.07588 ± 0.01529 for GINE-3. On the unseen 8×8 grid, the corresponding values are 0.01288 ± 0.00353, 0.17311 ± 0.01969, and 0.17236 ± 0.04577. These curves are consistent with solving a size-compatible global operator rather than transporting evidence through fixed depth. They do not establish extrapolation for arbitrary graph families.

5.3 CONDITIONAL STATE RECOVERY

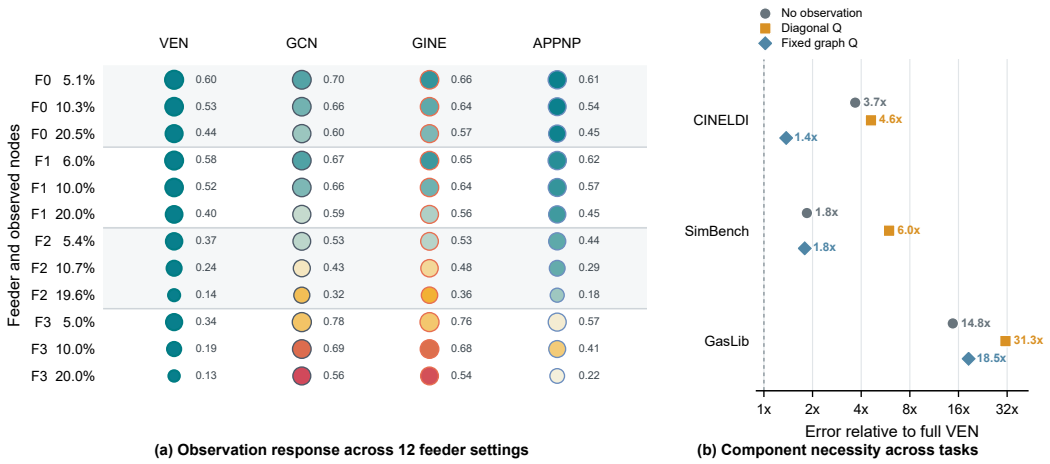
Table 1 summarizes seven tasks aligned with the model's conditional Gaussian semantics. LeakDB uses 31 water junctions and eight changing pressure sensors (KIOS Research and Innovation Center of Excellence, 2021). SimBench uses a 97-node low-voltage network with 512 chronological AC snapshots (Meinecke et al., 2020). CINELDI covers four 39–80-node Norwegian feeders and three PMU densities. GasLib-134 is a 134-node, 133-edge gas tree with 12 changing pressure sensors and 1,221 converged nomination states. EEGBCI reconstructs 48 of 64 electrodes from 16 observations across a subject holdout. AQI-36 reconstructs missing $PM_{2.5}$ values on 36 Beijing stations. Urban Air full437 extends the same measured air-quality source to 437 stations and 8,760 hourly snapshots; because the public file exposes an official evaluation mask only for the original 36 stations, we use a fixed finite-entry pilot mask and label this column as a scale candidate.

Table 1: Conditional state recovery. Lower is better. Values are mean ± sample standard deviation over five training runs; deterministic estimators with a fixed protocol have no deviation. The AQI and full437 Ridge controls use the same seed-specific training masks as the neural runs and therefore report their run variation. Superscripts S and M distinguish simulator-derived or benchmark-generated labels from measured targets. CINELDI reports the simple mean over 12 feeder–PMU settings, each evaluated with five runs. EEGBCI is RMSE in μV; AQI and full437 are $PM_{2.5}$ RMSE; all other columns are NRMSE. The full437 column uses a finite-entry pilot mask over 437 measured stations. A dash denotes a baseline not defined for that task or not covered under the complete table protocol.

Method	LeakDB^{S}	SimBench^{S}	CINELDI^{S}	GasLib-134^{S}	EEGBCI^{M}	AQI-36^{M}	full437^{M}
Ridge	1.4572	0.1913	.4482	0.3239	67.870	49.616±.035	31.9909±.1282
Interpolation	1.2068	–	–	–	62.547	23.891	28.7940
Physical estimator	–	0.9180	.7378	0.0529	–	–	–
GCN	1.1986±.0073	.7722±.0177	.5999±.0143	1.2440±.0285	63.385±.096	33.230±.744	26.1596±.0236
GINE	1.2115±.0194	.7096±.0223	.5884±.0106	1.2435±.0653	63.487±.206	32.159±1.528	25.6825±.0871
APPNP	1.1985±.0076	.6040±.0079	.4462±.0043	.8580±.0650	63.400±.073	61.373±4.357	56.5432±.3211
Polynormer	–	.4581±.0419	–	.4445±.0539	–	30.420±1.692	27.2580±.1784
PowerFlowNet	–	.5182±.0077	–	–	–	–	–
DGMRF	1.4033±.0682	.7098±.0434	.6496±.1143	.5700±.0815	63.225±.026	23.839±.020	26.6613±.5218
VEN	1.0660±.0133	.1786±.0007	.3742±.0028	.0465±.0012	61.302±.611	22.968±.103	25.5751±.0240

VEN is strongest in all seven columns among the evaluated methods, although the effect size is modest on measured EEG and full437. On full437, the five-seed VEN RMSE is 25.5751 ± 0.0240, versus 25.6825 ± 0.0871 for GINE, 26.1596 ± 0.0236 for GCN, and 27.2580 ± 0.1784 for Polynormer; Ridge and coordinate IDW reach 31.9909 ± 0.1282 and 28.7940, respectively. The full437 result uses the finite-entry pilot mask described above and is therefore a scale-candidate result rather than an official 437-station benchmark. Completing the previously absent same-task controls does not change the ordering on the original AQI task: global Ridge reaches 0.4482 CINELDI NRMSE (macro-averaged over all 12 settings), while AQI Ridge and APPNP reach 49.616 ± 0.035 and 61.373 ± 4.357 $PM_{2.5}$ RMSE, respectively, compared with 22.968 ± 0.103 for VEN. The task-matched Polynormer result is also available on AQI (30.420 ± 1.692), but remains behind VEN. DGMRF is the closest learned Gaussian comparator and remains behind VEN on every shared-snapshot task. On GasLib, the pressure-conditioned hydraulic MAP estimator narrows the comparison to 0.0529 versus 0.0465 ± 0.0012 NRMSE, testing whether learned correction remains useful after supplying the simulator. On CINELDI, VEN has the lowest five-seed mean in all 12 feeder–density settings; the deterministic nonlinear WLS reference obtains 0.7378 macro NRMSE. Recomputing the SimBench predictions with independent nonlinear AC equations gives an injection RMSE of 0.00447 ± 0.00014 MVA for VEN, versus 0.4293 ± 0.0618 for GCN and 0.4335 ± 0.0272 for GINE; ridge reaches 0.000351 MVA in this nearly linear fixed-topology regime. Appendix C reports the complete physical audit, paired snapshot distributions, bootstrap intervals, and input-parity checks.

The table is intentionally task-matched rather than a rectangular sweep of unrelated methods. GCN, GINE, APPNP, Ridge, and DGMRF have broad task coverage, while full437 reports the protocol-preserving baselines available for this larger measured extension. Polynormer is reported where its public core has a protocol-preserving adapter and five-seed coverage; PowerFlowNet is an AC power-flow model and is therefore restricted to SimBench. Interpolation and physical estimators are listed only when a canonical coordinate or forward-equation definition exists. GraphSAGE, GATv2, FGNN, and the official FISF feature-imputation core were additionally evaluated on the tasks for



(a) Observation response across 12 feeder settings **(b) Component necessity across tasks**

Figure 3: Observation use and component necessity. **(a)** CINELDI voltage-state recovery across 12 feeder–density settings. Each circle reports the five-run mean NRMSE; fill color encodes error relative to the best method within that setting, area encodes absolute NRMSE, and the outline identifies the method. Alternating bands group the three observation densities of each feeder. **(b)** Error inflation after removing the observation likelihood, edge coupling, or learned structural precision, normalized by the matched full VEN result and shown on a base-2 logarithmic axis.

which their input protocol is defined; their partial-coverage results are retained in the artifact audit and are not represented as missing cells in the main comparison.

