Monotonic Path Likelihood for Learning Full-Order Probabilities

Anonymous authors
Paper under double-blind review

Abstract

Learning score distributions from ranking observations requires linking item-level uncertainty to the probability of the observed order. Existing objectives often rely on restricted distribution families or cutpoint-based lower bounds. Learning flexible distributions through complete ordering events therefore calls for efficient probability evaluation and differentiation. We introduce Monotonic Path Likelihood (MPL), which addresses this need through a shared bin-mass representation for continuous parametric and nonparametric discrete distributions. Under conditional independence, strictly ordered bin assignments form monotonic paths. Aggregating their probability masses through shared prefixes gives an exact grid-event likelihood in $\mathcal{O}(LT)$ time for L items and T bins, enabling end-to-end optimization. Our analysis establishes monotonicity under nested grid refinement and continuous approximation error bounds, and characterizes the learning signal through the path posterior. Experiments on MultiDigit MNIST, MSLR-WEB30K, Istella, and multimodal score distribution recovery show improvements in long-list ranking and distribution recovery, with competitive performance on real-world learning-to-rank benchmarks.

1 Introduction

Learning to rank (LTR) is a fundamental task in information retrieval and recommendation, where a model learns from observed relevance or preferences to order new items (Liu, 2009; Rendle et al., 2009). To capture uncertainty in these judgments, a model can represent each item's score as a distribution whose scale and shape describe how the score may vary. A complete-order likelihood connects these distributions to ranking supervision by measuring the probability that jointly sampled scores follow the observed order. Using this likelihood as a training objective calls for efficient probability evaluation and differentiation through the score model.

Existing ranking methods construct tractable training objectives in several ways. RankNet and LambdaRank learn scoring functions through pairwise comparisons and metric-aware gradients (Burges et al., 2005; 2006). NeuralSort, SoftSort, and differentiable sorting networks smooth the sorting operation to support training with position- or list-level losses (Grover et al., 2019; Prillo & Eisenschlos, 2020; Petersen et al., 2021). ListMLE directly optimizes a complete-permutation likelihood through the Plackett–Luce (PL) sequential-choice factorization (Xia et al., 2008; Plackett, 1975). Under a random utility interpretation, this closed form corresponds to independent Gumbel scores with a common scale (Yellott, 1977); Appendix B.1 gives the derivation. Learning item-specific scales, tails, or multiple modes through a complete-order likelihood requires evaluating the ordering probabilities induced by more general score distributions.

For distributional ranking, the Full-Order Bound (FOB) learns score distributions using the probability mass of a single ordered rectangle within the complete-order event (Tang et al., 2026). Its coverage depends on cutpoints, which the Safe-Region Gradient Ascent (SRGA) solver optimizes before each model update; the solver's concavity guarantee relies on log-concave densities. To evaluate the complete event, independent-utility models express ordering likelihoods as nested integrals (Azari Soufiani et al., 2012). Baker (2020) evaluates these integrals by cumulative numerical integration, with cost linear in the list length and grid size, and derives likelihood derivatives for parameter fitting.

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

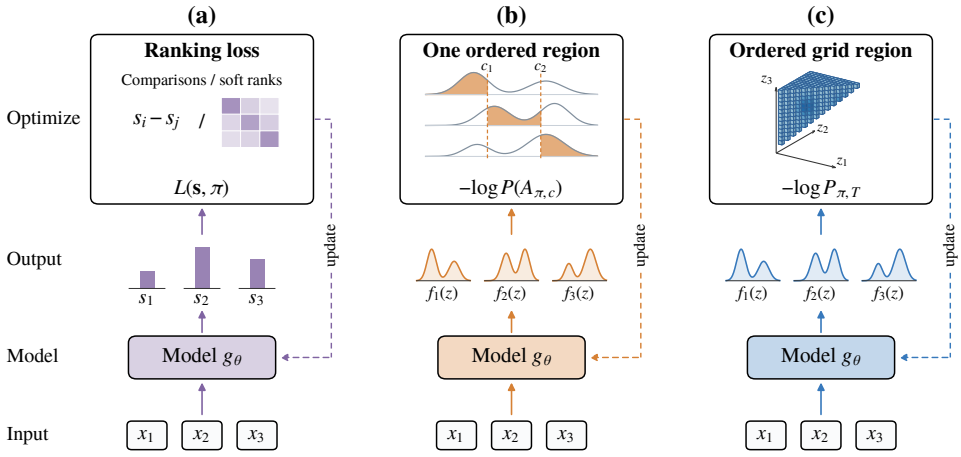

Figure 1: Ranking objectives for target order $x_1 \prec x_2 \prec x_3$ ($\pi = (1, 2, 3)$). **(a)** Scalar scores trained with ranking losses. **(b)** FOB maximizes the mass of $\mathcal{A}_{\pi,c} = \{z_1 \leq c_1 < z_2 \leq c_2 < z_3\}$, with cutpoints $c = (c_1, c_2)$, $c_1 < c_2$. **(c)** MPL maximizes $P_{\pi,T}$, the total mass of strictly ordered cells on a shared T-bin grid, shown on a uniform grid over $[0, 1]^3$.

