NUMERICAL REASONING ON KNOWLEDGE GRAPHS WITH CONVOLUTIONAL HISTOGRAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Numerical complex query answering methods often encode scalar values as latent vectors or learned density parameters. Although expressive, these representations do not preserve numerical order, distance, and uncertainty together in an explicit numerical space. Their intermediate numerical states can therefore be difficult to inspect in the original units. We present CHOREOGRAPH, a framework that represents each observed scalar as a localized histogram on an ordered grid defined separately for each numerical attribute. Attribute-wise quantile binning allocates finer resolution where training literals are dense. Relation projection kernels (RPKs) use one-dimensional convolution to model how numerical mass changes across relational edges and can be fitted efficiently. Distributional message passing then estimates unobserved numerical slots. CHOREOGRAPH composes numerical operations using histogram-valued intermediate states. This discrete representation supports comparison, set, and arithmetic operations without operator-specific learning. Each intermediate numerical state can therefore be traced to intervals in the original units. We evaluate CHOREOGRAPH on FB15K-237, FB15K, DB15K, and YAGO15K using benchmark queries from prior work. Across all four datasets, CHOREOGRAPH outperforms the evaluated baselines on most query structures. On FB15K, all RPKs are fitted in 9.4 seconds, and total training is $3.3\times$–$22.4\times$ faster than recent baselines.

1 INTRODUCTION

Knowledge graphs (KGs) organize facts as relations among entities and support applications ranging from search to question answering. A single fact rarely suffices for a realistic question. For example, answering *Which Academy Award-winning actors appeared in films directed by Christopher Nolan?* requires composing relational projections with set intersection. Complex query answering (CQA) evaluates such compositions over an incomplete KG.

Knowledge graph embedding (KGE) models map entities and relations into latent spaces that support link prediction and other downstream tasks (Bordes et al., 2013; Trouillon et al., 2016; Sun et al., 2019). In CQA, these representations and learned scores provide building blocks for multi-hop and logical reasoning (Hamilton et al., 2018; Ren et al., 2020; Arakelyan et al., 2021). Real KGs also contain numerical facts such as population, area, and dates. Adding the constraint *born after 1980* to the example above introduces an order constraint on a numerical attribute that must be composed with relational and logical operations. Literal-aware KGE methods extend structural representations with numerical side information (García-Durán & Niepert, 2018; Kristiadi et al., 2019). Numerical CQA requires more than incorporating literals into entity representations: it must carry numerical information through multiple operations, compare uncertain values, combine numerical branches, and return entities or numbers. Existing methods handle these requirements using scalar predictors, learned numerical distributions, parametric decoders, or embedded value nodes (Demir et al., 2023; Bai et al., 2023; Peng et al., 2024; Zhang et al., 2025).

These approaches can create a *representation–algebra mismatch*. A scalar is not merely an identifier: it has a known ordering, meaningful distances, and well-defined arithmetic operations. Learned vectors may capture these properties empirically. Their geometry, however, does not by itself ensure that numerical order is preserved or that vector-space operations correspond to scalar arithmetic.

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.

Parametric distributions, such as Gaussian mixture models, represent uncertainty more explicitly but constrain intermediate states to a chosen distribution family.

Latent numerical states can be difficult to inspect in the original units, while point decoding may discard uncertainty or multimodality. The issue is not whether neural embeddings can represent numbers, but which numerical properties should be learned and which can be specified directly. We instead ask: *what if reasoning operated directly on each attribute's ordered numerical axis?*

We answer this question with **C**onvolutional **H**istogram f**O**r numerical **RE**asoning **O**n knowledge **GRAPH**s (CHOREOGRAPH). Each observed numerical literal is represented as a localized histogram on an attribute-specific ordered grid. Attribute-wise quantile bins provide finer resolution where training literals are dense without placing years, lengths, and coordinates on a shared scale. A relation projection kernel (RPK) uses one-dimensional convolution to model relation-conditioned mass transport along an attribute grid. Distributional message passing then estimates unobserved numerical slots. Comparison, arithmetic, and compatible set operations are implemented through explicit transformations of histogram states, without operator-specific neural networks. A ComplEx backbone handles entity-space relational projection. Histogram states are used to score candidate entities and numerical answers; the restricted set–scalar arithmetic executor additionally decodes its second and subsequent operands to scalars.

The design has three practical benefits: accuracy, traceability, and efficient learning. Each intermediate histogram exposes its support, modes, and entropy on the ordered numerical axis, while RPKs make relation-conditioned transport explicit. Across benchmark queries from LitCQD (Demir et al., 2023) and NRN (Bai et al., 2023) on four KGs, CHOREOGRAPH outperforms the corresponding baselines on most query structures. Each supported RPK is fitted independently by ridge regression followed by simplex projection. On FB15K (Bordes et al., 2013), all RPKs are fitted in 9.4 seconds, and total training is $3.3\times$ faster than LitCQD and $22.4\times$ faster than NRN. Query-time execution is comparable to that of LitCQD under the same inference workload. An anonymized implementation of CHOREOGRAPH is available[1].

Our contributions are threefold:

- We represent numerical CQA states as explicit histograms on ordered attribute grids. Their support, uncertainty, and modes remain directly traceable throughout query composition.
- We introduce RPKs and distributional message passing, together with a query executor that performs comparison, set, and arithmetic operations directly on histogram states.
- Across benchmark queries from LitCQD and NRN, CHOREOGRAPH outperforms the corresponding baselines on most query structures and requires substantially less training time.

2 Related Work

Structural and numerical KG representations. KGE methods such as TransE, ComplEx, and RotatE learn entity and relation representations for knowledge graph completion (Bordes et al., 2013; Trouillon et al., 2016; Sun et al., 2019). Literal-aware extensions incorporate numerical side information into these representations, primarily to improve missing-fact prediction. ChainsFormer predicts a missing numerical attribute of a target entity using multi-hop relation–attribute chains as context (Zhao et al., 2025). Pretrained KGE scores can also serve as atomic predictors in decomposition-based CQA (Arakelyan et al., 2021). CHOREOGRAPH uses ComplEx for entity projection, while representing and composing intermediate numerical states separately.

Numerical complex query answering. LitCQD extends Continuous Query Decomposition (CQD) (Arakelyan et al., 2021) with a ComplEx-N3 structural predictor, TransEA point regression, continuous filters, and scalar aggregation (Demir et al., 2023; Lacroix et al., 2018; Wu & Wang, 2018). It retains a real-valued output, but represents numerical uncertainty indirectly through an entity beam and point predictions. NRN alternates between entity and numerical states. Its numerical states are parametric densities over DICE or sinusoidal embeddings, and its query operators are learned neural transformations (Bai et al., 2023; Sundararaman et al., 2020). CNEQ separates

[1] https://anonymous.4open.science/r/CHOREOGRAPH-B450/

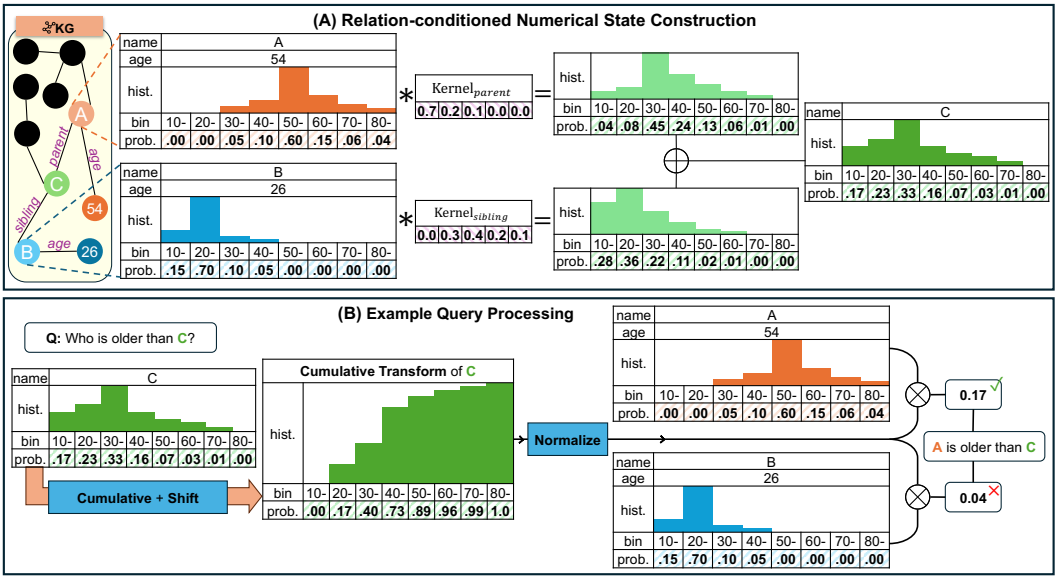

Figure 1: **CHOREOGRAPH method overview.** (A) RPKs transport observed literal histograms by zero-padded 1D convolution ($*$; Sec. 4.2), and incoming messages are combined by normalized entropy-weighted aggregation ($\oplus$; Eq. (4)) to estimate an unobserved numerical slot. (B) An order transform first produces an unnormalized comparison-score profile, which is normalized into a histogram-valued query state before reverse attribute projection ranks entities by distributional agreement ($\otimes$ denotes the histogram inner product; Sec. 4.4). Every intermediate numerical state remains defined on the attribute grid and in the original units.

a number-to-entity predictor from an entity filter and combines deterministic magnitude encoding with MLP-predicted Gaussian mixtures (Peng et al., 2024). CNR-NST represents intermediate answers as fuzzy candidate scores. It jointly learns entity–relation–entity, entity–attribute–value, and value–function–value predictors with Multi-ComplEx; binary arithmetic follows the fuzzy extension principle (Zhang et al., 2025). CHOREOGRAPH instead places probability mass directly on an ordered attribute grid and learns relation-conditioned transport on that grid. It neither decodes numerical distributions from latent query vectors nor encodes numerical order as learned edges in an augmented value graph.

3 PRELIMINARIES

Let $\mathcal{E}$, $\mathcal{R}$, $\mathcal{A}$, and $\mathcal{N}$ denote the sets of entities, directed relations, numerical attributes, and numerical literals. A numerical knowledge graph is $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{A}, \mathcal{N}, \mathcal{T}_{\text{rel}}, \mathcal{T}_{\text{num}}\}$. Its relational triples satisfy $(e_1, r, e_2) \in \mathcal{T}_{\text{rel}} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Its numerical triples satisfy $(e, a, n) \in \mathcal{T}_{\text{num}} \subseteq \mathcal{E} \times \mathcal{A} \times \mathcal{N}$, where $\mathcal{N} \subseteq \mathbb{R}$. Superscript tr restricts a set to the training graph. Let $\mathcal{E}_a^+ = \{e : \exists n\ (e, a, n) \in \mathcal{T}_{\text{num}}^{\text{tr}}\}$ denote entities observed with attribute a. We call each entity–attribute pair (e, a) a *numerical slot*.

A query composes operators whose inputs and outputs are either ENTITY states or NUMERIC histogram states. It may use entity-space relational projection rp, attribute projection ap, reverse attribute projection rap, numerical projection np, intersection, and union. A numerical state contains attribute-specific channels in their original units (e.g., years, meters, or degrees). It remains distribution-valued until rap returns to entities or the query terminates. RPK denotes a relation projection kernel for numerical transport, not the entity-ranking operator rp. Bold lowercase symbols are vectors, $[\boldsymbol{v}]_j$ is component j, and $\propto$ omits the normalization step in histogram-valued expressions. Histogram-valued operator outputs are normalized when they have positive total mass; zero-mass cases are handled separately by the executor. Appendix A summarizes the notation.

4 METHOD

Numerical CQA alternates between entity sets and distributions over scalar values. CHOREOGRAPH represents these two state types separately and makes every transition between them explicit. A ComplEx backbone performs entity-space relational projection. Each nonempty numerical state contains one or more histograms on attribute-specific ordered grids. The method consists of four stages: numerical literal initialization, relation-conditioned transport learning, distributional message passing, and query execution. Numerical operators act on histogram-valued states, with the scalar–decoding exceptions described in Sec. 4.5. Figure 1 presents the method.