For the full437 scale candidate, DGMRF obtains 26.6613 ± 0.5218 RMSE under the official one-layer unweighted option; this is below VEN and GINE but above GCN and Polynormer. The entry uses the shared-field adapter and the frozen finite-entry pilot mask, so it is interpreted as a protocol-matched scale result rather than an official native DGMRF benchmark.

5.4 Mechanism Ablations

Figure 3 tests the observation interface and removes one mathematical component at a time. Across all four CINELDI feeders, VEN improves as the observed fraction grows and remains below the evaluated message-passing baselines. *No observation* deletes the likelihood in Eq. (7); *diagonal* Q removes all edge blocks from Eq. (4); and *fixed graph* Q replaces learned coupling by a unit graph operator (a fixed physical operator on CINELDI). Every removal increases error in CINELDI, SimBench, and GasLib, so observations, cross-edge coupling, and learned structural precision account for distinct gains, answering Q3.

5.5 Evidence Distance and Online Updates

Proposition 3 guarantees a nonzero scalar path, but it does not say whether that path remains numerically useful. We therefore evaluate the normalized observation Jacobian $|\partial \mu_i^+ / \partial y_j| / |\partial \mu_j^+ / \partial y_j|$ on chains, balanced trees, and narrow grids generated by $Q = \alpha I + wL$, and group real-task errors by the distance from each hidden node to its nearest sensor. Figure 4(c) maps the useful chain range across coupling and grounding at $N = 256$ and $R = .1$, while panel (d) gives the five-seed SimBench result. All 243 combinations of topology, $N \in \{64, 128, 256\}$, $w \in \{.25, 1, 4\}$, $\alpha \in \{.01, .1, 1\}$, and $R \in \{.01, .1, 1\}$ have strictly positive influence, yet the useful range at normalized influence 10^{-3} varies from 3 to 138 chain hops. Thus all-pairs dependence is guaranteed in this scalar setting, while its magnitude is controlled by the graph and potential parameters.

The task-level result is stronger than positivity alone. On SimBench, VEN NRMSE rises smoothly from .1570 at one hop to .2320 at 13 or more hops; the best evaluated three-layer GNN reaches .7068 in the last bin. GasLib shows the same pattern, from .0258 to .0614 for VEN versus at least 1.1579

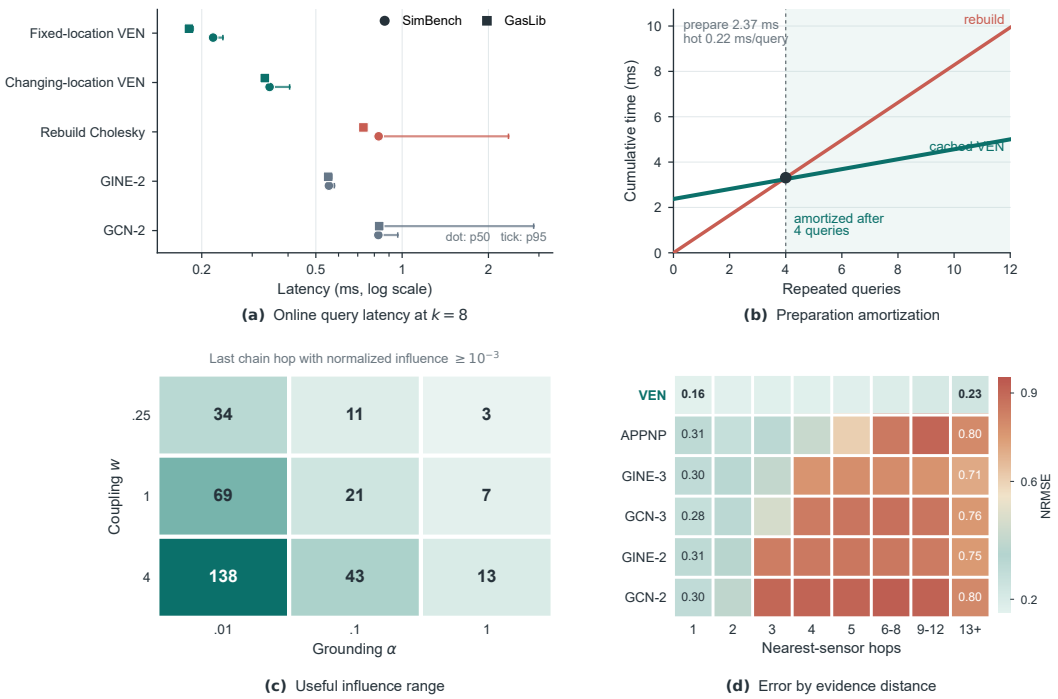


Figure 4: Global influence and repeated evidence updates. **(a)** Batch-one latency at $k = 8$ on SimBench and GasLib; dots mark p50 and terminal ticks mark p95 over 1,000 queries. Fixed-location VEN caches the selector gain, whereas changing-location VEN recomputes it. **(b)** SimBench preparation plus cumulative query time at $k = 8$; the cache amortizes after four queries against rebuilding Cholesky. **(c)** Last chain hop whose normalized influence remains at least 10^{-3} across coupling w and grounding α ($N = 256$, $R = .1$). **(d)** Five-seed SimBench NRMSE by nearest-sensor distance; endpoint values are printed in each row. Gaussian timings use float64 on one NVIDIA L40 after 20 warm-up queries; GCN/GINE use float32 and serve as application-latency references.

for the two- and three-layer GCN/GINE models. These bins answer Q4 for the evaluated low-fill systems; distance-independent recovery lies outside the supported claim.

Figure 4(a,b) separately tests the deployment mechanism. For 1,000 batch-one queries on one L40, fixed sensor locations allow VEN to cache the evidence gain and answer in about .18 ms on both SimBench and GasLib. Changing the locations costs .33–.41 ms. Against rebuilding the posterior Cholesky each time, the measured preparation amortizes after 3–6 queries over $k \in \{2, 4, 8, 16, 32\}$ sensors; Gaussian backend errors remain below 1.14×10^{-13}. Appendix J reports all three query streams, p50/p95 latency, memory, and timing boundaries.

Because VEN returns Σ_{post}, Appendix E reports marginal NLL, CRPS, and coverage; marginal-NLL training improves SimBench NLL from .139 to $-.520$ at a 0.9% NRMSE cost, while cross-subject and chronological shifts remain calibration boundaries. Appendix F gives the native DGMRF and large-field results, and Appendix I gives timing, memory, and backend diagnostics.

6 Conclusion

We introduced VEN, which learns positive local Gaussian potentials and composes them by exact variable elimination. Its posterior is well posed and permutation equivariant, while Schur, Cholesky, and Woodbury provide equivalent inference backends. Controlled extrapolation, six public recovery tasks, a 437-station measured extension, ablations, and online queries support the method on reusable low-fill systems. The scope is set by fill-in, cache size, shared precision, and calibration under distribution shift (Appendices I, H, and G).

REFERENCES

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, 2019.

Brandon Amos and J. Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, 2017.

Jacob Bamberger, Federico Barbero, Xiaowen Dong, and Michael M. Bronstein. Bundle neural networks for message diffusion on graphs. In *International Conference on Learning Representations*, 2025.

Benjamin P. Chamberlain, James Rowbottom, Maria I. Gorinova, Michael M. Bronstein, Stefan Webb, and Emanuele Rossi. Graph neural networks inspired by classical iterative algorithms. In *International Conference on Machine Learning*, 2021.

Timothy A. Davis. *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2006.

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In *International Conference on Learning Representations*, 2024.

Lill Mari Engan, Stine Ekrheim, Sigurd Bjarghov, Jonatan Ralf Axel Klemets, Ivan Schytte, and Gerd Kjølle. Reference dataset for semi-urban and rural norwegian low voltage distribution grids. *Data in Brief*, 59:111453, 2025. doi: 10.1016/j.dib.2025.111453.

Benjamin Frot, Luke Jostins, and Gilean McVean. Graphical model selection for gaussian conditional random fields in the presence of latent variables. *Journal of the American Statistical Association*, 114(526):723–734, 2019. doi: 10.1080/01621459.2018.1434531.

Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. In *Advances in Neural Information Processing Systems*, 2020.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 2017.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020.