Building on this cumulative computation principle, we introduce **Monotonic Path Likelihood (MPL)**, a framework for learning score distributions through complete ordering events. A shared grid represents continuous parametric and nonparametric discrete distributions through their bin probability masses, allowing both to use the same learning objective. Under conditional independence, strictly ordered bin assignments form monotonic paths. Aggregating their probability masses through shared prefixes gives an exact grid-event likelihood in $\mathcal{O}(LT)$ time for L items and T bins, enabling direct end-to-end optimization of the score model. The framework also handles tied labels by marginalizing over the internal orders of relevance blocks. Figure 1 summarizes how scalar-score and distributional objectives use ranking supervision.

Our main contributions are:

- A **framework for learning score distributions from full orders** that unifies continuous parametric and nonparametric discrete representations, with exact grid-event likelihood evaluation in $\boldsymbol{\mathcal{O}(LT)}$ time for strict total orders and $\boldsymbol{\mathcal{O}(LT^2)}$ marginalization for general ordered relevance blocks.
- **Monotonicity and approximation error guarantees** connecting grid-event probabilities to continuous ordering probabilities, together with a characterization of how posterior path probabilities guide learning.
- An **empirical evaluation** of ranking quality, generalization across list lengths, training cost, and distribution recovery on image ranking, real-world LTR, and multimodal recovery tasks.

2 Related Work

From pairwise preferences to listwise objectives. RankNet learns from pairwise score differences, while LambdaRank weights updates by their effect on retrieval metrics (Burges et al., 2005; 2006). ListNet and ListMLE incorporate list-level structure through probability objectives, with ListMLE using the PL sequential-choice factorization for a complete-permutation likelihood (Cao et al., 2007; Xia et al., 2008). Bayesian PL models parameter uncertainty, and partitioned-preference extensions handle grouped labels without enumerating within-group permutations (Guiver & Snelson, 2009; Ma et al., 2021).

Differentiable sorting. Continuous sorting relaxations let positional losses transmit gradients to input scores. NeuralSort constructs soft rank assignments, optimal-transport methods softly match items to positions, SoftSort approximates the argsort operator, and DiffSort relaxes comparison-and-swap operations (Grover et al., 2019; Cuturi et al., 2019; Prillo & Eisenschlos, 2020; Petersen et al.,

2021). PiRank and NeuralNDCG apply these ideas to scalable ranking and NDCG optimization (Swezey et al., 2021; Pobrotyn & Białobrzeski, 2021). Later work improves monotonicity, swap accuracy, and optimization using curvature information (Petersen et al., 2022; Kim et al., 2024; Petersen et al., 2024). LapSum uses Laplace CDF sums and their inverse to unify differentiable ranking, sorting, top-k selection, and permutations (Struski et al., 2025). SoftRank uses score-induced rank distributions to construct smooth ranking-metric objectives (Taylor et al., 2008).

Ordering probabilities and dynamic programming. Computing and differentiating ordering probabilities is central to likelihood-based learning of score distributions. Azari Soufiani et al. (2012) express permutation likelihoods as nested integrals and develop Monte Carlo EM for estimation. Baker (2020) evaluates ranking probabilities using cumulative numerical integration in $\mathcal{O}(LT)$ time, with quadrature error analysis and likelihood derivatives for parameter fitting. MPL shares this cumulative computation principle and develops a bin-mass formulation for learning score distributions from ranking supervision. It gives exact grid-event probabilities, supports posterior supervision, and enables approximation analysis through event inclusion under grid refinement. It also marginalizes ordered relevance blocks on the grid. Related order-statistics recurrences compute joint CDFs of sorted values (von Schroeder & Dickhaus, 2020); CTC, Soft-DTW, and differentiable dynamic programming aggregate paths through shared states (Graves et al., 2006; Cuturi & Blondel, 2017; Mensch & Blondel, 2018). The single-rectangle FOB objective retains a subset of the continuous ordering event and requires inner cutpoint optimization (Tang et al., 2026).