4.1 NUMERICAL LITERAL INITIALIZATION

For each numerical observation (e, a, n), we place n on an ordered attribute grid, separating its position from initialization smoothing.

Attribute-wise quantile binning. Numerical attributes in KGs are often unevenly distributed. Uniform-width bins preserve numerical order but can allocate resolution inefficiently under such distributions. Appendix D.3 illustrates this effect using FB15K birth-year and longitude literals.

We therefore build B *quantile bins* for each attribute. Their $B + 1$ boundaries are estimated from the attribute's training literals using equal-frequency binning (Dougherty et al., 1995):

$$b_{a,j} = \mathrm{Quantile}_{j/B}\big(\{n : (e, a, n) \in \mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}\}\big), \quad j = 0, \ldots, B. \tag{1}$$

We write $\mathcal{B}_a = (b_{a,0}, \ldots, b_{a,B})$ for the resulting grid. The intervals $[b_{a,j}, b_{a,j+1})$ target approximately equal training-literal counts, although ties and finite-sample effects can cause uneven occupancy. Each bin's width $b_{a,j+1} - b_{a,j}$ contracts in dense regions and expands in sparse regions. These grids remain separate even for attributes that share a unit, so no common numerical scale is imposed. Rank-based allocation preserves bin assignments under positive affine unit conversions and accommodates skewed value ranges. We fit every boundary using $\mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}$ only. Values outside the fitted range are clipped to the nearest endpoint rather than removed. Appendix D.3 reports how often held-out literals fall within these training-derived ranges.

Gaussian literal initialization. For each observed numerical triple $(e, a, n) \in \mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}$, we initialize the histogram of slot (e, a) by mapping its literal n onto the attribute grid $\mathcal{B}_a$. A one-hot assignment can give nearby values disjoint representations when they fall on opposite sides of a bin boundary. We instead use a Gaussian soft assignment over neighboring bins. For a positive-width bin j satisfying $b_{a,j} \le n < b_{a,j+1}$, define the continuous grid coordinate $\chi_a(n) = j + (n - b_{a,j})/(b_{a,j+1} - b_{a,j}) \in [0, B]$. Repeated quantile boundaries are retained rather than merged. Literal assignment uses a right-sided boundary search; when the selected bin has zero width, its continuous grid coordinate is set to the bin-index midpoint $j + \frac{1}{2}$. The normalized initial histogram $\boldsymbol{p}^0_{e,a}$ is

$$p^0_{e,a,j} \propto \exp\left[-\frac{(j + \frac{1}{2} - \chi_a(n))^2}{2\sigma^2}\right], \qquad j = 0, \ldots, B - 1, \tag{2}$$

where the omitted constant normalizes over bins. The Gaussian width σ controls smoothing in grid coordinates. The state of slot (e, a) contains a conditional histogram $\boldsymbol{p}_{e,a}$ and an attribute-existence score $\eta_{e,a} \in [0, 1]$. The histogram describes possible values conditional on attribute presence; observed slots initialize the graph-based existence component to one. Keeping these components separate prevents numerical spread from being conflated with evidence of attribute presence.

4.2 LEARNING RELATION-CONDITIONED NUMERICAL TRANSPORT

Numerical values are not simply copied along relational edges. Parents and children tend to differ in birth year, and neighboring places exhibit structured coordinate changes. Such changes may also be uncertain or multimodal. A generic vector projection can learn these patterns but does not expose their effect on the ordered numerical axis. We model the effect directly: each directed relation r transports an attribute-a distribution by convolution. The same local kernel is applied across grid

positions, with boundary effects handled by truncation and renormalization. The kernel represents a distribution over shifts in grid coordinates, while the grid retains numerical order.

For each directed relation–attribute pair (r, a) with at least one training path, we learn a relation projection kernel (RPK) $\boldsymbol{k}_{r,a}$. Reciprocal relations are treated as distinct directed relations, and their supported RPKs are fitted independently of the corresponding forward-relation kernels. The kernel has odd width K and lies on the probability simplex $\Delta_K = \{\boldsymbol{k} \in \mathbb{R}_+^K : \sum_j k_j = 1\}$. Its center denotes zero shift, and entries on either side denote negative or positive shifts. Let $*_{\mathrm{zp}}$ denote zero-padded one-dimensional convolution. The numerical transport operator is $\mathcal{P}_{r,a}(\boldsymbol{p}) \propto \boldsymbol{p} *_{\mathrm{zp}} \boldsymbol{k}_{r,a}$. Mass shifted outside the grid is truncated, not wrapped, before histogram normalization. This operator acts only on the conditional histogram; we model the attribute-existence score separately.

The supervision for (r, a) consists of one-hop edges whose endpoints both have observed values for attribute a: $\mathcal{D}_{r,a} = \{(s, t) : (s, r, t) \in \mathcal{T}_{\mathrm{rel}}^{\mathrm{tr}},\ s, t \in \mathcal{E}_a^+\}$. We call each member of $\mathcal{D}_{r,a}$ an ***RPK training path***. For a training path (s, t), $\boldsymbol{p}_{s,a}^0$ and $\boldsymbol{p}_{t,a}^0$ are the initial histograms of the source and target slots (s, a) and (t, a), respectively, as defined in Eq. (2). Let $\boldsymbol{C}(\boldsymbol{p})$ be the zero-padded Toeplitz matrix satisfying $\boldsymbol{C}(\boldsymbol{p})\boldsymbol{k} = \boldsymbol{p} *_{\mathrm{zp}} \boldsymbol{k}$. For each supported pair, we first solve an unconstrained ridge problem (Hoerl & Kennard, 1970). We then project its solution onto the simplex:

$$\widetilde{\boldsymbol{k}}_{r,a} = \arg\min_{\boldsymbol{k} \in \mathbb{R}^K} \sum_{(s,t) \in \mathcal{D}_{r,a}} \left\| \boldsymbol{C}(\boldsymbol{p}_{s,a}^0)\boldsymbol{k} - \boldsymbol{p}_{t,a}^0 \right\|_2^2 + \lambda \|\boldsymbol{k}\|_2^2, \qquad \boldsymbol{k}_{r,a} = \Pi_{\Delta_K}(\widetilde{\boldsymbol{k}}_{r,a}). \tag{3}$$

Here $\lambda \geq 0$ is the ridge coefficient, and Π_{Δ_K} is Euclidean projection onto the simplex. The simplex constraint makes each kernel a distribution over shifts and ensures nonnegative convolution outputs. Ridge regularization stabilizes estimation when training evidence is limited or redundant. The objective factorizes over (r, a), so fitting one kernel requires neither query supervision nor gradients through another kernel. We solve Eq. (3) only when $|\mathcal{D}_{r,a}| > 0$ and call such a pair *supported in training*. An unsupported pair uses the uniform kernel $k_{r,a,j} = 1/K$ and a separate reliability weight (Appendix B.3).

Traceable numerical states. Histogram states expose support, modes, spread, and entropy directly on the attribute grid, where bin j maps to $[b_{a,j}, b_{a,j+1})$. RPKs expose probability over grid shifts: central mass indicates value preservation, asymmetry indicates direction of change, and broad kernels represent variable changes. Because quantile bins have unequal widths, these shifts are defined in grid coordinates rather than as constant additive offsets. Figure 1 illustrates this traceability; we use the term operationally, without claiming a post-hoc explanation model.

4.3 Distributional Message Passing

RPKs are fitted from observed source–target pairs, but queries may reach unobserved numerical slots. CHOREOGRAPH propagates observed histograms along training edges, using RPKs to form relation-conditioned messages.

The observed histograms in Eq. (2) seed L layers of message passing over $\mathcal{T}_{\mathrm{rel}}^{\mathrm{tr}}$. At layer 0, only observed numerical slots are instantiated as message sources; unobserved uniform fallback states do not propagate. At layer $\ell \in \{0, \ldots, L-1\}$, edge (s, r, t) sends $\boldsymbol{m}_{s \to t,a}^{(\ell)} = \mathcal{P}_{r,a}(\boldsymbol{p}_{s,a}^{(\ell)})$ to slot (t, a). For its incoming messages $\{\boldsymbol{m}_i\}$, we aggregate by

$$H(\boldsymbol{p}) = -\sum_j p_j \log(p_j + \epsilon), \qquad \alpha_i = \frac{e^{-H(\boldsymbol{m}_i)}}{\sum_\nu e^{-H(\boldsymbol{m}_\nu)}}, \qquad \boldsymbol{p}_{t,a}^{(\ell+1)} = \sum_i \alpha_i \boldsymbol{m}_i, \tag{4}$$

where H is a numerically stabilized approximation to Shannon entropy, $\epsilon > 0$ ensures numerical stability, and α_i is the normalized message weight. Low-entropy messages receive more weight. After each layer, observed slots are reset to their anchor histograms. Inferred slots can serve as sources in the next layer. If no numerical message arrives, the conditional histogram remains uniform. Attribute existence is tracked separately by the attribute-existence score $\eta_{e,a}$. Its graph component propagates for two layers using a reliability-weighted noisy-OR (Pearl, 1988) and clamps observed slots to one. Inferred first-layer existence scores can serve as sources in the second layer. A classifier over frozen ComplEx entity features is fused with the graph score after propagation. Appendix B.3 gives the full construction, reliability, fusion, and calibration details.

4.4 HYBRID ENTITY–NUMERICAL QUERY EXECUTION

Complex queries move between entity and numerical states. CHOREOGRAPH keeps these domains explicit: entity-space relational projection acts on entity scores, numerical operators act on histograms, and attribute projection and reverse attribute projection connect them. The executor propagates either an entity score vector $\boldsymbol{z}$ or a numerical state $\mathcal{H} = \{(a, \eta_a, \boldsymbol{p}_a, \mathcal{B}_a)\}$. Here z_e scores entity e, and $z_e = -\infty$ excludes it. Each tuple in $\mathcal{H}$ contains attribute a, its existence score η_a, conditional histogram $\boldsymbol{p}_a$, and grid $\mathcal{B}_a$.

The numerical-literal operator returns a numerical state with a single attribute channel, $\mathrm{nv}_a(x) = \{(a, 1, \boldsymbol{g}_a(x), \mathcal{B}_a)\}$, where the Gaussian literal histogram $\boldsymbol{g}_a(x)$ is obtained by substituting literal x into Eq. (2). Entity and numerical-literal anchors initialize the two state types.

Entity-space relational projection. An entity anchor assigns score zero to itself and $-\infty$ elsewhere. Projection then uses a ComplEx backbone with max-plus propagation over retained source entities and batched candidate scoring. ComplEx scoring details appear in Appendix B.1.

Attribute projection maps an entity state to a numerical state. Rather than selecting one entity and regressing one scalar, it combines precomputed slot states according to the entity scores. Let $\mathcal{S} = \{e \in \mathcal{E} : z_e > -\infty\}$ denote the active entities. For $e \in \mathcal{S}$, let $\omega_e = \exp(z_e)/\sum_{u\in\mathcal{S}} \exp(z_u)$ be its normalized entity weight. Attribute projection returns a channel with $\eta_a = \sum_{e\in\mathcal{S}} \omega_e \eta_{e,a}$. When $\eta_a > 0$, its conditional histogram is $\boldsymbol{p}_a = \eta_a^{-1} \sum_{e\in\mathcal{S}} \omega_e \eta_{e,a} \boldsymbol{p}_{e,a}$. Thus each source contributes according to its softmax-normalized entity weight and attribute-existence score.

Reverse attribute projection maps a numerical state back to entities. It compares distributions within the same attribute channel rather than replacing the query histogram $\boldsymbol{p}_a^{\mathrm{qry}}$ by its mode. Candidate entity e receives score $s_{\mathrm{rap}}(e \mid a, \boldsymbol{p}_a^{\mathrm{qry}}) = \eta_{e,a} \langle \boldsymbol{p}_a^{\mathrm{qry}}, \boldsymbol{p}_{e,a} \rangle$.