KIOS Research and Innovation Center of Excellence. Leakdb: Leakage diagnosis benchmark datasets. `https://github.com/KIOS-Research/LeakDB`, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan G"unnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

Lin Lin, Chao Yang, Juan C. Meza, Jianfeng Lu, Lexing Ying, and Weinan E. Selinv—an algorithm for selected inversion of a sparse symmetric matrix. *ACM Transactions on Mathematical Software*, 37(4):1–19, 2011. doi: 10.1145/1916461.1916464.

Nan Lin, Stavros Orfanoudakis, Nathan Ordonez Cardenas, Juan S. Giraldo, and Pedro P. Vergara. Powerflownet: Power flow approximation using message passing graph neural networks. *arXiv preprint arXiv:2311.03415*, 2024.

Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between gaussian fields and gaussian markov random fields: The stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B*, 73(4):423–498, 2011. doi: 10.1111/j.1467-9868.2011.00777.x.

Steffen Meinecke, Džanan Sarajlić, Simon R. Drauz, Andreas Klettke, Lars-Peter Lauven, Christian Rehtanz, Albert Moser, and Martin Braun. Simbench—a benchmark dataset of electric power systems to compare innovative solutions based on power flow analysis. *Energies*, 13(12):3290, 2020. doi: 10.3390/en13123290.

Joel Oskarsson, Per Sidén, and Fredrik Lindsten. Scalable deep gaussian markov random fields for general graphs. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17117–17137. PMLR, 2022. URL https://proceedings.mlr.press/v162/oskarsson22a.html.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Ladislav Rampasek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, 2022.

Haavard Rue and Leonhard Held. *Gaussian Markov Random Fields: Theory and Applications*. Chapman and Hall/CRC, 2005.

Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the Royal Statistical Society: Series B*, 71(2):319–392, 2009. doi: 10.1111/j.1467-9868.2008.00700.x.

Gerwin Schalk, Dennis J. McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R. Wolpaw. Bci2000: A general-purpose brain-computer interface (bci) system. *IEEE Transactions on Biomedical Engineering*, 51(6):1034–1043, 2004. doi: 10.1109/TBME.2004.827072.

Martin Schmidt, Denis Aßmann, Robert Burlacu, Jesco Humpola, Imke Joormann, Nikolaos Kanelakis, Thorsten Koch, Djamal Oucherif, Marc E. Pfetsch, Lars Schewe, Robert Schwarz, and Mathias Sirvent. Gaslib—a library of gas network instances. *Data*, 2(4):40, 2017. doi: 10.3390/data2040040.

Raviteja Vemulapalli, Oncel Tuzel, and Ming-Yu Liu. Deep gaussian conditional random field network: A model-based deep network for discriminative denoising. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016a. doi: 10.1109/CVPR.2016.519.

Raviteja Vemulapalli, Oncel Tuzel, Ming-Yu Liu, and Rama Chellappa. Gaussian conditional random field network for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016b. doi: 10.1109/CVPR.2016.351.

Yair Weiss and William T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Xiuwen Yi, Yu Zheng, Junbo Zhang, and Tianrui Li. St-mvl: Filling missing values in geo-sensory time series data. In *International Joint Conference on Artificial Intelligence*, pp. 2704–2710, 2016.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? In *Advances in Neural Information Processing Systems*, 2021.

A Proofs

A.1 Proof of Proposition 1

For any nonzero z,

$$z^\mathsf{T} Q z = \sum_i z_i^\mathsf{T} D_i z_i + \sum_e (B_e z)^\mathsf{T} W_e (B_e z).$$

The first sum is strictly positive because each $D_i \succ 0$, while the second is nonnegative. Hence $Q \succ 0$. Furthermore, $u^\mathsf{T} S^\mathsf{T} R^{-1} S u = (Su)^\mathsf{T} R^{-1} (Su) \geq 0$, so adding the likelihood term preserves positive definiteness.

A.2 Proof of Proposition 2

Let P denote the block permutation induced by relabeling nodes. Shared local maps produce the same blocks attached to relabeled nodes and edges, hence $Q' = PQP^\mathsf{T}$ and $b' = Pb$. A consistently relabeled observation operator is $S' = SP^\mathsf{T}$. Therefore

$$Q'_{\text{post}} = PQ_{\text{post}}P^\mathsf{T}, \qquad h'_{\text{post}} = P(b + S^\mathsf{T} R^{-1} y).$$

Using $(PAP^\mathsf{T})^{-1} = PA^{-1}P^\mathsf{T}$ for a permutation matrix gives $\mu'_{\text{post}} = P\mu_{\text{post}}$ and $\Sigma'_{\text{post}} = P\Sigma_{\text{post}}P^\mathsf{T}$.

A.3 Proof of Proposition 3

The scalar matrix $Q = \operatorname{diag}(m) + L_w$ has positive diagonal, nonpositive off-diagonal entries, and is strictly positive definite. Positive conductances on a connected graph make it irreducible. It is therefore an irreducible nonsingular M-matrix, whose inverse is strictly positive elementwise. Since $\partial \mu_0 / \partial b = Q^{-1}$, every source coordinate influences every state coordinate.

A.4 Proof of Proposition 4

Completing the square in z_E and integrating it yields Eqs. (13)–(14); repeated elimination is block Gaussian factorization. Cholesky solves the same normal equations $Q_{\text{post}}\mu_{\text{post}} = b + S^\mathsf{T} R^{-1} y$. Applying the matrix inversion lemma to $Q_{\text{post}} = Q + S^\mathsf{T} R^{-1} S$ yields Eq. (12); multiplying the posterior natural parameter and simplifying gives Eq. (11). Uniqueness from Proposition 1 completes the equivalence.

B Dataset and Experimental Protocol

Table 2: Primary datasets and task structure. Labels marked "simulated" are generated from public topology and inputs by the named physical simulator; they are not field measurements.

Dataset	Nodes	Edges	Samples	Observed state	Split	Label source
LeakDB Hanoi	31	33	temporal series	8 pressures	chronological	hydraulic benchmark
SimBench LV rural2	97	–	512	8 PMUs	60/20/20 time	pandapower AC (simulated)
CINELDI LV	39–80	radial	4×512	5/10/20% PMUs	60/20/20 time	real loads + AC (simulated)
GasLib-134	134	133	1,221	12 pressures	chronological	nominations + pandapipes (simulated)
EEGBCI	64	spatial	subject windows	16 electrodes	subject 8/9/10	measured EEG
AQI-36	36	spatial	8,759 hours	varying stations	official time masks	measured $PM_{2.5}$
Urban Air full437	437	1,158 kNN	8,760 hours	finite-entry mask	chronological	measured $PM_{2.5}$

LeakDB. We use the Hanoi pressure-recovery protocol with 31 estimated junctions and eight randomly varying pressure sensors. Metrics are computed on the remaining 23 nodes. Hydraulic interpolation, ridge, GCN, GraphSAGE, GATv2, GINE, APPNP, and a factor-graph network receive identical inputs.

SimBench. The 1-LV-rural2–0-sw network supplies time-varying load and generation profiles; pandapower produces voltage magnitude and angle labels. Eight PMUs observe both channels with noise. The target excludes observed coordinates and the slack node. The official cores of Polynormer and PowerFlowNet are connected to the shared task input/output adapter without changing their internal computation.

CINELDI. We use the four public Norwegian low-voltage reference feeders with 39, 50, 56, and 80 nodes (Engan et al., 2025). Each provides 512 chronological snapshots driven by public consumer profiles. PMU counts approximate 5, 10, and 20% of each feeder; tests score unobserved non-reference states. The 12-setting aggregate is descriptive; every setting contains five independent training runs.

GasLib-134. GasLib provides network parameters and nominations rather than unique official pressure labels (Schmidt et al., 2017). We generate 1,221 converged operating states under one documented pandapipes policy. Each sample uses noisy boundary flow features and 12 changing pressure sensors, and scores 122 unobserved nodes. Consequently, the result is a standard simulation on a public engineering instance rather than field pressure validation.

Table 3: GasLib-134 input parity. Every method receives the same split, normalization, node evidence, and topology. "Used" distinguishes inputs accepted by the model family from inputs available to the adapter; no hidden pressure or test statistic is supplied.