3 Method

3.1 Score distributions and the target event

Our goal is to learn score distributions that assign high probability to observed rankings. We first define this objective for continuous scores and distinct labels, then introduce a grid representation that also supports directly learned discrete distributions.

Consider L items with inputs $x_{1:L}$ and labels $y_{1:L}$. Let $[L] = \{1, \dots, L\}$ and let Π_L denote its permutations. The target order $\pi = (\pi_1, \dots, \pi_L) \in \Pi_L$ satisfies $y_{\pi_1} < \cdots < y_{\pi_L}$. The model specifies an absolutely continuous density f_j for each item's latent score Z_j. We suppress dependence on the inputs and shared parameters θ. Under conditional independence given $x_{1:L}$, the joint score density is $\prod_j f_j(z_j)$, and integrating it over all score configurations consistent with π gives the complete-order likelihood.

Definition 3.1 (Continuous full-order probability)**.** *The continuous ordering event associated with permutation π is*

$$\mathcal{E}_\pi = \left\{ z_{1:L} \in \mathbb{R}^L : z_{\pi_1} < z_{\pi_2} < \cdots < z_{\pi_L} \right\}. \tag{1}$$

Its probability and negative log-likelihood are

$$P_\pi^\star = \mathbb{P}(Z_{1:L} \in \mathcal{E}_\pi) = \int_{\mathcal{E}_\pi} \prod_{j=1}^{L} f_j(z_j)\, \mathrm{d}z_{1:L} \in [0, 1], \tag{2}$$

$$\mathcal{L}_\pi^\star = -\log P_\pi^\star \in [0, +\infty]. \tag{3}$$

Minimizing $\mathcal{L}_\pi^\star$ trains the score distributions to assign high probability to the observed order. Optimizing this objective for flexible distributions requires evaluating the ordering probability and propagating its gradient to the score model.

3.2 Monotonic Path Likelihood on a shared grid

We obtain a tractable approximation by restricting the continuous ordering event to strictly ordered bin assignments on a shared grid. The probability of this event depends only on each item's bin masses, providing a common objective for different score distributions. Partition the real line into T bins $B_t = (b_{t-1}, b_t]$, with $-\infty = b_0 < \cdots < b_T = +\infty$, and write $[T] = \{1, \dots, T\}$. Each item j is represented by normalized bin probability masses

$$q_j(t) = \mathbb{P}(Z_j \in B_t), \qquad q_j(t) \geq 0, \qquad \sum_{t=1}^{T} q_j(t) = 1. \tag{4}$$

For a continuous cumulative distribution function (CDF) F_j, these masses are $q_j(t) = F_j(b_t) - F_j(b_{t-1})$. A nonparametric discrete head can learn them directly on T ordered states. Both representations use the same objective below, with T controlling resolution and computational cost.

Assigning item π_i to bin t_i guarantees the target order whenever $t_1 < \cdots < t_L$. Such an assignment traces a monotonic path through the items in target order, moving to a strictly higher bin at each step. Let $\mathcal{M}_{L,T} = \{(t_1, \ldots, t_L) \in [T]^L : t_1 < \cdots < t_L\}$ collect these paths. Under conditional independence, path $m = (t_1, \ldots, t_L)$ has probability mass $w_\pi(m) = \prod_{i=1}^{L} q_{\pi_i}(t_i)$. Summing over the disjoint path events defines MPL:

$$\begin{aligned} P_{\pi,T} &= \sum_{m \in \mathcal{M}_{L,T}} w_\pi(m) = \sum_{(t_1,\ldots,t_L) \in \mathcal{M}_{L,T}} \prod_{i=1}^{L} q_{\pi_i}(t_i), \\ \mathcal{L}_{\text{MPL}}(\pi) &= -\log P_{\pi,T}. \end{aligned} \tag{5}$$

We use $T \geq L$ to allow strictly ordered assignments. For independent discrete scores, $P_{\pi,T}$ is the exact probability of the target order. For continuous scores, it is the exact probability of strict bin order and a lower approximation to $P^\star_\pi$, since correctly ordered scores that share a bin are omitted.

Figure 1 illustrates these grid cells for three items and places the objective alongside scalar-score losses and FOB. Nested grid refinement can recover additional order-consistent mass, as formalized below. Appendix A compares the grid event with the FOB rectangle.