4.5 HISTOGRAM-SPACE OPERATORS AND NUMERICAL DECODING

Histogram-valued operators allow multi-step composition without decoding every intermediate state to a scalar. An operator may move, sharpen, broaden, or combine nonnegative mass on the ordered grid without requiring an immediate scalar decode. The restricted set–scalar executor is an exception: it retains the first operand as a distribution but decodes later operands, as detailed below.

Comparison and arithmetic operators. For ordered bins $j = 0, \ldots, B-1$, the less-than and greater-than transforms first form cumulative comparison-score profiles from the source histogram $\boldsymbol{p}$:

$$[\mathrm{lt}(\boldsymbol{p})]_j = \sum_{i<j} p_i, \qquad [\mathrm{gt}(\boldsymbol{p})]_j = \sum_{i>j} p_i. \tag{5}$$

These profiles are not normalized value distributions: each component is a candidate-wise comparison probability at the bin level, and the components need not sum to one. During numerical projection, the profile is combined with its incoming channel mass and the resulting nonempty state is normalized jointly over its attribute–bin entries. Under this convention, gt accumulates source mass in bins above the candidate bin, while lt accumulates mass below it. Both transforms use the grid ordering and require no learned comparison network. Equality to a literal x is scored using the same Gaussian histogram as numerical decoding: $\mathrm{eq}(\boldsymbol{p}, x) = \langle \boldsymbol{p}, \boldsymbol{g}_a(x) \rangle$. For entity filtering, each comparison score is multiplied by the corresponding attribute-existence score $\eta_{e,a}$. Numerical projection operates directly on discrete bin masses rather than on a persistent continuous density. For scale transforms $f(x) = cx$, the executor evaluates the normalized bin-mass vector in grid coordinates at inverse-mapped output representatives using interpolation, and then renormalizes the result. For binary arithmetic, let v_i and u_j denote the representative scalar values of input bins with masses p_i and q_j. For output bins $\{I_k\}$, the full arithmetic executor uses a product joint distribution over the input bins and accumulates mass as

$$[\mathrm{arith}_f(\boldsymbol{p}, \boldsymbol{q})]_k \propto \sum_{i,j} p_i q_j \, \mathbb{1}[f(v_i, u_j) \in I_k]\,. \tag{6}$$

The output grid spans the finite derived values. Because Eq. (6) accumulates discrete bin masses rather than continuous density values, it does not require a bin-width Jacobian. The set–scalar

executor below instead transports mass associated with candidate literal values. We discretize each result on an adaptive output grid spanning its finite derived values while preserving total mass. The full histogram pushforward in Eq. (6) uses both input histograms without first reducing either to a point estimate.

Our set–scalar arithmetic queries use a restricted form of this pushforward. The first operand remains distribution-valued, while each later operand is either an explicit numerical literal or a singleton-target attribute projection. These later operands are represented as histogram states and decoded to a single candidate scalar before arithmetic. Masses associated with identical output values are summed before the result is discretized back to a histogram. Only the first operand remains distribution-valued; Appendix B.4 details implementation.

Intersection and union. For m entity branches $\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}$, scores from different paths may have different scales. We use standardized-sum intersection $\mathcal{I}(\boldsymbol{z}^{(1:m)}) = \sum_{\nu=1}^{m} (\boldsymbol{z}^{(\nu)} - \mu_\nu)/\tau_\nu$. Here μ_ν and τ_ν are the mean and standard deviation of the finite scores in branch ν. Union uses the elementwise maximum, $\mathcal{U}(\boldsymbol{z}^{(1:m)}) = \max_\nu \boldsymbol{z}^{(\nu)}$. For compatible attribute channels, numerical intersection and union use normalized elementwise products and maxima, respectively: $[\mathcal{I}(\boldsymbol{p}^{(1:m)})]_j \propto \prod_\nu p_j^{(\nu)}$ and $[\mathcal{U}(\boldsymbol{p}^{(1:m)})]_j \propto \max_\nu p_j^{(\nu)}$. These numerical operators implement soft aggregation over histogram states. If an intersection has zero total product mass, the corresponding numerical channel is omitted, yielding an empty numerical state when no compatible channel remains.

Compatible channels must use the same unit and, when specified by the query, the same attribute. Before combination, we align their grids using the mass-preserving interval-overlap rule specified in Appendix B.2. Incompatible channels remain separate rather than being placed on one attribute grid. Nonempty numerical outputs can therefore be passed to subsequent operators that satisfy the same compatibility requirements.

At the end of a query with a numerical answer, the executor performs numerical-literal ranking or scalar prediction. For candidate literal x, we construct $\boldsymbol{g}_a(x)$ on the final attribute grid and score it as $s_{\mathrm{num}}(x) = \eta_a \langle \boldsymbol{p}_a, \boldsymbol{g}_a(x) \rangle$. This inner product measures agreement with the candidate's histogram rather than reading a single bin. For scalar prediction, we return the center of the three-bin window with maximum mass. The executor thus supports both numerical-literal ranking and scalar prediction, with broad or multimodal numerical states represented explicitly before decoding. Appendix B.2 derives the ridge solution and establishes closure for the specified nonempty histogram operations.

5 Experiments

Datasets, queries, and baselines. We evaluate on FB15K-237 (Toutanova & Chen, 2015) and the numerical KGs FB15K, DB15K, and YAGO15K (Liu et al., 2019). FB15K-237 numerical literals come from LitCQD (Demir et al., 2023); we regenerate queries following the LitCQD and NRN (Bai et al., 2023) query structures on shared, leakage-controlled splits (Appendix C.2). We compare with LitCQD and with NRN using GQE, Q2B, and Q2P backbones (Hamilton et al., 2018; Ren et al., 2020; Bai et al., 2022), and with a ComplEx backbone-only baseline (Trouillon et al., 2016). Candidate ranking uses filtered mean reciprocal rank (MRR), while scalar prediction uses mean absolute error (MAE) and mean squared error (MSE) on literals normalized to $[0, 1]$. MRR is reported as a percentage, and MAE and MSE are multiplied by 10^2. Avg_M is the macro-average over query structures, while Avg_W weights them by query count. We report baselines that can be trained and evaluated end-to-end under this protocol. Appendices C.1–C.4 give hyperparameter, protocol, query-label, and baseline details.

5.1 Main Results

Table 1 reports results on LitCQD benchmark queries. CHOREOGRAPH outperforms LitCQD on most query structures across all four datasets, with its largest MRR gains on nr and $\mathrm{3p3i}$. LitCQD remains better on nrp and nrpi on YAGO15K. For scalar prediction, LitCQD has lower $\mathrm{1ap}$ MAE and MSE on all four datasets, while CHOREOGRAPH has lower $\mathrm{2pa}$ MAE and MSE on all four.

Table 1: Results on LitCQD benchmark queries. Candidate-ranking columns report MRR (%); scalar-prediction columns report MAE or MSE, multiplied by 10^2 for readability. Mean values are shown; full results with standard deviations are reported in Appendix D.5.

Dataset	Method	1rp	2rp	ppp	2pi	3p3i	2pip	pppi	2pu	2pup
FB15K-237	LitCQD	40.88	14.92	9.07	33.63	45.72	23.26	26.73	15.02	11.40
	CHOREOGRAPH	**44.35**	**18.74**	**14.42**	**37.76**	**49.57**	**25.26**	**30.32**	**18.69**	**15.14**
FB15K	LitCQD	44.56	16.77	10.32	35.50	47.72	25.89	31.11	18.34	11.59
	CHOREOGRAPH	**47.47**	**19.53**	**15.78**	**39.24**	**50.83**	**27.31**	**33.71**	**22.05**	**14.98**
DB15K	LitCQD	27.29	8.87	6.24	28.96	52.44	15.11	27.08	5.38	5.05
	CHOREOGRAPH	**27.90**	**10.10**	**9.22**	**37.62**	**64.20**	**20.67**	**30.65**	**5.99**	**6.03**
YAGO15K	LitCQD	24.21	5.70	4.04	18.01	36.28	9.63	14.91	5.30	4.05
	CHOREOGRAPH	**24.39**	**5.71**	**6.14**	**23.84**	**41.65**	**12.85**	**17.77**	**5.85**	**4.45**

Dataset	Method	nr	nrp	nrpi	1ap(MAE)	1ap(MSE)	pa(MAE)	pa(MSE)	2pa(MAE)	2pa(MSE)
FB15K-237	LitCQD	3.38	4.69	18.14	**5.29**	**1.19**	3.68	**0.70**	4.90	0.77
	CHOREOGRAPH	**19.21**	**7.49**	**20.83**	5.82	1.77	**3.51**	0.77	**4.00**	**0.63**
FB15K	LitCQD	2.79	5.02	19.88	**5.20**	**1.21**	2.77	**0.34**	6.18	1.36
	CHOREOGRAPH	**24.82**	**6.76**	**27.13**	5.97	2.00	**2.67**	0.39	**3.89**	**0.75**
DB15K	LitCQD	0.74	2.95	14.37	**3.53**	**0.97**	**2.63**	**0.37**	3.62	0.42
	CHOREOGRAPH	**10.02**	**4.02**	**21.58**	4.07	1.49	2.75	0.45	**2.80**	**0.32**
YAGO15K	LitCQD	1.57	**2.00**	**20.74**	**3.27**	**0.52**	2.48	0.54	4.04	0.66
	CHOREOGRAPH	**3.86**	1.59	20.02	3.60	0.67	**2.28**	**0.43**	**2.57**	**0.45**

Table 2: MRR (%) on NRN benchmark queries.

Dataset	Method	Avg_M	Avg_W	1p	2p	2i	3i	pi	ip	2u	up
FB15K	GQE + NRN	13.17	13.65	13.56	5.37	24.37	32.49	6.31	15.61	4.44	3.19
	Q2B + NRN	13.58	14.07	14.65	4.69	25.25	33.09	6.46	16.00	5.12	3.38
	Q2P + NRN	14.12	14.49	14.75	7.33	24.90	31.66	7.77	16.32	5.80	4.44
	ComplEx	16.81	16.45	23.44	10.19	29.03	**39.52**	10.58	17.40	1.86	2.44
	CHOREOGRAPH	**22.61**	**22.24**	**29.25**	**16.68**	**30.72**	36.16	**18.90**	**21.35**	**12.85**	**14.97**
DB15K	GQE + NRN	8.73	8.48	4.97	2.07	16.59	31.16	2.84	9.87	0.90	1.48
	Q2B + NRN	9.52	9.24	5.44	1.87	18.53	33.19	3.18	11.23	1.04	1.65
	Q2P + NRN	9.35	9.12	4.43	2.44	18.86	32.32	2.90	11.02	1.12	1.69
	ComplEx	11.18	10.32	9.68	4.34	21.36	**35.41**	5.96	10.61	0.77	1.29
	CHOREOGRAPH	**15.81**	**15.02**	**14.50**	**11.79**	**22.36**	34.35	**14.69**	**15.22**	**3.83**	**9.72**
YAGO15K	GQE + NRN	6.69	6.87	4.78	2.00	12.74	19.99	2.49	8.66	1.25	1.59
	Q2B + NRN	7.58	7.84	5.30	1.90	14.18	22.77	3.06	10.20	1.52	1.72
	Q2P + NRN	6.20	6.21	4.59	2.61	12.24	15.02	2.95	8.72	1.51	1.96
	ComplEx	9.19	9.00	8.85	4.61	16.17	26.96	5.02	10.01	0.83	1.07
	CHOREOGRAPH	**21.20**	**20.84**	**15.85**	**13.92**	**21.05**	**29.64**	**34.50**	**39.97**	**4.76**	**9.91**

Table 2 reports results on NRN benchmark queries. CHOREOGRAPH achieves the highest MRR on seven of eight structures for FB15K and DB15K and on all eight for YAGO15K. It also improves both averages on every dataset. Compared with the ComplEx backbone alone, CHOREOGRAPH improves both averages on all three datasets, indicating that its gains are not solely due to backbone choice. ComplEx remains better on 3i for FB15K and DB15K. Full results with standard deviations for both benchmarks are provided in Appendix D.5. Because aggregate MRR mixes structural and numerical queries, Appendix D.1 reports results restricted to queries containing at least one numerical operator. Appendix D.2 provides an FB15K example that traces learned numerical transport to a held-out target on the attribute grid. We also evaluate hyperparameter sensitivity in Appendix D.4.