Input	Exact definition	VEN	GINE	GCN/APPNP	Polynormer
Dynamic node	Noisy normalized boundary flow; nonzero-flow indicator	Used	Used	Used	Used
Static node	Normalized (x, y); normalized pressure bounds; source/sink/internal one-hot	Used	Used	Used	Used
Pressure evidence	Normalized noisy squared pressure at 12 nodes; binary observation mask	Likelihood	Channels	Channels	Channels
Topology	Same 134 nodes and 133 undirected connections	Used	Used	Used	Used
Continuous edge	Normalized length, diameter, roughness, heat transfer	Used	Used	–	–
Edge type	Pipe, short pipe, compressor, control-valve one-hot	Used	Used	–	–

GCN and APPNP do not define edge-feature arguments, and the official Polynormer core consumes topology but not edge attributes; their adapters therefore cannot use the final two rows without changing model computation. All four still receive exactly the same dynamic, static, and observed-pressure node channels. VEN uses static node and edge attributes only for Q, while its source map receives all node channels.

EEGBCI and AQI. EEGBCI uses measured 64-channel recordings from PhysioNet (Schalk et al., 2004); training, validation, and test use subjects 8, 9, and 10, respectively. AQI-36 follows the public $PM_{2.5}$ missing-value protocol on 36 Beijing stations (Yi et al., 2016). Urban Air full437 uses the same public measured source extended to 437 stations; because its public evaluation mask covers only the original 36 stations, we hold out 20% of finite entries per timestamp using a fixed seed-7 mask and report this as a scale-candidate protocol. All tasks keep their public temporal order and score only hidden coordinates.

Optimization. Every formal result uses seeds 7, 42, 123, 2002, and 2027 with fixed data seed 7. Hyperparameters and stopping times are selected from validation data independently for each architecture; the test split is not used for model selection. The local factor and source maps are one-hidden-layer SiLU MLPs with the widths in Table 4; their final outputs parameterize scalar factors or the required Cholesky entries. Softplus transforms and diagonal floors (10^{-4}, or 10^{-3} for EEGBCI) enforce positivity. AdamW uses the listed weight decay and gradients are clipped to norm 5. Table 4 records the selected VEN configurations for the primary tasks. Baseline searches use the same split, stopping rule, and maximum epoch budget. All reported variation is the sample standard deviation over independent training runs, not repeated evaluation of one checkpoint.

Table 4: Selected VEN training configurations for the primary tasks. "Minimum" is the earliest epoch at which validation-based early stopping is allowed.

Dataset	Hidden	Depth	Batch	LR	WD	Max.	Min.	Patience	Observations
LeakDB Hanoi	32	2	512	.003	10^{-4}	120	–	20	8 pressure sensors
SimBench	32	2	64	.003	10^{-4}	300	–	30	8 PMUs
CINELDI	32	2	32	.001	10^{-4}	1,000	200	100	5/10/20% PMUs
GasLib-134	32	2	64	.003	10^{-4}	1,000	200	100	12 pressure sensors
EEGBCI	32	2	512	.003	10^{-4}	100	–	15	16 electrodes
AQI-36	32	2	256	.003	10^{-4}	1,000	200	100	25% masked stations
Urban Air full437	32	2	256	.003	10^{-4}	1,000	200	100	20% finite-entry mask

The CINELDI search uses learning rate .003 for the graph-neural baselines and .001 for VEN; other rows use .003 for all neural models after validation selection. The machine-readable records retain the realized stopping epoch, parameter count, checkpoint, and per-seed metrics. Runtime measurements use PyTorch float32 on an NVIDIA L40 and synchronize CUDA at every timing boundary.

Table 5: GasLib-134 validation diagnostics (mean ± sample standard deviation, five runs). All models use the same chronological split and train-only target normalization. Best epoch is selected only by validation NRMSE. Polynormer uses its official core and repository training phases.

Model	Parameters	Best epoch	Validation NRMSE	Test NRMSE
VEN	964	608.8±273.6	.0421±.0011	.0465±.0012
GCN	3,713	28.2±14.8	.7626±.0094	1.2440±.0285
GINE	10,563	33.4±27.2	.7857±.0385	1.2435±.0653
APPNP	1,473	414.8±50.8	.3801±.0175	.8580±.0650
Polynormer	11,426	162.0±42.6	.2293±.0303	.4445±.0539

GCN and GINE attain their best validation scores early and then complete the shared minimum-epoch window without recovering; APPNP and the official global Polynormer core are materially stronger but retain the same ranking on validation and test. The large test errors therefore reflect model mismatch plus chronological shift rather than nonfinite training or test-set model selection.

C Complete Controlled Extrapolation Results

Table 6: Fixed-interaction extrapolation, NRMSE ↓ (mean ± sample standard deviation, five runs).

Model	Chain 16	Chain 64	Chain 256	Grid 4×4	Grid 8×8
VEN	**.00445±.00077**	**.00615±.00049**	**.00662±.00077**	**.00509±.00057**	**.01288±.00353**
GCN-2	.09936±.00123	.11157±.00155	.11497±.00249	.16469±.00203	.21060±.01038
GCN-3	.05680±.00153	.06610±.00168	.06817±.00153	.09941±.00241	.17311±.01969
GINE-2	.07182±.00100	.09460±.00504	.10506±.00695	.12638±.00060	.19651±.02047
GINE-3	.03204±.00081	.06237±.01125	.07588±.01529	.06199±.00052	.17236±.04577
APPNP	.47856±.00058	.48274±.00128	.48298±.00141	.49209±.00308	.50125±.00619

D Complete CINELDI Feeder Results

Table 7 expands the four-feeder, three-observation-density aggregate reported in the main text. Each cell is computed from five independent trainings with the formal seed set. The advantage is largest on the 80-node feeder and remains present as PMU density changes, rather than arising from one favorable sensor count.

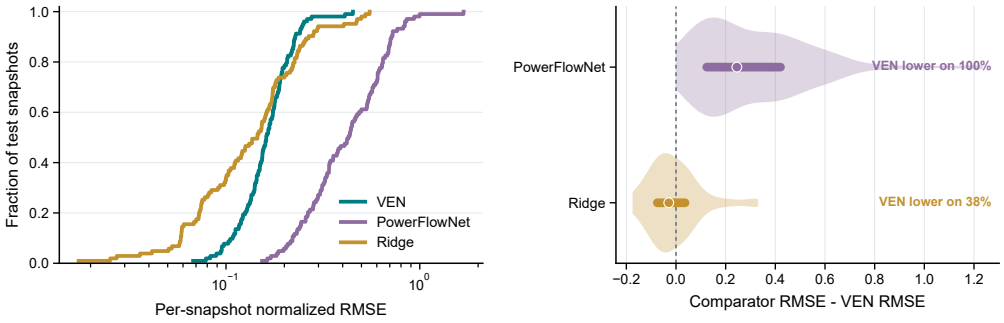
D.1 Paired effect sizes

For the settings with retained per-snapshot predictions, Table 8 bootstraps test snapshots in matched pairs. The reported difference is baseline error minus VEN error, so positive values favor VEN. This

Table 7: CINELDI conditional voltage-state recovery for every feeder and PMU density (test NRMSE ↓, mean ± sample standard deviation over five runs).

Feeder	PMUs	VEN	GCN	GINE	APPNP
Grid 0	2	**.60105±.01170**	.70104±.04503	.65997±.03040	.60526±.02129
	4	**.53434±.00814**	.66257±.02434	.63943±.03365	.54041±.01209
	8	**.44437±.00548**	.60229±.02194	.56648±.02446	.45155±.00362
Grid 1	3	**.58030±.00558**	.67326±.02220	.64705±.01093	.62061±.02208
	5	**.51787±.00779**	.65730±.03073	.64020±.01760	.56694±.00771
	10	**.40032±.00369**	.58821±.03494	.56304±.01662	.44842±.00627
Grid 2	3	**.36730±.01612**	.53051±.01289	.52528±.04044	.44010±.01226
	6	**.24334±.00146**	.42985±.01309	.48032±.04342	.29435±.01024
	11	**.13583±.00091**	.32336±.00905	.35517±.02973	.18453±.00090
Grid 3	4	**.34081±.01319**	.77771±.01020	.75874±.00757	.57162±.01228
	8	**.19187±.00625**	.68910±.01578	.68298±.01413	.41435±.01408
	16	**.13259±.00191**	.56370±.02832	.54234±.01734	.21676±.01043

complements standard deviations across training seeds by measuring the size of the per-snapshot advantage.