5.2 Set–Scalar Arithmetic

We evaluate the restricted set–scalar arithmetic executor in Eq. (9) on five query structures. Table 3 reports the results. The 2pb structure is the most challenging on all three datasets, while direct 2b and 3b arithmetic perform comparably overall. Appendix B.4 gives query generation, splits, filtering, and uncertainty-estimation details.

Table 3: CHOREOGRAPH MRR (%) on set–scalar arithmetic queries.

Dataset	Avg_M	2b	3b	bp	pb	2pb
FB15K	5.27 ± 0.13	7.20 ± 0.69	8.36 ± 1.99	5.55 ± 2.16	2.91 ± 1.38	2.31 ± 0.38
DB15K	4.18 ± 1.03	5.13 ± 1.56	4.89 ± 1.25	5.10 ± 3.38	4.24 ± 1.38	1.54 ± 0.58
YAGO15K	10.14 ± 0.75	13.81 ± 1.29	15.93 ± 2.86	10.08 ± 2.26	6.78 ± 1.32	4.09 ± 1.18

5.3 Core Ablations

Table 4 evaluates Gaussian literal initialization, quantile binning, learned RPKs, and numerical message passing. The first two ablations use one-hot histograms and uniform-width bins, respectively. The RPK ablation replaces relation-conditioned transport with uniform transport, while the message-passing ablation disables propagated slot states. Message passing has the largest effect, reducing MRR by 3.72–17.20 points when removed. Learned RPKs improve MRR over uniform transport on all four datasets. Gaussian initialization and quantile binning have smaller effects overall. On DB15K, removing the latter has little effect, while removing the former slightly improves MRR.

Table 4: Core ablations measured by macro MRR (%) over seven LitCQD numerical-filter query structures. Results are mean $\pm$ sample standard deviation across three runs.

Variant	FB15K-237	FB15K	DB15K	YAGO15K
CHOREOGRAPH	**17.30 ± 1.43**	**21.64 ± 0.20**	10.43 ± 0.81	**5.79 ± 0.81**
w/o Gaussian Literal Initialization	17.07 ± 1.25	20.79 ± 0.33	**10.47 ± 0.86**	5.67 ± 0.90
w/o Quantile Binning	15.45 ± 1.83	19.87 ± 0.43	10.23 ± 0.96	5.09 ± 0.89
w/o RPK	13.57 ± 0.69	16.35 ± 0.63	9.79 ± 0.79	4.30 ± 0.71
w/o Message Passing	4.25 ± 0.67	4.44 ± 0.35	3.66 ± 0.16	2.07 ± 0.39

5.4 Efficiency

Table 5 reports efficiency on FB15K. CHOREOGRAPH fits RPKs independently and precomputes numerical slot states with two-layer message passing rather than training operator-specific neural networks. Its method-specific components add 78.0 seconds beyond 1,067.7 seconds of ComplEx training. Under this timing protocol, total training is $3.3\times$ faster than LitCQD and $22.4\times$ faster than NRN. On the matched inference workload, CHOREOGRAPH and LitCQD take 1.94 and 1.79 ms per query, respectively. NRN uses a different workload, so its latency is only a system-level reference. Appendix C.5 gives timing, memory, and parameter details, while Appendix B.5 gives the training complexity.

Table 5: Training time, inference latency, and model size on FB15K. LitCQD and CHOREOGRAPH use the same inference workload, whereas NRN uses a different workload and is shown only as a system-level reference. A plus sign separates backbone and method-specific components.

Method	Training time (s)	Inference/query (ms)	Model size (M)
LitCQD	3,826.0	**1.79**	35.28 + 0.03
NRN	25,685.0	11.25	**11.45**
CHOREOGRAPH	**1,067.7 + 78.0**	1.94	35.28 + 0.17

6 Conclusion

CHOREOGRAPH represents numerical states as explicit histograms on ordered attribute grids. Quantile binning, relation-conditioned transport, and message passing support numerical query execution in histogram space. Across benchmark queries from LitCQD and NRN, the method outperforms the corresponding baselines on most query structures. Histogram bins correspond to intervals in the original units, while RPKs expose relation-conditioned transport on the attribute grids. RPK fitting uses independent ridge solves followed by simplex projection and incurs little training overhead. These results highlight the value of explicit numerical representations for accurate and traceable query answering with efficient numerical-component fitting.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript drafting and editing, literature search and summarization, and writing and revising general-purpose code. The authors formulated the research problem, developed the core method, and made the final decisions regarding the mathematical claims, proofs, experimental design, and interpretation of results. The authors reviewed all AI-assisted text and code, tested the resulting implementation, and verified the cited literature, data provenance, and reported results. The authors take full responsibility for the paper's text, claims, code, and experimental findings.

ETHICS STATEMENT

This work studies numerical reasoning on established knowledge-graph benchmarks and does not involve collecting new data from human participants. KG reasoning systems can inherit omissions and biases from their source graphs. Explicit histograms make numerical distributions inspectable but do not eliminate these data-related risks.

REPRODUCIBILITY STATEMENT

Attribute grids, RPKs, and calibration quantities are estimated from training data, and model parameters are fitted using training data only. Validation data are used for hyperparameter selection and early stopping. The Method section defines the numerical representation, RPK fitting procedure, message passing, and query operators; Appendix B.2 provides derivations and formal properties. Appendix C.1 reports the shared hyperparameters, while Appendices C.2–C.4 document the evaluation protocol, query-label mapping, and baseline inclusion criteria. The timing procedure, parameter accounting, and training complexity are described in Appendices C.5 and B.5. Anonymous source code is available at `https://anonymous.4open.science/r/CHOREOGRAPH-B450/`. Its `README.md` documents environment setup and required external benchmark artifacts, and `scripts/README.md` lists entry points for split-aware data preparation, backbone training on the training graph only, and evaluation. The repository also provides the histogram, RPK, message-passing, query-adapter, and metric implementations together with unit tests; benchmark data and pretrained checkpoints must be obtained separately as described in the repository.

REFERENCES

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *International Conference on Learning Representations*, 2021.

Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu Song. Query2Particles: Knowledge graph reasoning with particle embeddings. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 2703–2714, 2022.

Jiaxin Bai, Chen Luo, Zheng Li, Qingyu Yin, Bing Yin, and Yangqiu Song. Knowledge graph reasoning over entities and numerical values. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 57–68, 2023.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26, pp. 2787–2795, 2013.

Caglar Demir, Michel Wiebesiek, Renzhong Lu, Axel-Cyrille Ngonga Ngomo, and Stefan Heindorf. LitCQD: Multi-hop reasoning in incomplete knowledge graphs with numeric literals. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 617–633, 2023.

James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 194–202. Morgan Kaufmann, 1995.

Alberto García-Durán and Mathias Niepert. KBLRN: End-to-end learning of knowledge base representations with latent, relational, and numerical features. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 372–381, 2018.

William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems*, volume 31, pp. 2030–2041, 2018.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. Incorporating literals into knowledge graph embeddings. In *The Semantic Web – ISWC 2019*, pp. 347–363, 2019.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 2863–2872, 2018.

Ye Liu, Hui Li, Alberto García-Durán, Mathias Niepert, Daniel Oñoro-Rubio, and David S. Rosenblum. MMKG: Multi-modal knowledge graphs. In *The Semantic Web – ESWC 2019*, pp. 459–474, 2019.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

Xianshu Peng, Wei Wei, Kaihe Xu, and Dangyang Chen. CNEQ: Incorporating numbers into knowledge graph reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10029–10042, 2024.

Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2Box: Reasoning over knowledge graphs in vector space using box embeddings. In *International Conference on Learning Representations*, 2020.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019.

Dhanasekar Sundararaman, Shijing Si, Vivek Subramanian, Guoyin Wang, Devamanyu Hazarika, and Lawrence Carin. Methods for numeracy-preserving word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 4742–4753, 2020.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

Theo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 2071–2080, 2016.

Yanrong Wu and Zhichun Wang. Knowledge graph embedding with numeric attributes of entities. In *Proceedings of the Third Workshop on Representation Learning for NLP*, pp. 132–136, 2018.

Jun Zhang, Haihong E, Tianyi Hu, Yifan Zhu, Meina Song, and Haoran Luo. Complex numerical reasoning with numerical semantic pre-training framework. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 15480–15514, 2025.

Ze Zhao, Bin Lu, Xiaoying Gan, Gu Tang, Luoyi Fu, and Xinbing Wang. ChainsFormer: Numerical reasoning on knowledge graphs from a chain perspective. In *Proceedings of the 2025 IEEE 41st International Conference on Data Engineering*, pp. 2963–2976, 2025.

The appendix is organized as follows: Appendix A summarizes the notation; Appendix B provides methodological and mathematical details; Appendix C documents the experimental setup and reproducibility protocol; Appendix D presents additional experimental analyses; and Appendix E discusses limitations.

A Notation Reference

Table 6 summarizes the notation used throughout the paper. Local summation indices and standard operators are defined where they first appear and are omitted here.

B Methodological and Mathematical Details

This appendix provides implementation and mathematical details of CHOREOGRAPH, including the structural backbone, formal properties of the numerical representation, attribute-existence estimation, arithmetic execution, and computational complexity.

B.1 Structural Backbone Details

An entity anchor e assigns $z_e = 0$ to itself and $z_{e'} = -\infty$ to every other entity $e' \neq e$. Let $\boldsymbol{c}_e, \boldsymbol{c}_r \in \mathbb{C}^d$ denote the ComplEx embeddings of entity e and relation r. Here Re extracts the real part, and the overline denotes complex conjugation. The ComplEx backbone (Trouillon et al., 2016) scores a triple and applies entity-space relational projection as

$$\phi_r(e_1, e_2) = \mathrm{Re}\langle \boldsymbol{c}_{e_1}, \boldsymbol{c}_r, \overline{\boldsymbol{c}_{e_2}} \rangle\,, \qquad [\mathrm{rp}_r(\boldsymbol{z})]_t = \max_{s \in \mathcal{E}} \{z_s + \phi_r(s,t)\}. \tag{7}$$

The implementation approximates this projection by retaining up to 512 top-scoring active source entities and batching candidate scoring. We train this backbone only on relational training triples. The backbone operates on entity states and does not replace numerical histograms during query execution.

B.2 Formal Properties of the Numerical Representation

Proposition 1 (independent RPK fitting). For a relation–attribute pair (r, a) supported in training, define

$$\boldsymbol{A}_{r,a} = \sum_{(s,t) \in \mathcal{D}_{r,a}} \boldsymbol{C}_s^\top \boldsymbol{C}_s + \lambda \boldsymbol{I}, \qquad \boldsymbol{b}_{r,a} = \sum_{(s,t) \in \mathcal{D}_{r,a}} \boldsymbol{C}_s^\top \boldsymbol{p}^0_{t,a}, \tag{8}$$

where $\boldsymbol{C}_s = \boldsymbol{C}(\boldsymbol{p}^0_{s,a})$. If $\lambda > 0$, the unconstrained ridge stage in Eq. (3) has the unique solution $\widetilde{\boldsymbol{k}}_{r,a} = \boldsymbol{A}_{r,a}^{-1} \boldsymbol{b}_{r,a}$. Euclidean projection then gives the unique fitted kernel $\boldsymbol{k}_{r,a} = \Pi_{\Delta_K}(\widetilde{\boldsymbol{k}}_{r,a})$. Each (r, a) pair can be fitted independently.

Justification. When $\lambda > 0$, the ridge Hessian $2\boldsymbol{A}_{r,a}$ is positive definite. Its stationary point is therefore the unique minimizer. The probability simplex is nonempty, closed, and convex, so Euclidean projection onto it is also unique. Given the initialized histograms and fixed hyperparameters, both stages use only the training paths in $\mathcal{D}_{r,a}$. RPK fitting consequently needs neither query supervision nor gradients through another kernel. CHOREOGRAPH uses this two-stage estimator. In general, it differs from minimizing the ridge objective directly under the constraint $\boldsymbol{k} \in \Delta_K$.