(a) Error distribution over the chronological test split **(b) Paired per-snapshot advantage**

Figure 5: Paired SimBench error distributions. **(a)** Empirical cumulative distributions of per-snapshot normalized RMSE on the chronological test split. **(b)** Paired error differences relative to VEN; positive values favor VEN. Violin shapes show the 103 paired test snapshots, thick segments show the interquartile range, and points show the median. Each snapshot value is first averaged across the same five training runs.

VEN has lower error than PowerFlowNet on every retained snapshot and lower error than ridge on 38% of them. Its lower aggregate NRMSE than ridge is therefore driven by reducing larger errors rather than by winning uniformly across time.

Table 8: Paired NRMSE effects on held-out snapshots. Intervals are percentile 95% bootstrap confidence intervals.

Task	Comparator	Difference	95% CI
SimBench	PowerFlowNet	.34070	[.31733, .36418]
SimBench	Ridge	.01270	[.00207, .02345]
CINELDI Grid 3/8	PowerFlowNet	.20528	[.18294, .22562]
CINELDI Grid 3/8	Fixed physical Q	.07172	[.06207, .08131]

D.2 External physical consistency

Cached VEN lies between GCN and GINE in hot throughput on SimBench and exceeds both on GasLib; cold-to-hot output differences stay below 1.67×10^{-6}. Ridge remains substantially faster as a single dense linear map. The preparation, cold-query, hot-query, and memory measurements appear in Table 21.

Table 9: External equation checks and physical estimators. Residual rows recompute independent AC or hydraulic equations from model predictions rather than using the learned Gaussian residual. WLS/MAP rows directly solve the documented physical model and are included as state-estimation baselines.

Task	Model	State NRMSE	External residual	Physical convergence
SimBench AC	VEN	**.17856±.00072**	.00447±.00014 MVA	–
	GCN	.77224±.01765	.42933±.06179 MVA	–
	GINE	.71764±.01626	.43352±.02723 MVA	–
	Ridge	.19126	**.000351 MVA**	–
	Regularized nonlinear WLS	.91803	–	100.0%
GasLib hydraulic	VEN	**.04655±.00115**	1.182±.056 NRMSE	**100.0%**
	GCN	1.24398±.02852	16.931±1.231	10.3%
	GINE	1.24348±.06531	16.904±1.214	19.3%
	Ridge	.32388	**.492**	**100.0%**
	Pressure-conditioned hydraulic MAP	.05292	–	100.0%

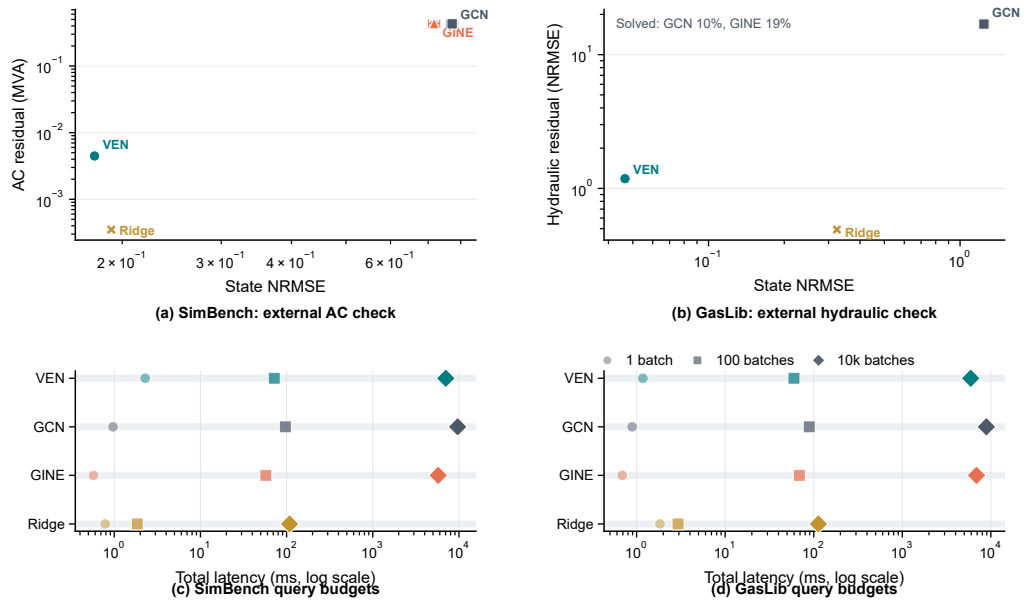


Figure 6: External consistency and repeated-query cost. **(a)** SimBench state error versus AC injection residual recomputed with independent nonlinear power-flow equations. **(b)** GasLib pressure error versus hydraulic residual on ordinary-pipe nodes; the annotation reports the fraction of test states for which the external solve converges. Points show five-run means for neural models and the deterministic ridge result. **(c,d)** Total latency at budgets of 1, 100, and 10,000 query batches, including one-time preparation and median hot-query cost. Timing uses an NVIDIA L40, float32, batch 256, ten warm-up iterations, and nine repeats of 200 synchronized iterations.

GasLib residuals exclude 87 compressor/control-valve endpoints whose ordinary pipe flow is not determined uniquely by node pressure; 47 ordinary-pipe nodes remain. Failed hydraulic solves are retained through the convergence rate rather than silently removed. The deterministic GasLib estimator uses the noisy boundary nominations, public pipe and device parameters, and the 12 noisy observed pressures. A finite-difference Gauss–Newton MAP update estimates only slack pressure around the documented 60-bar prior; compressor ratio and valve setpoint remain at the documented derived-benchmark policy. It reads no hidden pressure, converges in 2.14 iterations on average, and processes both the fixed-policy and MAP estimates in 0.117 seconds per state. SimBench WLS uses pandapower's nonlinear estimator plus a per-bus operating-point prior fitted on training states; it converges in 2.89 iterations and processes 19.1 states/s.

E Marginal Posterior Evaluation

For unobserved coordinate i, marginal NLL is

$$\mathcal{L}_{\mathrm{mNLL}} = \frac{1}{2|U|} \sum_{i \in U} \left[\log(2\pi\sigma_i^2) + \frac{(z_i - \mu_i)^2}{\sigma_i^2} \right].$$

CRPS is computed analytically for each Gaussian marginal. A validation-only scalar calibration uses

$$\alpha^* = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \frac{(z_i - \mu_i)^2}{\sigma_i^2}, \qquad \Sigma_{\mathrm{cal}} = \alpha^* \Sigma_{\mathrm{raw}}.$$

It does not change the mean. For SimBench, channelwise calibration uses a separate validation scale for voltage and angle because their units and shifts differ.

Table 10: Representative marginal uncertainty results.

Task	Objective	NLL	CRPS	Cov. 90%	Cov. 95%
EEGBCI	MSE, raw	1.974	.908	92.34	95.12
SimBench	MSE, raw	.139	.127	99.84	99.96
SimBench	mNLL, raw	**-.520**	**.0896**	94.08	96.54

On EEGBCI, a scalar fitted to validation subject 9 reduces interval width but undercovers unseen subject 10, so the raw posterior is the more defensible report. On SimBench, direct marginal-NLL training makes the raw variance substantially more useful while increasing NRMSE by approximately 0.9%. Neither result establishes calibration under arbitrary distribution shift.

E.1 Observation-noise mechanism

The likelihood covariance in Eq. (7) should distinguish uncertainty already learned from the training distribution from known sensor noise at query time. When sensor specifications provide the latter, we update only that component:

$$R_{\mathrm{sample}} = \left(R_{\mathrm{learned}} - R_{\mathrm{sensor,train}}\right) + R_{\mathrm{sensor,sample}}.$$

Table 11 tests this identity on CINELDI. "Naive scaled" multiplies the complete learned variance and therefore also scales model and residual uncertainty. Under a common threefold noise increase, the decomposed update improves NRMSE and NLL over both the unchanged and naively scaled alternatives in all five runs. When observations are simultaneously sparse, its gain over fixed R is negligible, which indicates that missing evidence, rather than the variance parameter alone, dominates that regime.

Table 11: Five-seed CINELDI observation-noise study. Coverage is reported in percent. The base condition is identical for all three R rules and is shown once.