Proposition 2 (closure of nonempty histogram operators). For a nonnegative vector $\boldsymbol{v}$ with $\|\boldsymbol{v}\|_1 > 0$, define $\mathcal{N}(\boldsymbol{v}) = \boldsymbol{v}/\|\boldsymbol{v}\|_1$. If every input histogram lies in Δ_B, then zero-padded convolution with a simplex RPK, elementwise product intersection, and elementwise maximum union, whenever their unnormalized output has positive total mass, return an element of Δ_B after applying $\mathcal{N}$.

Justification. Convolution, products, and maxima preserve nonnegativity. Truncating a zero-padded convolution can remove boundary mass but cannot create negative mass. Any nonzero output therefore becomes a valid histogram after normalization. The closure statement applies only to nonempty outputs; a zero-mass intersection causes the corresponding channel to be omitted and may therefore yield an empty numerical state.

Table 6: Notation used in this paper.

Symbol	Definition
$\mathcal{G}$	Numerical knowledge graph
$\mathcal{E}, \mathcal{R}, \mathcal{A}, \mathcal{N}$	Sets of entities, directed relations, numerical attributes, and numerical literals
$\mathcal{T}_{\mathrm{rel}}, \mathcal{T}_{\mathrm{num}}$	Relational and numerical triple sets
$(e_1, r, e_2),\ (e, a, n)$	Relational and numerical triples; (e, a) is a numerical slot
$(\cdot)^{\mathrm{tr}}$	Quantity restricted to the training graph
$(a, \eta_a, \boldsymbol{p}_a, \mathcal{B}_a)$	One attribute channel
$\mathcal{S},\ \omega_e$	Active entity set and normalized entity weight in attribute projection
$\mathcal{E}_a^+$	Entities observed with attribute a in training
$B,\ \mathcal{B}_a,\ b_{a,j}$	Number of bins, attribute grid for a, and its jth boundary
$\boldsymbol{p}_{e,a}^0,\ \boldsymbol{p}_{e,a}$	Anchor and inferred conditional histograms of slot (e, a)
$\eta_{e,a}$	Attribute-existence score of slot (e, a)
$\boldsymbol{g}_a(x),\ \sigma$	Gaussian histogram for literal x and its smoothing width in grid coordinates
$K,\ \boldsymbol{k}_{r,a}$	RPK width and relation projection kernel for (r, a)
$\mathcal{P}_{r,a}$	Convolutional transport induced by $\boldsymbol{k}_{r,a}$
$\mathcal{D}_{r,a}$	One-hop training paths that support $\boldsymbol{k}_{r,a}$
$\boldsymbol{C}(\boldsymbol{p})$	Toeplitz convolution matrix built from histogram $\boldsymbol{p}$
$\lambda,\ \Pi_{\Delta_K}$	Ridge coefficient and Euclidean projection onto the probability simplex
$\ell,\ L,\ \boldsymbol{m},\ H,\ \alpha$	Message-passing layer index, number of layers, numerical message, entropy, and aggregation weight
$\rho_{r,a}$	Training-path reliability of relation–attribute pair (r, a)
$\boldsymbol{\psi}(e),\ \boldsymbol{\theta}_a,\ \beta_a,\ \xi_a$	ComplEx-derived classifier feature vector, weight vector, bias, and prevalence-calibration offset
$\boldsymbol{z},\ \mathcal{H}$	Entity score vector and numerical state
$\boldsymbol{c}_e, \boldsymbol{c}_r \in \mathbb{C}^d;\ \phi_r(e_1, e_2)$	ComplEx embeddings and relational-triple score
$\mathrm{nv}_a(x)$	Numerical-literal operator for literal x on attribute a
$\mathrm{rp}, \mathrm{ap}, \mathrm{rap}$	Entity-space relational, attribute, and reverse attribute projection
$\mathrm{lt}, \mathrm{gt}, \mathrm{eq}$	Less-than, greater-than, and equality operators
$\mathrm{np}_f, \mathrm{arith}_f$	Unary numerical transform and binary arithmetic pushforward induced by f
$\mathcal{I}, \mathcal{U}$	Intersection and union operators
$s_{\mathrm{rap}},\ s_{\mathrm{num}}$	Reverse-attribute-projection and numerical-decoding scores

Discrete mass transport and grid alignment. Histogram entries are discrete bin masses rather than samples of a persistent continuous density. Repeated quantile boundaries are retained, so zero-width bins may occur.

For arithmetic, the executor transports these masses using representative scalar values as described in Eq. (6). For unary scaling, it interpolates the normalized mass vector in grid coordinates and renormalizes the result.

Grid alignment for set operations uses interval-overlap resampling. For a positive-width source bin $I_j = [b_j, b_{j+1}]$ and target bin $J_k = [c_k, c_{k+1}]$, source mass p_j contributes

$$m_{j\to k} = p_j \frac{|I_j \cap J_k|}{b_{j+1} - b_j}.$$

Here, $|I_j \cap J_k|$ denotes the length of the overlap. A zero-width source bin is instead treated as a point mass at its repeated boundary. Zero-width target bins receive no interval-overlap mass, and mass outside the target range is assigned to the nearest endpoint bin. This resampling preserves total probability mass up to floating-point precision, and any nonzero output is normalized.

Lemma 1 (Invariance of Gaussian literal initialization to positive affine transformations). Let $g(n) = cn+d$ with $c > 0$. Recomputing the attribute grid after applying g to all training literals gives boundaries $g(b_{a,j})$. Empirical ranks and bin membership remain unchanged. For a positive-width bin, the within-bin coordinate is also unchanged because $(g(n) - g(b_{a,j}))/(g(b_{a,j+1}) - g(b_{a,j})) = (n - b_{a,j})/(b_{a,j+1} - b_{a,j})$. For a zero-width bin, the midpoint convention is unchanged because positive affine transformations preserve ties and bin indices. Positive affine unit conversions therefore leave Gaussian literal initialization unchanged. General strictly increasing transformations preserve sample order, although linearly interpolated empirical boundaries need not transform exactly.

B.3 Attribute-Existence Construction

The attribute-existence model estimates whether an attribute is present without altering its conditional histogram. We define RPK reliability from the number of supporting training paths: $\rho_{r,a} = \log(|\mathcal{D}_{r,a}| + 2)/\log(\max_{r',a'} |\mathcal{D}_{r',a'}| + 2)$.

Let $o_{e,a} = \mathbb{1}[e \in \mathcal{E}_a^+]$ denote whether slot (e, a) is observed in training, and initialize $g_{e,a}^{(0)} = o_{e,a}$. For $\ell \in \{1, 2\}$, we compute a reliability-weighted noisy-OR over training edges and their reciprocals:

$$\widetilde{g}_{t,a}^{(\ell)} = 1 - \prod_{(s,r,t)\in\mathcal{T}_{\mathrm{rel}}^{\mathrm{tr},\pm}} \left(1 - g_{s,a}^{(\ell-1)} \rho_{r,a}\right),$$

followed by the observed-slot clamp

$$g_{t,a}^{(\ell)} = \begin{cases} 1, & o_{t,a} = 1, \\ \widetilde{g}_{t,a}^{(\ell)}, & \text{otherwise.} \end{cases}$$

Thus inferred first-layer existence scores can serve as sources in the second layer, while observed slots retain a graph-based existence score of one throughout propagation.

In parallel, a logistic classifier uses features derived from frozen ComplEx entity embeddings. Let $\boldsymbol{\psi}(e)$ be the ℓ_2-normalized concatenation of the real and imaginary parts of the frozen ComplEx entity embedding. The classifier is defined by $\eta_{e,a}^{\mathrm{cls}} = \mathrm{sigmoid}(\boldsymbol{\theta}_a^\top \boldsymbol{\psi}(e) + \beta_a + \xi_a)$. We fit offset ξ_a on training data so that the mean predicted score matches the Laplace-smoothed prevalence $(|\mathcal{E}_a^+| + 1)/(|\mathcal{E}| + 2)$. The final score is $\eta_{e,a} = \eta_{e,a}^{\mathrm{cls}} g_{e,a}^{(2)}$. Supported pairs use their fitted RPKs. Unsupported pairs use the uniform kernel $k_{r,a,j} = 1/K$ and receive the minimum positive reliability under this log-shifted rule rather than being blocked from existence propagation.

B.4 Set–Scalar Arithmetic Execution

We generate 100 queries for each of five structures. The first numerical operand has one to nine answers. Each later operand is either a same-attribute singleton literal or an attribute projection with a singleton answer. The same-attribute restriction concerns operand provenance; arithmetic treats the decoded values as scalars and does not infer derived physical units. The symbol b denotes set–scalar arithmetic, and p denotes projection. The structures $2\mathrm{b}$ and $3\mathrm{b}$ apply arithmetic to two and

three operands, respectively. The bp structure projects after arithmetic, pb contains one projected input, and $2\mathrm{pb}$ contains projected inputs on both sides.

For an m-operand arithmetic query, we represent operand t by attribute channel $(a, \eta_t, \boldsymbol{p}_t, \mathcal{B}_a)$. The first operand remains a scored set, while each later operand is either an explicit numerical literal or a singleton-target attribute projection; both are represented as histogram states. Given candidate literals $\mathcal{N}_{\mathrm{cand}}$, we score the first operand as $w(x) \propto \eta_1 \langle \boldsymbol{p}_1, \boldsymbol{g}_a(x) \rangle$. For each later operand $t \geq 2$, let $s_t(x) \propto \eta_t \langle \boldsymbol{p}_t, \boldsymbol{g}_a(x) \rangle$ denote its candidate score, and decode $c_t = \arg\max_{x \in \mathcal{N}_{\mathrm{cand}}} s_t(x)$. For $f \in \{+, -, \times, \div\}$, the output is

$$p_{\mathrm{out}}(y) \propto \sum_{x \in \mathcal{N}_{\mathrm{cand}}} w(x)\, \delta(y - f(\cdots f(f(x, c_2), c_3) \cdots, c_m)) \,. \tag{9}$$

Masses assigned to identical output values are summed before discretization onto a newly constructed output grid spanning the finite derived values.

For each run, the KG is independently split into training, validation, and test partitions, and queries are generated from the corresponding cumulative graph views using the same run-specific random seed. Hard answers are defined as answers reachable in the cumulative test graph but not in the cumulative validation graph, and duplicate serialized queries are discarded. For $3\mathrm{b}$, each of the four arithmetic operators and each of the four combinations of later operand sources occurs in 25 queries; later attribute projections are restricted to same-attribute singleton answers. Undefined divisions and non-finite arithmetic outputs are rejected. To quantify uncertainty due to query sampling, we report 95% query-bootstrap confidence intervals computed by the percentile method from 10,000 bootstrap resamples, using the whole query as the resampling unit. Table 7 reports the resulting confidence intervals for 2b and 3b.

Table 7: 95% query-bootstrap confidence intervals for MRR (%) on direct set–scalar arithmetic queries.

Dataset	2b	3b
FB15K	[5.11, 9.54]	[6.02, 10.99]
DB15K	[3.38, 7.13]	[3.15, 6.85]
YAGO15K	[10.69, 17.19]	[12.76, 19.20]

B.5 Computational Complexity

We analyze the cost of fitting the numerical components and precomputing numerical states after training ComplEx. Table 5 reports backbone and CHOREOGRAPH-specific training separately. Constructing the attribute grids requires sorting the training literals for each attribute. Across all attributes, this costs at most $O(|\mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}| \log |\mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}|)$; initializing their B-bin Gaussian histograms adds $O(|\mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}|B)$.

RPK supervision is sparse. A relational triple $(e_1, r, e_2) \in \mathcal{T}^{\mathrm{tr}}_{\mathrm{rel}}$ contributes a training path only for attributes observed at both endpoints. The number of path contributions is therefore $\sum_{(e_1,r,e_2) \in \mathcal{T}^{\mathrm{tr}}_{\mathrm{rel}}} |\{a : e_1, e_2 \in \mathcal{E}^+_a\}|$, which is upper-bounded by $|\mathcal{T}^{\mathrm{tr}}_{\mathrm{rel}}||\mathcal{A}|$. For each path, the $B \times K$ Toeplitz design in Eq. (3) contributes to a $K \times K$ Gram matrix in $O(BK^2)$ time. Solving one $K \times K$ system costs $O(K^3)$. At most $|\mathcal{R}||\mathcal{A}|$ relation–attribute pairs are supported. The worst-case RPK training cost is therefore

$$O\big(|\mathcal{T}^{\mathrm{tr}}_{\mathrm{rel}}||\mathcal{A}|BK^2 + |\mathcal{R}||\mathcal{A}|K^3\big) \,. \tag{10}$$

This bound is conservative. The endpoints of an edge usually share only a few numerical attributes, and unsupported (r, a) pairs require no solve. We also group repeated, bin-centered source and target histograms into sufficient statistics. This grouping avoids repeated path-level matrix products. Message passing precomputes numerical slot states for L layers. In the worst case, each relational edge carries a length-B histogram for each attribute, and convolution with a width-K RPK costs $O(BK)$. The resulting message-passing cost is $O(L|\mathcal{T}^{\mathrm{tr}}_{\mathrm{rel}}||\mathcal{A}|BK)$.

Finally, the attribute-existence classifier scores $|\mathcal{E}||\mathcal{A}|$ entity–attribute pairs using features derived from frozen d-dimensional ComplEx embeddings. Each full optimization step costs $O(|\mathcal{E}||\mathcal{A}|d)$.

The classifier is trained for at most 300 epochs with validation-based early stopping. Treating this epoch cap as a fixed constant and combining the terms gives the CHOREOGRAPH-specific training and state-precomputation bound

$$O\big(|\mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}| \log |\mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}| + |\mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}|B + |\mathcal{T}^{\mathrm{tr}}_{\mathrm{rel}}||\mathcal{A}|BK^2 + |\mathcal{R}||\mathcal{A}|K^3 + L|\mathcal{T}^{\mathrm{tr}}_{\mathrm{rel}}||\mathcal{A}|BK + |\mathcal{E}||\mathcal{A}|d\big)\,. \tag{11}$$

In our setting, B, K, and L are also treated as fixed constants. Because $|\mathcal{R}| \leq |\mathcal{T}^{\mathrm{tr}}_{\mathrm{rel}}|$, treating these quantities as constants reduces the bound to $O(|\mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}| \log |\mathcal{T}^{\mathrm{tr}}_{\mathrm{num}}| + |\mathcal{T}^{\mathrm{tr}}_{\mathrm{rel}}||\mathcal{A}| + |\mathcal{E}||\mathcal{A}|d)$. Numerical-component training therefore scales with the observed graph rather than the space of possible queries.

C Experimental Setup and Reproducibility

This appendix specifies the shared hyperparameter configuration, evaluation protocol, query-label mapping, baseline inclusion criteria, and efficiency accounting used in the experiments.

C.1 Hyperparameter Selection

We use one numerical configuration across all datasets and query types. The histogram size is $B = 256$, the RPK width is $K = 127$, the Gaussian width is $\sigma = 1$ bin, the ridge coefficient is $\lambda = 10^{-4}$, and the number of message-passing layers is $L = 2$. We selected these shared values using validation queries and did not tune them separately for each dataset. Attribute-grid boundaries, RPKs, relation reliabilities, and attribute-existence calibration are instead estimated from the corresponding training graph. These quantities are data-dependent estimates rather than validation-tuned hyperparameters. The ComplEx backbone uses embedding dimension $d = 1000$, N3 regularization coefficient 0.01, learning rate 0.1, 100 epochs, and batch size 1024. This matches the ComplEx configuration used for LitCQD. The attribute-existence classifier is trained with weighted binary cross-entropy using Adam with learning rate 10^{-2}. Each positive entity–attribute pair is accompanied by five uniformly sampled negative entities for the same attribute, and positive examples receive a loss weight 5. Training runs for at most 300 epochs with validation every 10 epochs and early stopping after eight consecutive validation checks without improvement. We use $\epsilon = 10^{-7}$ when clipping existence probabilities for numerical stability during calibration and noisy-OR computation. Appendix D.4 examines sensitivity to B, K, and L.

C.2 Evaluation Protocol Details

We use controlled partitions to prevent information leakage across training, validation, and test splits. Relational facts in both directions between the same entity pair are assigned to the same relational partition. Otherwise, a held-out relational fact could be exposed through its reverse-direction counterpart. Likewise, all values of a numerical slot are assigned to the same numerical partition so that a held-out slot is not partially observed through another value of the same entity–attribute pair. We construct attribute grids, RPK targets, and attribute-existence labels from the training partition and train structural backbones only on its relational triples. Because these controlled partitions define new training graphs, we retrain every evaluated baseline on the same partitions rather than reuse published checkpoints or reported scores. For each baseline, we use the best-performing configuration reported in the original paper when available; otherwise, we follow the configuration used in the authors' reported experiments or released defaults. No baseline configuration is selected using our test partitions. Following the LitCQD and NRN protocols (Demir et al., 2023; Bai et al., 2023), we rank hard answers after filtering answers reachable in the preceding graph partition.

For the ComplEx-only baseline reported in Tables 2, 10, and 13, we augment the relational graph with value nodes indexed by numerical value and unit. Numerical attribute facts become directed entity-to-value links with separate reverse links, while numerical predicates become directed links between value nodes. ComplEx embeds both entity and value nodes and learns these relations from the training graph. Consequently, attribute projection, reverse attribute projection, and numerical projection are evaluated as learned relational projections.

C.3 Evaluation Query Labels

Table 8 maps LitCQD query names to the cross-method reporting labels. NRN query names are unchanged.

Table 8: LitCQD query names and cross-method reporting labels.

Scope	Original → reporting label
Entity	1p → 1rp, 2p → 2rp, 3p → ppp, 2i → 2pi, 3i → 3p3i, ip → 2pip, pi → pppi, 2u → 2pu, up → 2pup
Numerical	{ai-eq, ai-lt, ai-gt} → nr (query-count-weighted), 2ai → 2ai, aip → nrp, pai → nrpi, au → au, 1ap → 1ap, 2ap → pa, 3ap → 2pa

C.4 Baseline Inclusion Criteria

We include a baseline only when its pipeline can be trained and evaluated end-to-end on the controlled splits. The CNEQ archive we examined omits the processed graphs, generated queries, and checkpoints required for this evaluation. The CNR-NST authors provided query and answer splits. However, the code release we examined lacks the reported checkpoints and a complete conversion and execution pipeline for the LitCQD queries. Our reconstruction produced substantially different results. We therefore discuss both methods as related work but do not report results from the incomplete reconstruction.

C.5 Efficiency Measurement Details

The timings in Table 5 exclude data loading, cache serialization, and other file I/O. All timing experiments were conducted on the same machine with a single NVIDIA RTX A6000 48 GB GPU and two AMD EPYC 7352 CPUs. LitCQD optimizes relational and attribute losses jointly, so its 3,826-second training time cannot be broken down by component. Its checkpoint contains 35.28M entity and relation parameters and 0.03M attribute-regression parameters.

For CHOREOGRAPH, we time each stage separately: 1,067.7 seconds for ComplEx training, 9.4 seconds for RPK fitting, 11.7 seconds for attribute-existence classifier fitting, and 56.9 seconds for two-layer message passing. These stages take 1,145.68 seconds in total. During two-layer message passing, peak host memory usage is 2.80 GiB, while peak CUDA allocated and reserved memory are 0.572 GiB and 0.875 GiB, respectively. The resulting compressed numerical-state cache occupies approximately 163 MiB. It stores 1,088 supported relation–attribute kernels of width 127, or 138,176 coefficients. Together with 30,015 classifier parameters, the method-specific component contains 0.17M parameters. Cached slot states and dense placeholders for unsupported pairs are not learned parameters. The 0.17M figure therefore refers only to the numerical component, not the full model.

LitCQD and CHOREOGRAPH inference use the same workload of 110,414 LitCQD queries. Their measured latencies are therefore directly comparable. NRN uses a different query workload, so its timing is a system-level reference rather than a matched latency comparison.

D Additional Experimental Analysis

This appendix reports per-structure results for numerical queries, a traceability example, quantile-binning and numerical-range analyses, hyperparameter sensitivity, and full benchmark results with standard deviations.

D.1 Queries Containing Numerical Operators

Tables 9 and 10 report per-structure results on queries containing at least one numerical operator. In the LitCQD notation, ai-eq, ai-lt, and ai-gt are single-attribute equality and order filters. The 2ai structure intersects two attribute filters, and nrp applies entity-space relational projection after

Table 9: MRR (%) on LitCQD numerical-filter queries.

Dataset	Method	ai-eq	ai-lt	ai-gt	2ai	nrp	nrpi	au
FB15K-237	LitCQD	4.34 ± 0.67	2.78 ± 1.11	2.84 ± 0.72	2.84 ± 1.47	4.69 ± 1.25	18.14 ± 1.92	0.97 ± 0.49
	CHOREOGRAPH	**15.83 ± 1.87**	**22.95 ± 1.31**	**19.32 ± 0.90**	**17.92 ± 3.00**	**7.49 ± 2.22**	**20.83 ± 2.61**	**16.75 ± 1.75**
FB15K	LitCQD	3.54 ± 0.84	3.18 ± 0.44	1.52 ± 0.88	3.99 ± 0.43	5.02 ± 0.48	19.88 ± 2.29	0.57 ± 0.10
	CHOREOGRAPH	**18.52 ± 2.41**	**30.23 ± 2.44**	**26.77 ± 0.11**	**20.69 ± 1.31**	**6.76 ± 0.48**	**27.13 ± 2.70**	**21.37 ± 0.78**
DB15K	LitCQD	1.02 ± 0.53	0.84 ± 0.58	0.36 ± 0.18	1.02 ± 0.65	2.95 ± 0.58	14.37 ± 1.84	0.19 ± 0.05
	CHOREOGRAPH	**6.65 ± 0.32**	**12.13 ± 1.59**	**11.27 ± 2.89**	**10.13 ± 2.80**	**4.02 ± 1.17**	**21.58 ± 1.28**	**7.20 ± 0.97**
YAGO15K	LitCQD	1.68 ± 0.54	0.40 ± 0.03	1.28 ± 0.47	0.59 ± 0.23	**2.00 ± 0.27**	**20.74 ± 0.45**	0.56 ± 0.26
	CHOREOGRAPH	**3.67 ± 1.42**	**7.53 ± 1.40**	**2.79 ± 0.49**	**2.15 ± 0.28**	1.59 ± 0.24	20.02 ± 4.15	**2.77 ± 0.71**

Table 10: MRR (%) on NRN benchmark queries containing at least one numerical operator.