Condition	R handling	NRMSE ↓	Normalized NLL ↓	90% coverage	95% coverage
Base	Learned training R	.50436±.00738	.61034±.04455	94.00±.83	96.33±.82
Common 3× noise	Fixed training R	.52920±.00844	.65140±.05653	92.88±1.17	95.57±1.08
	Naive scaled R	.54816±.01411	.78171±.02896	95.24±.54	97.23±.58
	Decomposed known-noise R	**.52307±.00746**	**.62448±.03838**	94.86±.39	96.95±.49
Sparse + 3× noise	Fixed training R	.69961±.03826	.95118±.04628	91.90±2.06	95.07±1.45
	Naive scaled R	.76506±.03683	1.12372±.12136	90.34±4.74	94.02±3.49
	Decomposed known-noise R	.69946±.04267	.94805±.04995	92.36±2.36	95.40±1.66

F DGMRF Comparisons and Large-Field Boundary

DGMRF is the closest directly executable probabilistic graph baseline (Oskarsson et al., 2022). We evaluate it under a task-matched shared-field protocol and under its native fixed-mask protocol.

Shared-field protocol. DGMRF keeps the official `DGMRF/FlexLayer` composition and precision $Q = G^{\mathsf{T}}G$. A training wrapper shares this operator across independent snapshots and maximizes their exact Gaussian likelihood; a node-feature or snapshot-context linear mean supplies the task covariates. Parameters are selected on validation masks, frozen, and then conditioned separately for every test mask. SimBench and CINELDI use one official scalar DGMRF per voltage channel because the source model is scalar; the other tasks use one scalar model. LeakDB uses hydraulic conductance, EEGBCI and AQI use spatial similarity, and the remaining engineering graphs use unweighted adjacency. No test snapshot is used for parameter fitting. Shared-field rows use three layers and the five formal run seeds; the full437 scale row uses the official one-layer unweighted option (`dist_weight=0`) for numerical stability. Table 1 reports the resulting task metrics. We denote this shared-snapshot training interface DGMRF; the official DGMRF precision operator is retained.

Native fixed-mask protocol. We use the `graph-dgmrf` repository's bundled `gmrf_prec_mix32_random` task: 5,000 nodes, 14,970 undirected Delaunay edges, 50% observed nodes, and a zero-mean scalar field. The official three-layer DGMRF runs in its recommended configuration for 50,000 iterations. VEN uses the same graph, values, fixed mask, and evaluation protocol, and treats the public node coordinates as static graph attributes in $Q_\theta(G, a)$. No hidden target values enter these attributes. The comparison matches task and data while preserving each method's complete attributed-graph parameterization. VEN has 16 shared local-potential coefficients, fitted by exact observed-data marginal likelihood; sparse factorization computes its posterior mean without a dense covariance. Both methods use seeds $\{7, 42, 123, 2002, 2027\}$.

Table 12: DGMRF Mix task (mean ± sample standard deviation, five runs; lower is better). DGMRF uses the official core from the `graph-dgmrf` repository at commit `b29f8b4`; the adapter modernizes runtime dependencies. Both methods share the graph, field, mask, splits, and metrics, while each retains its documented graph parameterization. Table MAE is measured against sampled hidden values; posterior-moment diagnostics use the latent generating mean and standard deviation.

Model	MAE	RMSE	CRPS	95% interval
DGMRF	**.012646±.000002**	**.015882±.000004**	**.008965±.000003**	**.07544±.00028**
VEN	.013077±.000009	.016437±.000009	.009322±.000008	.08144±.00096

DGMRF has the better latent-mean MAE ($.002728 \pm .000011$ versus $.004369 \pm .000025$), sample-wise point error, CRPS, and interval score, while VEN has the lower latent-standard-deviation MAE ($.001544 \pm .000129$ versus $.002015 \pm .000044$). End-to-end fit-and-inference wall time is 279.2 ± 11.5 s for official DGMRF on one L40 and 53.0 ± 31.3 s for sparse VEN with CPU SuperLU. These times document the reproduced paths on different devices and optimizers. DGMRF is stronger in its native fixed-mask scalar-field regime; VEN remains close in sample-wise point error and supplies the block-state, changing-evidence interface evaluated in the main tasks.

Public large scalar fields. California housing and NREL Wind use the public graph, node features, field, and fixed random mask distributed with the DGMRF repository. The same five seeds determine independent training runs; DGMRF uses the official three-layer configuration, while VEN learns positive local polynomial potentials and an attributed information term, then performs exact sparse conditioning. Table 13 separates predictive scores from the numerical quantities that control the VEN solve.

Table 13: Public large-field recovery (mean ± sample standard deviation, five runs; lower RMSE/CRPS is better). "Solve" is VEN sparse factorization plus posterior mean, and fill is the LU-to-posterior nonzero ratio. DGMRF uses its official training and inference path; its fit time is not compared with the different sparse-optimization path used by VEN.

Dataset	Nodes	Edges	VEN RMSE	DGMRF RMSE	VEN CRPS	DGMRF CRPS	VEN solve (s)	LU fill
California	20,536	61,585	.059319±.000021	.058031±.000009	.032351±.000036	.030394±.000020	.0738±.0086	6.864
NREL Wind	126,652	379,933	.29092±.00210	.27273±.00063	.13080±.00044	.12256±.00025	.6435±.0234	8.233

DGMRF is more accurate on both native static-field tasks, by 2.2% RMSE on California and 6.7% on Wind. The scale result addresses sparse conditioning: VEN processes a 126,652-node, 379,933-edge graph with 7.30 million LU nonzeros and subsecond factorization plus posterior mean. A seed-7 memory audit records 0.94 GB peak RSS. California's five VEN optimizations reach their fixed 70-iteration budget; on Wind, three of five satisfy the optimizer stopping criterion within the 50-iteration budget. These experiments establish a large-sparse-graph operating point and a closest-mechanism boundary, with DGMRF retaining the predictive advantage on its native tasks.

Matched Wind inference audit. Table 14 separates prediction quality from the numerical query path. The official GPU adapter changes only device construction and the scope of `no_grad`; it retains the three-layer `DGMRF/FlexLayer` operator, printed checkpoint parameters, float64 conjugate gradients, and official tolerance. Across five seeds, reproduced RMSE differs from the official logs by at most 2.51×10^{-7}, and the recorded CG iteration counts match. The latency entries document the CPU and native-GPU solver paths separately; RMSE is the protocol-comparable model result.

Table 14: NREL Wind fitted-model inference audit over five seeds. VEN uses sparse CPU SuperLU; DGMRF uses its official float64 CG on CPU or its native CUDA path. Preparation is sparse factorization for VEN and fitted-model query preparation for DGMRF. RSS is process peak memory; CUDA allocation is reported separately. Different devices and direct/iterative solvers preclude a universal speed ranking.

Path	RMSE ↓	Preparation (s)	Cold query (s)	Hot p50 (s)	Peak RSS (GiB)	CUDA alloc. (GiB)
VEN sparse CPU	.290921±.002101	.6691±.0308	.01846±.00519	.01368±.00323	.823±.009	–
Official DGMRF CPU CG	.272729±.000631	–	45.94±1.80	44.59±2.43	126.50±9.49	–
Official DGMRF native GPU	.272729±.000631	.4480±.1140	.9162±.0807	.8831±.1121	1.119±.006	.0845

G Ordinary Graph Prediction Boundary

G.1 Citation link prediction

We fix a 70/10/20 undirected positive-edge split before constructing any model graph. Validation and test positives are removed from Q, propagation adjacency, and structural features; negatives do not overlap positives. Every neural encoder uses the same symmetric inner-product link head.

Table 15: Hits@100 ↑ on leakage-audited citation link prediction (five runs).

Model	Cora	CiteSeer	PubMed
VEN	.258±.014	.464±.009	.134±.003
GCN	.383±.012	.460±.008	.365±.005
GraphSAGE	.405±.052	.449±.005	.309±.015
GATv2	.527±.136	.446±.013	.224±.004
GIN	**.594±.018**	**.514±.008**	**.458±.009**

G.2 Heterogeneous node classification

Relation types enter a scalar precision as

$$Q = \operatorname{diag}(d_{t(i)}) + \sum_{(i,j,r)\in E} w_r (e_i - e_j)(e_i - e_j)^{\mathsf{T}}, \qquad d_t, w_r > 0.$$

Training labels are Gaussian observations. Validation and test labels are never inserted as evidence. On ACM, VEN obtains 91.61 ± 0.75 Macro-F1, below R-GCN's 92.85 ± 0.13. On DBLP, VEN obtains 84.38 ± 0.56, compared with 84.33 ± 0.76 for R-GCN, 83.78 ± 0.82 for GATv2, and 82.79 ± 1.87 for GCN. This establishes interface breadth but not a general classification advantage.

G.3 Unobserved small-graph interface

MUTAG tests the $S = \varnothing$ interface on batches of small graphs rather than conditional state recovery. A pooled classifier reads the posterior mean of a continuous Gaussian VEN with hidden width 32. The dense and multilevel-Schur rows use the same local-potential architecture and differ only in the numerical realization of its Gaussian solve.

Table 16: MUTAG graph classification (five runs). Throughput is measured after warm-up; times are mean training seconds per seed.

Model / backend	Accuracy ↑	Train time	Graphs/s ↑
VEN / dense Cholesky	82.86±4.66	2.08±.22	26,056±2,474
VEN / multilevel Schur	83.57±5.42	10.24±.36	3,666±53
GCN-2	75.00±5.65	2.96±1.96	35,727±3,719
GINE-2	83.57±4.07	3.52±1.33	51,639±6,191

The result verifies that optional observations are an interface of one architecture: setting S empty does not require a second model definition. It also illustrates backend selection. Dense Cholesky is about seven times faster than the current Schur implementation for padded batches of at most 28 nodes, while predictive differences remain within training variability. This experiment does not establish an advantage on general graph classification.

H State-Dependent Precision Extension

The standard architecture reuses $Q_\theta(G, a)$. A direct extension allows the local Gram factors to depend on current sample features, producing $Q_\theta(x, G, a)$. On the WCCI/Grid2Op topology-shift protocol, five-seed complete sample-dependent precision reduces unseen-topology NRMSE from 0.5617 ± 0.0500 to 0.4776 ± 0.0196 and cross-season NRMSE from 0.4936 ± 0.0079 to 0.4789 ± 0.0089. Mean training time rises from 999 to 2111 seconds, and all sample-dependent runs reach the 1,000-epoch budget. This extension quantifies the accuracy–reuse tradeoff induced by state-dependent curvature; the reusable-precision architecture remains the main model.

I Backend Equivalence and Scaling

For identical Q, b, S, R, y and one checkpoint, direct Schur, dense Cholesky, and Woodbury posterior means agree to a maximum absolute error of 9.54×10^{-7} in float32. Cached and cold outputs differ

by at most 1.67×10^{-6} on SimBench and 3.58×10^{-7} on GasLib. These tolerances verify that backend choice does not define a different learned architecture.

Numerical conditioning. Table 17 audits the five formal GasLib checkpoints rather than assuming that positive definiteness implies benign conditioning. Spectra are recomputed in float64 from the learned float32 matrices. The base precision is positive definite in every run but has a condition number near 10^5. Adding the twelve pressure likelihood factors raises the smallest eigenvalue and reduces the median posterior condition number by more than one order of magnitude. All 1,225 test posteriors are finite and positive definite; however, the largest observed posterior condition numbers remain on the order of 10^4. Thus the evaluated protocol is numerically stable without a spectral penalty, but the construction does not provide a uniform condition-number guarantee.

Table 17: GasLib-134 precision conditioning over five formal runs (mean $\pm$ sample standard deviation). Each run contains 245 test observation masks. Posterior columns first summarize masks within a run and then aggregate the resulting statistic across runs.

Quantity	Base Q	Posterior Q^+
Minimum eigenvalue	$(2.14\pm0.57) \times 10^{-3}$	$(1.07\pm0.30) \times 10^{-2}$
κ_2: minimum across masks	–	$(1.07\pm0.26) \times 10^{3}$
κ_2 (base) / median across masks	$(1.05\pm0.33) \times 10^{5}$	$(2.93\pm0.68) \times 10^{3}$
κ_2: maximum across masks	–	$(2.09\pm0.63) \times 10^{4}$
Finite positive-definite systems	5/5 runs	1,225/1,225 queries

The same audit on the two-channel SimBench precision recomputes 192-dimensional spectra in float64 for all five formal checkpoints. All 515 test posteriors are finite and positive definite. The base minimum eigenvalue is $.01072 \pm .00215$ with $\kappa_2 = (6.14 \pm 0.60) \times 10^4$; observations raise the minimum eigenvalue to a per-run median of $.12968 \pm .01279$ and reduce median κ_2 to $(5.07 \pm 0.97) \times 10^3$. The maximum condition number across masks is $(1.16 \pm 0.29) \times 10^4$. Thus the scalar GasLib finding extends to a block state, while still leaving a nontrivial conditioning range.

J Online Evidence and Distance Diagnostics

Online query streams. We fix the learned precision and issue 1,000 batch-one queries for each sensor count $k \in \{2, 4, 8, 16, 32\}$. The fixed-location stream changes only y and caches $A_S = KS^\mathsf{T}(R + SKS^\mathsf{T})^{-1}$. The changing-location stream resamples both S and y, while the insertion/deletion stream modifies one sensor at a time. Base-covariance preparation and fixed-selector-gain preparation are each repeated 20 times; the reported preparation is the sum of their medians. Query latency follows 20 warm-up queries and excludes global CUDA initialization. Gaussian backends use float64; GCN/GINE latency references use float32. Break-even is $\lceil t_{\text{prep}}/(t_{\text{chol}} - t_{\text{woodbury}}) \rceil$.

Table 18: Online Woodbury summary on one NVIDIA L40. Ranges span $k = 2$ to 32 sensors. Each case contains 1,000 queries. Backend error is the maximum coordinatewise difference from rebuilding Cholesky.

Dataset	Query stream	Preparation (ms)	Woodbury p50 (ms)	Woodbury p95 (ms)	Cholesky p50 (ms)	Break-even	Max. error
SimBench	Fixed locations	2.365–2.478	.178–.219	.186–.237	.786–.828	4–5	1.10×10^{-13}
	Changing locations	2.228	.329–.412	.338–.952	.787–.790	5–6	1.13×10^{-13}
	One insertion/deletion	2.228	.329–.412	.339–2.574	.786–.790	5–6	1.10×10^{-13}
GasLib	Fixed locations	1.478–1.532	.179–.181	.185–.214	.732–.736	3	3.78×10^{-15}
	Changing locations	1.341	.331–.367	.339–.961	.733–.735	4	3.78×10^{-15}
	One insertion/deletion	1.341	.329–.371	.340–.987	.732–.735	4	3.78×10^{-15}

Controlled influence. For $Q = \alpha I + wL$, we evaluate chains, balanced trees, and narrow grids at $N \in \{64, 128, 256\}$, $w \in \{.25, 1, 4\}$, $\alpha \in \{.01, .1, 1\}$, and observation-noise variance $R \in \{.01, .1, 1\}$. Every entry of the single-observation response is positive in all 243 configurations, but the minimum absolute influence is 3.23×10^{-196}. At $N = 256, w = 1, \alpha = .1, R = .1$, the furthest

normalized response is 2.30×10^{-35} on the 255-hop chain, 2.30×10^{-4} on the 15-hop tree, and 2.51×10^{-9} on the 66-hop narrow grid. On chains, the last hop above 10^{-3} normalized influence ranges from 3 hops at $(w, \alpha) = (.25, 1)$ to 138 hops at $(4, .01)$. For one observation, changing R scales absolute response but cancels in the response normalized by its self-influence.

Distance-conditioned task results. Table 19 gives the endpoint bins underlying Figure 4(d) and its GasLib replication. Bins group hidden coordinates by the nearest observed node using the fixed test masks; each model has five independently trained seeds.

Table 19: NRMSE by nearest-sensor distance (mean over five runs; overall values include sample standard deviation). Lower is better.