Dataset	Method	Avg_M	Avg_W	1p	2p	2i	3i	pi	ip	2u	up
FB15K	GQE + NRN	12.71 ± 0.30	10.94 ± 0.25	1.16 ± 0.13	5.16 ± 0.57	28.47 ± 0.37	**40.60 ± 0.54**	5.39 ± 0.47	15.16 ± 0.39	3.06 ± 0.36	2.64 ± 0.24
	Q2B + NRN	12.75 ± 0.46	11.09 ± 0.40	1.22 ± 0.26	4.32 ± 0.56	28.59 ± 0.53	40.39 ± 0.64	5.71 ± 0.70	15.46 ± 0.97	3.47 ± 0.21	2.85 ± 0.31
	Q2P + NRN	12.89 ± 1.11	11.29 ± 1.03	0.90 ± 0.31	5.41 ± 0.62	**28.61 ± 1.41**	39.82 ± 2.70	6.32 ± 1.15	15.14 ± 2.16	3.90 ± 0.30	3.05 ± 0.50
	ComplEx	13.25 ± 0.37	11.19 ± 0.29	1.88 ± 0.23	8.02 ± 0.24	28.06 ± 0.98	40.00 ± 0.56	9.42 ± 0.63	15.27 ± 0.75	1.05 ± 0.06	2.31 ± 0.13
	CHOREOGRAPH	**19.19 ± 0.20**	**17.26 ± 0.38**	**24.14 ± 1.50**	**15.43 ± 0.45**	21.69 ± 0.37	30.18 ± 0.83	**17.68 ± 0.66**	**17.79 ± 0.28**	**11.96 ± 0.26**	**14.67 ± 1.63**
DB15K	GQE + NRN	7.58 ± 0.03	6.34 ± 0.04	0.97 ± 0.20	2.20 ± 0.35	14.23 ± 0.17	28.24 ± 0.75	3.74 ± 0.43	9.45 ± 0.29	0.53 ± 0.07	1.29 ± 0.12
	Q2B + NRN	8.24 ± 0.32	6.88 ± 0.27	1.55 ± 0.28	1.96 ± 0.27	15.62 ± 0.96	29.66 ± 1.08	4.24 ± 0.60	10.79 ± 0.70	0.64 ± 0.03	1.42 ± 0.12
	Q2P + NRN	7.67 ± 0.76	6.44 ± 0.65	0.96 ± 0.25	1.96 ± 0.20	14.49 ± 1.00	28.36 ± 2.49	3.65 ± 1.06	10.08 ± 0.83	0.63 ± 0.08	1.24 ± 0.22
	ComplEx	9.02 ± 0.11	7.23 ± 0.13	3.00 ± 0.34	4.05 ± 0.05	**17.47 ± 0.58**	**31.03 ± 0.48**	5.80 ± 0.35	9.04 ± 0.48	0.51 ± 0.02	1.23 ± 0.04
	CHOREOGRAPH	**14.46 ± 0.43**	**12.78 ± 0.29**	**14.74 ± 0.07**	**14.11 ± 0.28**	14.83 ± 1.22	27.87 ± 2.92	**16.23 ± 1.60**	**14.26 ± 0.48**	**3.59 ± 0.27**	**10.09 ± 0.22**
YAGO15K	GQE + NRN	9.97 ± 0.75	3.56 ± 0.26	1.16 ± 0.26	2.37 ± 0.33	24.92 ± 1.69	37.81 ± 3.38	2.89 ± 0.36	8.22 ± 0.34	0.96 ± 0.10	1.39 ± 0.33
	Q2B + NRN	10.53 ± 0.86	4.07 ± 0.32	1.34 ± 0.34	2.29 ± 0.17	26.52 ± 2.84	37.81 ± 2.69	3.62 ± 0.58	9.91 ± 0.77	1.21 ± 0.07	1.53 ± 0.30
	Q2P + NRN	10.36 ± 1.42	3.81 ± 0.82	0.99 ± 0.23	2.34 ± 0.40	26.60 ± 2.93	38.37 ± 3.92	3.29 ± 1.51	8.64 ± 2.50	1.15 ± 0.11	1.53 ± 0.34
	ComplEx	10.21 ± 1.31	3.99 ± 0.32	1.47 ± 0.19	4.33 ± 0.21	24.98 ± 3.84	34.81 ± 5.48	5.10 ± 0.54	9.34 ± 0.39	0.65 ± 0.06	1.03 ± 0.10
	CHOREOGRAPH	**31.34 ± 1.20**	**23.11 ± 0.72**	**38.08 ± 1.69**	**22.11 ± 1.77**	**32.95 ± 3.50**	**44.06 ± 4.42**	**43.98 ± 1.00**	**54.08 ± 4.25**	**4.89 ± 0.02**	**10.60 ± 1.39**

a filter. The nrpi structure intersects an entity-space relational branch with a filter, while au unions two attribute filters. On LitCQD queries, CHOREOGRAPH outperforms LitCQD on 26 of the 28 dataset–structure comparisons, with the only exceptions being nrp and nrpi on YAGO15K. On NRN queries, CHOREOGRAPH outperforms all NRN variants on all structures except 2i and 3i on FB15K and DB15K. Compared with the ComplEx backbone alone, CHOREOGRAPH improves both averages on all three datasets and all structures except 2i and 3i on FB15K and DB15K. In the full YAGO15K NRN benchmark, 17.1% of 1p queries, 72.3% of pi queries, and 56.3% of ip queries contain at least one numerical operator.

D.2 Traceability Example

Figure 2 shows an example of learned numerical transport on FB15K. The relational training graph contains the edge from *Poland* to *Podlaskie Voivodeship* through `first_level_divisions`. *Poland*'s area, 312,679, is observed in the numerical training split, whereas the target entity's area, 20,180, is held out in the test split. The corresponding `first_level_divisions–area` RPK is fitted from 87 one-hop training paths and assigns the most probability mass to a shift of -29 bins. Applying the kernel shifts the histogram mode from bin 239 to bin 210, adjacent to the held-out

target in bin 209. This example makes the learned numerical transport directly inspectable on the ordered grid.

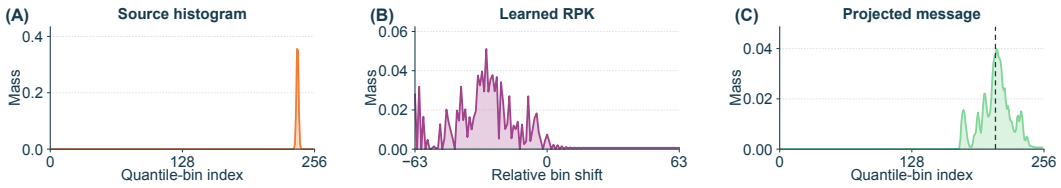

Figure 2: **Trace of a learned relation projection kernel on FB15K.** (A) Observed source histogram for *Poland*'s area. (B) Learned RPK for `first_level_divisions` and `area`. (C) Projected message after convolution. The projected mode lies adjacent to the held-out target bin (dashed line).

D.3 Quantile Binning and Numerical Range Coverage

Quantile-binning illustration. FB15K birth years span $[-383, 1999.9]$ in the training split. The central 99% lie in $[1625.7, 1990.4]$, and 91.3% lie between 1900 and 2000. Uniform-width bins therefore allocate much of their resolution to a sparsely populated historical range. FB15K longitude literals span $[-175.2°, 179.2°]$. Their distribution is multimodal. In particular, 54.1% lie between $-130°$ and $-60°$, while 26.0% lie between $-15°$ and $15°$. Uniform-width binning assigns the same resolution per degree to these dense and sparse regions. Figure 3 illustrates these distributions and the resulting quantile-bin occupancy.

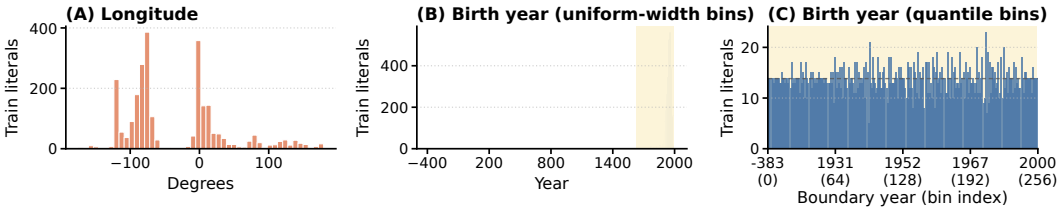

Figure 3: **Attribute-wise quantile binning allocates resolution according to the density of training literals.** (A) FB15K longitude literals have a multimodal distribution. (B) With 256 uniform-width bins, birth years occupy a narrow part of the observed range. (C) Mapping the same values to 256 quantile bins produces near-uniform occupancy. Each tick gives a bin index and its boundary in years; repeated years account for the remaining variation in occupancy. The yellow regions in (B) and (C) contain the central 99% of the birth-year training literals.

Quantile-bin boundaries use only numerical training triples. To measure endpoint clipping, Table 11 compares each held-out literal with the training minimum and maximum of its attribute. On every dataset, at least 98.45% of held-out literals lie within the fitted range. Every held-out attribute is also represented in training. We clip the remaining values to the nearest endpoint rather than discard them. The grid therefore uses no held-out values while retaining every query.

Table 11: Fraction of held-out literals within attribute ranges fitted to training literals.

Dataset	Held-out literals	In range (%)	Outside range, count (%)
FB15K-237	2,993	99.20	24 (0.80)
FB15K	3,009	98.80	36 (1.20)
DB15K	4,853	98.45	75 (1.55)
YAGO15K	2,361	99.92	2 (0.08)

D.4 Hyperparameter Sensitivity

Figure 4 varies the histogram size B, RPK width K, and number of message-passing layers L one at a time around the shared configuration in Appendix C.1. All other settings are fixed, and results are

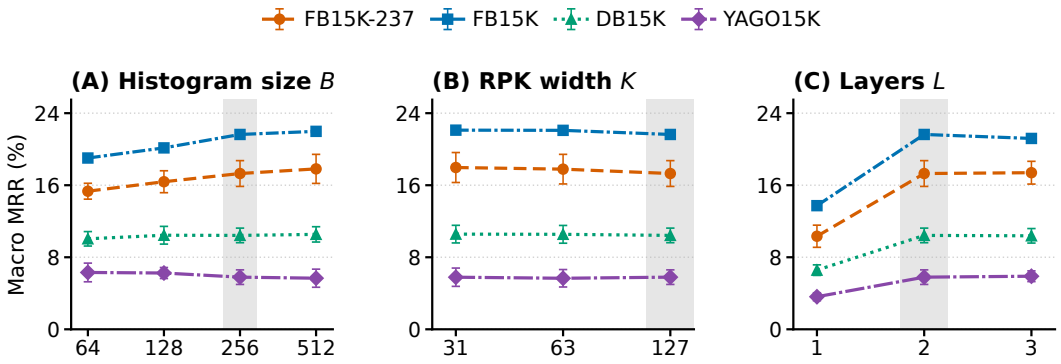

Figure 4: **Hyperparameter sensitivity across four datasets.** Macro MRR (%) over seven numerical-filter query structures as a function of (A) histogram size B, (B) RPK width K, and (C) message-passing layers L. Each panel varies one hyperparameter while holding the others fixed. Points show the mean across three KG splits; error bars show sample standard deviation. Gray bands mark the CHOREOGRAPH settings.

averaged over three KG splits. During training, these hyperparameters affect only grid construction, RPK fitting, and message passing, none of which involves iterative optimization: the ridge stage of Eq. (3) is closed-form, and message passing runs exactly L layers. The results therefore reflect the estimator rather than optimization convergence, and no setting requires retraining the ComplEx backbone or the attribute-existence classifier.

Histogram size. Larger B improves performance on both Freebase datasets with diminishing returns: doubling B from 128 to 256 adds 0.91 and 1.49 MRR points on FB15K-237 and FB15K, but doubling it to 512 adds only 0.51 and 0.34. The effect is small on DB15K and slightly favors coarser grids on YAGO15K; on both datasets, all settings lie within 0.64 points, comparable to the across-split standard deviations. In contrast, message passing, the Gram-matrix term of RPK fitting (Appendix B.5), the numerical-state cache, and most query-time histogram operations grow linearly with B, and the binary pushforward in Eq. (6) grows quadratically, so moving to $B = 512$ at least doubles these costs for at most 0.51 points.

RPK width. Narrower kernels yield modest gains on three datasets (at most 0.67 points) and barely affect YAGO15K. A narrower kernel has fewer coefficients, which lowers estimation variance, but the width also caps each transport step at $(K-1)/2$ bins, or about a fraction $(K-1)/(2B)$ of the attribute's training literals, since each quantile bin holds about $1/B$ of them. With $B = 256$, $K = 31$ allows at most 15 bins (about 6%) and cannot represent the -29-bin shift in Appendix D.2, whereas $K = 127$ allows 63 bins (about 25%). The gains therefore indicate that lower variance outweighs the lost range for these filter queries, not that narrow kernels suffice for every relation–attribute pair, so we keep $K = 127$ as a conservative default. Its cost is small: at $K = 127$, RPK fitting and message passing take 66.3 seconds on FB15K, under 6% of total training time (Appendix C.5).

Message-passing layers. Reducing message passing to one layer consistently hurts performance (by 2.17 to 7.91 points), while a third layer provides no consistent improvement over two (at most 0.45 points in either direction). One layer reaches only slots adjacent to an observed slot of the same attribute, whereas two layers also reach two-hop neighborhoods (Sec. 4.3). Farther evidence is attenuated: all anchors share the same Gaussian width, and convolution with a simplex kernel cannot lower entropy apart from boundary truncation, so messages relayed over more hops tend to be broader, and Eq. (4) down-weights them whenever a shorter-range message reaches the same slot. The cost, however, does not saturate: the set of active message sources never shrinks across layers, so a third layer adds at least half of the two-layer propagation work (56.9 seconds on FB15K).

The shared configuration was selected using validation queries and applied without dataset-specific tuning, so it need not maximize test performance for every dataset; it is within 0.67 points of the best setting on every dataset. We keep σ and λ fixed: removing the smoothing controlled by σ, as in

the one-hot variant of Table 4, changes MRR by at most 0.85 points, and λ mainly ensures a unique ridge solution (Appendix B.2), while the simplex projection constrains each kernel to a distribution over shifts. Moreover, this analysis covers LitCQD numerical-filter queries; it does not establish which setting would be best for NRN or other query structures.

D.5 Full Benchmark Results with Standard Deviations

Tables 12 and 13 report the full LitCQD and NRN benchmark results with standard deviations, respectively. The main paper reports only the corresponding mean values in Table 1 and 2 for readability.

Table 12: Full results on LitCQD benchmark queries. Candidate-ranking columns report MRR (%); scalar-prediction columns report MAE or MSE, multiplied by 10^2 for readability. Values are reported as mean ± standard deviation.

Dataset	Method	1rp	2rp	ppp	2pi	3p3i	2pip	pppi	2pu	2pup
FB15K-237	LitCQD	40.88 ± 0.34	14.92 ± 0.13	9.07 ± 0.63	33.63 ± 0.25	45.72 ± 0.29	23.26 ± 0.64	26.73 ± 0.80	15.02 ± 0.65	11.40 ± 0.36
	CHOREOGRAPH	**44.35 ± 0.36**	**18.74 ± 0.48**	**14.42 ± 0.32**	**37.76 ± 0.71**	**49.57 ± 0.23**	**25.26 ± 0.58**	**30.32 ± 0.91**	**18.69 ± 0.65**	**15.14 ± 0.44**
FB15K	LitCQD	44.56 ± 0.32	16.77 ± 0.35	10.32 ± 0.52	35.50 ± 0.71	47.72 ± 0.86	25.89 ± 0.38	31.11 ± 0.28	18.34 ± 0.28	11.59 ± 0.30
	CHOREOGRAPH	**47.47 ± 0.37**	**19.53 ± 0.64**	**15.78 ± 0.42**	**39.24 ± 0.60**	**50.83 ± 0.61**	**27.31 ± 0.27**	**33.71 ± 0.36**	**22.05 ± 0.07**	**14.98 ± 0.46**
DB15K	LitCQD	27.29 ± 0.43	8.87 ± 0.14	6.24 ± 0.49	28.96 ± 0.74	52.44 ± 0.63	15.11 ± 0.18	27.08 ± 0.59	5.38 ± 0.30	5.05 ± 0.23
	CHOREOGRAPH	**27.90 ± 0.44**	**10.10 ± 0.14**	**9.22 ± 0.62**	**37.62 ± 0.82**	**64.20 ± 0.56**	**20.67 ± 0.41**	**30.65 ± 0.85**	**5.99 ± 0.26**	**6.03 ± 0.16**
YAGO15K	LitCQD	24.21 ± 0.42	5.70 ± 0.19	4.04 ± 0.17	18.01 ± 0.16	36.28 ± 0.99	9.63 ± 0.65	14.91 ± 0.17	5.30 ± 0.10	4.05 ± 0.18
	CHOREOGRAPH	**24.39 ± 0.33**	**5.71 ± 0.17**	**6.14 ± 0.07**	**23.84 ± 0.03**	**41.65 ± 0.72**	**12.85 ± 0.48**	**17.77 ± 0.36**	**5.85 ± 0.19**	**4.45 ± 0.11**

Dataset	Method	nr	nrp	nrpi	1ap(MAE)	1ap(MSE)	pa(MAE)	pa(MSE)	2pa(MAE)	2pa(MSE)
FB15K-237	LitCQD	3.38 ± 0.69	4.69 ± 1.25	18.14 ± 1.92	**5.29 ± 0.19**	**1.19 ± 0.06**	3.68 ± 0.34	**0.70 ± 0.15**	4.90 ± 0.40	0.77 ± 0.15
	CHOREOGRAPH	**19.21 ± 0.57**	**7.49 ± 2.22**	**20.83 ± 2.61**	5.82 ± 0.09	1.77 ± 0.04	**3.51 ± 0.81**	0.77 ± 0.40	**4.00 ± 0.71**	**0.63 ± 0.19**
FB15K	LitCQD	2.79 ± 0.73	5.02 ± 0.48	19.88 ± 2.29	**5.20 ± 0.16**	**1.21 ± 0.07**	2.77 ± 0.66	**0.34 ± 0.13**	6.18 ± 0.26	1.36 ± 0.08
	CHOREOGRAPH	**24.82 ± 0.40**	**6.76 ± 0.48**	**27.13 ± 2.70**	5.97 ± 0.16	2.00 ± 0.11	**2.67 ± 0.44**	0.39 ± 0.12	**3.89 ± 0.65**	**0.75 ± 0.21**
DB15K	LitCQD	0.74 ± 0.30	2.95 ± 0.58	14.37 ± 1.84	**3.53 ± 0.03**	**0.97 ± 0.05**	**2.63 ± 0.83**	**0.37 ± 0.29**	3.62 ± 0.74	0.42 ± 0.13
	CHOREOGRAPH	**10.02 ± 1.44**	**4.02 ± 1.17**	**21.58 ± 1.28**	4.07 ± 0.11	1.49 ± 0.09	2.75 ± 0.88	0.45 ± 0.32	**2.80 ± 0.29**	**0.32 ± 0.04**
YAGO15K	LitCQD	1.57 ± 0.50	**2.00 ± 0.27**	**20.74 ± 0.45**	**3.27 ± 0.10**	**0.52 ± 0.07**	2.48 ± 0.49	0.54 ± 0.44	4.04 ± 0.43	0.66 ± 0.16
	CHOREOGRAPH	**3.86 ± 1.20**	1.59 ± 0.24	20.02 ± 4.15	3.60 ± 0.19	0.67 ± 0.02	**2.28 ± 0.40**	**0.43 ± 0.35**	**2.57 ± 0.57**	**0.45 ± 0.22**

E Limitations

CHOREOGRAPH has several limitations. Unsupported relation–attribute pairs cannot learn informative numerical transport, and values outside a training-derived attribute grid are clipped to its endpoints. Performance also degrades on some query structures that intersect multiple uncertain numerical branches. Finally, entity-space relational projection depends on a ComplEx backbone, so structural reasoning quality remains limited by the underlying KGE model.

Table 13: Full MRR (%) results on NRN benchmark queries. Values are mean ± sample standard deviation over three KG splits.

Dataset	Method	Avg_M	Avg_W	1p	2p	2i	3i	pi	ip	2u	up
FB15K	GQE + NRN	13.17 ± 0.70	13.65 ± 0.69	13.56 ± 1.17	5.37 ± 0.65	24.37 ± 0.17	32.49 ± 0.46	6.31 ± 0.23	15.61 ± 0.93	4.44 ± 1.32	3.19 ± 0.71
	Q2B + NRN	13.58 ± 0.80	14.07 ± 0.80	14.65 ± 1.13	4.69 ± 0.51	25.25 ± 0.52	33.09 ± 0.32	6.46 ± 0.73	16.00 ± 1.51	5.12 ± 1.41	3.38 ± 0.68
	Q2P + NRN	14.12 ± 2.21	14.49 ± 2.37	14.75 ± 1.69	7.33 ± 1.09	24.90 ± 3.12	31.66 ± 4.05	7.77 ± 1.31	16.32 ± 3.05	5.80 ± 1.99	4.44 ± 1.65
	ComplEx	16.81 ± 0.33	16.45 ± 0.35	23.44 ± 0.73	10.19 ± 0.26	29.03 ± 0.75	**39.52 ± 0.40**	10.58 ± 0.48	17.40 ± 0.66	1.86 ± 0.07	2.44 ± 0.12
	CHOREOGRAPH	**22.61 ± 0.10**	**22.24 ± 0.20**	**29.25 ± 0.54**	**16.68 ± 0.15**	**30.72 ± 0.19**	36.16 ± 0.24	**18.90 ± 0.42**	**21.35 ± 0.42**	**12.85 ± 0.28**	**14.97 ± 1.38**
DB15K	GQE + NRN	8.73 ± 0.39	8.48 ± 0.33	4.97 ± 0.78	2.07 ± 0.33	16.59 ± 0.52	31.16 ± 1.60	2.84 ± 0.93	9.87 ± 0.48	0.90 ± 0.41	1.48 ± 0.27
	Q2B + NRN	9.52 ± 0.68	9.24 ± 0.61	5.44 ± 0.77	1.87 ± 0.18	18.53 ± 1.06	33.19 ± 2.38	3.18 ± 0.97	11.23 ± 1.36	1.04 ± 0.42	1.65 ± 0.35
	Q2P + NRN	9.35 ± 1.52	9.12 ± 1.45	4.43 ± 1.29	2.44 ± 0.56	18.86 ± 2.08	32.32 ± 5.27	2.90 ± 0.32	11.02 ± 1.98	1.12 ± 0.56	1.69 ± 0.72
	ComplEx	11.18 ± 0.21	10.32 ± 0.21	9.68 ± 0.20	4.34 ± 0.14	21.36 ± 0.43	**35.41 ± 0.19**	5.96 ± 0.39	10.61 ± 0.47	0.77 ± 0.06	1.29 ± 0.02
	CHOREOGRAPH	**15.81 ± 0.21**	**15.02 ± 0.11**	**14.50 ± 0.69**	**11.79 ± 0.25**	**22.36 ± 0.48**	34.35 ± 1.34	**14.69 ± 1.01**	**15.22 ± 0.32**	**3.83 ± 0.22**	**9.72 ± 0.19**
YAGO15K	GQE + NRN	6.69 ± 0.41	6.87 ± 0.37	4.78 ± 0.69	2.00 ± 0.30	12.74 ± 0.35	19.99 ± 0.75	2.49 ± 0.31	8.66 ± 0.75	1.25 ± 0.46	1.59 ± 0.61
	Q2B + NRN	7.58 ± 0.34	7.84 ± 0.32	5.30 ± 0.50	1.90 ± 0.12	14.18 ± 0.14	22.77 ± 0.42	3.06 ± 0.07	10.20 ± 1.22	1.52 ± 0.45	1.72 ± 0.55
	Q2P + NRN	6.20 ± 2.34	6.21 ± 2.56	4.59 ± 0.88	2.61 ± 0.58	12.24 ± 3.36	15.02 ± 8.07	2.95 ± 0.79	8.72 ± 3.55	1.51 ± 0.58	1.96 ± 0.92
	ComplEx	9.19 ± 0.20	9.00 ± 0.18	8.85 ± 0.37	4.61 ± 0.20	16.17 ± 0.23	26.96 ± 0.49	5.02 ± 0.37	10.01 ± 0.34	0.83 ± 0.05	1.07 ± 0.09
	CHOREOGRAPH	**21.20 ± 0.27**	**20.84 ± 0.20**	**15.85 ± 0.17**	**13.92 ± 1.16**	**21.05 ± 0.09**	**29.64 ± 0.42**	**34.50 ± 0.38**	**39.97 ± 0.54**	**4.76 ± 0.02**	**9.91 ± 1.33**