Dataset	Model	Overall	1 hop	13+ hops
SimBench	VEN	.17855±.00074	**.15701**	**.23197**
	GCN-2	.77224±.01765	.29772	.80125
	GINE-2	.71878±.01255	.30663	.75178
	GCN-3	.67634±.01997	.27532	.75727
	GINE-3	.62166±.01179	.29847	.70676
	APPNP	.60402±.00788	.30907	.79722
GasLib	VEN	.04654±.00115	**.02578**	**.06137**
	GCN-2	1.24398±.02852	.47940	1.16240
	GINE-2	1.24345±.06534	.47938	1.15794
	GCN-3	1.05970±.01230	.28017	1.16167
	GINE-3	1.11252±.17042	.36540	1.16978
	APPNP	.85797±.06501	1.07841	1.23553

K Iterative and Scaling Backends

Gaussian belief propagation. Table 20 replaces the exact backend after training while holding Q, b, S, R, y fixed. Canonical-form block GBP is exact on a tree after messages traverse the graph; on Cora it is an iterative loopy method whose convergence is measured rather than assumed. Both diagnostics use seed 7, float64, one L40, and synchronized timing.

Table 20: GBP backend diagnostic. Error is the maximum coordinatewise posterior-mean difference from dense Cholesky; residual is $\|Q^+\mu - b^+\|/\|b^+\|$. Memory is peak incremental GPU allocation. GasLib batches 245 changing-evidence states; Cora solves seven class right-hand sides.

Graph	Iter.	Error	Residual	GBP ms	Chol. ms	GBP MiB	Chol. MiB
GasLib-134	55	6.38×10^{-13}	1.81×10^{-12}	76.03	**7.01**	**5.99**	68.25
Cora	80	2.03×10^{-9}	4.22×10^{-10}	42.94	**15.02**	**6.74**	112.15

GBP reduces peak memory by $11.4\times$ on GasLib and $16.6\times$ on Cora but is $10.8\times$ and $2.9\times$ slower, respectively. The loopy result converges for this strictly grounded M-matrix precision with damping 0.25; it is not a guarantee for arbitrary learned block precisions. GBP is therefore a valid low-memory backend when convergence is monitored, while exact cached backends remain preferable in the moderate systems evaluated here.

Figure 7 measures the three equivalent numerical backends beyond the two application graphs. The protocol uses an NVIDIA L40, float32, batch 256, synchronized CUDA timing, identical posterior inputs, and four observed scalar coordinates per query. Stable-size cases use nine repeats of 500 iterations; large cases use seven repeats of 100 iterations. Throughput is the median over repeats and memory is peak incremental allocation.

The reported multilevel Schur implementation uses a deterministic low-degree independent-set schedule and a dense root, not AMD or nested dissection. Consequently, Figure 7 compares the implemented backend policies rather than claiming an ordering-optimal Schur result; ordering sensitivity remains future work.

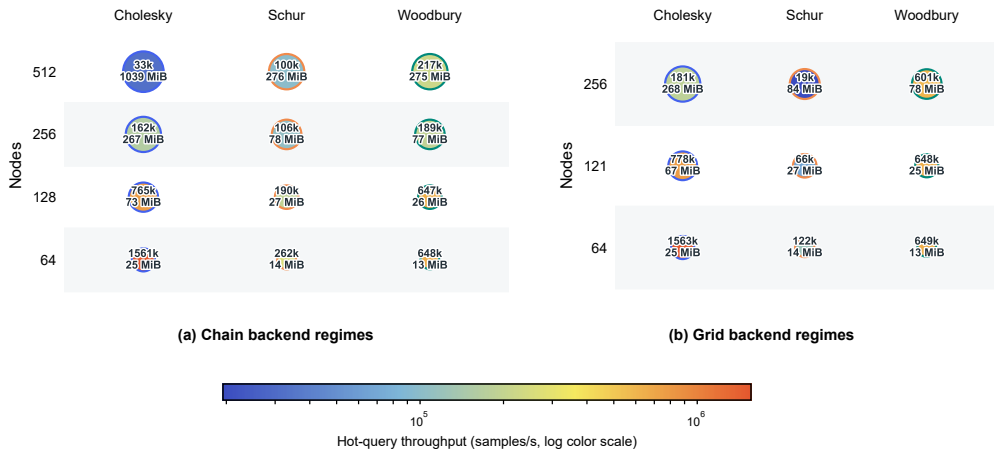


Figure 7: Cost regimes of equivalent inference backends on **(a)** chains and **(b)** square grids. Rows vary node count and columns vary the numerical backend. Circle color encodes median hot-query throughput, circle area encodes peak incremental GPU memory, and labels report both quantities. Cholesky is fast at small sizes but its dense cached batch grows rapidly; Schur uses less memory but incurs fill-in and scheduling overhead on grids; Woodbury occupies the favorable regime for larger repeated sparse-observation cases. Backend choice changes computational cost, not the posterior map.

The implementation therefore selects a backend from graph size, fill pattern, and observation reuse. Table 21 provides the exact application measurements underlying the repeated-query curves in Figure 6. VEN preparation builds the reusable base covariance cache, while ridge preparation fits its linear map. Operational cold latency is preparation plus one stable query after CUDA initialization; the hot path excludes preparation.

Table 21: Unified inference benchmark. Throughput is median hot snapshots/s and memory is peak incremental MiB. Measurements use an NVIDIA L40, float32, batch 256, ten warm-up iterations, and nine repeats of 200 synchronized iterations.

Dataset	Model	Prep. (ms)	Cold (ms)	Hot/batch (ms)	Hot throughput	Peak MiB
SimBench (97 nodes)	VEN	1.580	2.284	0.704	363,667	15.56
	GCN	–	0.967	0.967	264,781	34.90
	GINE	–	0.574	0.574	**445,635**	33.36
	Ridge	0.774	0.784	0.0107	23,830,257	3.95
GasLib (134 nodes)	VEN	0.585	1.177	0.592	**432,204**	8.71
	GCN	–	0.890	0.890	287,597	48.92
	GINE	–	0.689	0.689	371,798	46.51
	Ridge	1.827	1.838	0.0111	23,053,076	14.88

L Reproducibility Statement

Run identity and model selection. Every formal run records the dataset source, preprocessing-cache hash, fixed split, data seed, run seed, model configuration, validation-selected checkpoint, and final test metrics. The formal run seeds are $\{7, 42, 123, 2002, 2027\}$ and the data seed is 7. Thus topology, generated operating states, chronological boundaries, and test masks remain fixed while parameter initialization, minibatch order, and training-time masking vary. Early stopping monitors the dataset's primary validation error after the minimum epoch in Table 4; the selected checkpoint is evaluated once on the test split. Mean and sample standard deviation are computed across the five selected checkpoints. Paired analyses additionally retain a stable sample identifier and each model's prediction for that sample.

Metrics and exclusions. NRMSE divides RMSE by the test-target standard deviation computed from training normalization statistics. RMSE, MAE, NLL, and coverage are evaluated only on the coordinates designated hidden before model execution. Observed coordinates, power-system reference buses, and task-defined non-target nodes are excluded identically for every model. Link prediction constructs the train graph only after holding out validation and test edges; heterogeneous classification inserts training labels only. Physical residual evaluators consume saved predictions and independent simulator equations. Non-convergent GasLib hydraulic reconstructions remain visible through the convergence rate, and endpoints whose flow is not identifiable from pressure alone are excluded by one fixed rule shared across models.

Numerics and timing. Gaussian systems use float32 in the reported neural experiments, positive diagonal floors in the learned Gram parameterization, and a small diagonal jitter recorded in each configuration. A run is marked failed if training produces a nonfinite objective, the solver produces a nonfinite posterior, or the fixed wall-clock budget is exceeded; failed runs are not replaced by a new seed. Backend equivalence uses identical tensors and reports the maximum coordinatewise posterior difference. Runtime measurements synchronize CUDA, separate preparation, cold query, and hot cached query, and record batch size, dtype, device, warm-up count, repetitions, aggregation statistic, and peak incremental memory. Dataset loading, training, and checkpoint deserialization are outside the hot-query boundary.

Artifact organization. The anonymous artifact maps every manuscript table to immutable JSON, per-seed configurations, aggregation output, and logs. It also records repository URLs and commits, environment lockfiles, preprocessing commands, and hardware metadata. Baseline adapters start from public implementations; model layers and forward computations are retained. Compatibility changes cover task input/output adapters, fixed split injection, result logging, and the shared metric interface. DGMRF additionally replaces the field-specific training entry point by the exact shared-snapshot likelihood described in Appendix F; the official precision operator itself is unchanged.