3.3 Shared-prefix computation

Directly summing the $\binom{T}{L}$ feasible paths becomes costly as the list and grid grow. Their shared prefixes allow us to reuse computation: all prefixes ending in the same bin admit the same next bins. Let $A_i(t)$ collect the probability mass of feasible assignments to the first i items in target order that end in bin t:

$$A_i(t) = \sum_{\substack{1 \leq t_1 < \cdots < t_{i-1} < t_i = t \\ t_1, \ldots, t_i \in [T]}} \prod_{k=1}^{i} q_{\pi_k}(t_k). \tag{6}$$

The cumulative mass $C_i(t)$ combines prefixes ending in any bin up to t:

$$C_i(t) = \sum_{u=1}^{t} A_i(u), \qquad C_i(0) = 0. \tag{7}$$

Placing item π_i in bin t extends every prefix ending before t, whose mass is $C_{i-1}(t-1)$. Multiplying by $q_{\pi_i}(t)$ and accumulating along the grid gives

$$\begin{aligned} A_1(t) &= q_{\pi_1}(t), && t \in [T], \\ C_1(t) &= C_1(t-1) + A_1(t), && t \in [T], \\ A_i(t) &= q_{\pi_i}(t) C_{i-1}(t-1), && i = 2, \ldots, L, \quad t \in [T], \\ C_i(t) &= C_i(t-1) + A_i(t), && i = 2, \ldots, L, \quad t \in [T], \end{aligned} \tag{8}$$

$$P_{\pi,T} = C_L(T). \tag{9}$$

Each feasible assignment has a unique final bin and preceding prefix, so the recurrence counts its mass exactly once. Scanning bins in increasing order updates each cumulative mass in constant time, giving one linear scan per item. Figure 2 illustrates how prefixes combine to extend a path.

Theorem 3.2 (Complexity for strict total orders)**.** *A strict permutation of length L with T bins can be evaluated in $\mathcal{O}(LT)$ time and $\mathcal{O}(T)$ streaming working memory. Storing all input masses and the states needed for backpropagation requires $\mathcal{O}(LT)$ memory.*

This recurrence supports differentiation through the bin masses to the score model. Appendix B.2 proves the complexity bounds, and Appendix A provides pseudocode and a worked update.

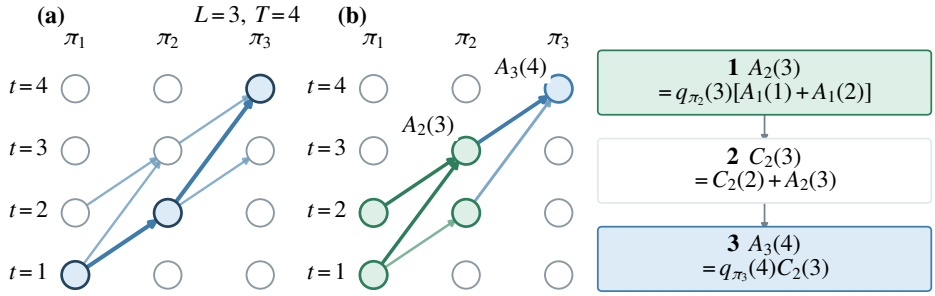

Figure 2: Feasible paths and the shared-prefix recurrence in MPL. Each column represents an item in the target permutation, and each row a bin index $t \in [T]$. **(a)** Light-blue edges show all feasible transitions; dark-blue edges mark path $m = (1, 2, 4)$. **(b)** Green edges enter $A_2(3)$ and blue edges enter $A_3(4)$. Steps 1–3 aggregate feasible predecessors, update cumulative prefix masses, and extend the next node, respectively.

3.4 Continuous approximation

The recurrence evaluates the grid likelihood exactly. For continuous score models, we now relate that likelihood to the full-order probability. The gap arises when correctly ordered scores occupy the same bin; refining the grid can separate them.

Let $\mathcal{G}$ denote a shared grid and $\mathcal{E}_{\pi,\mathcal{G}}$ its strict bin-order event, with probability $P_{\pi,\mathcal{G}} = P_{\pi,T}$. Write $\mathcal{G} \preceq \mathcal{G}'$ when each bin of $\mathcal{G}'$ lies within a bin of $\mathcal{G}$. For fixed score distributions, this nested refinement preserves every previously feasible assignment and may admit additional ones:

Theorem 3.3 (Monotonicity under grid refinement)**.** *For any nested partitions $\mathcal{G} \preceq \mathcal{G}'$, we have*

$$\mathcal{E}_{\pi,\mathcal{G}} \subseteq \mathcal{E}_{\pi,\mathcal{G}'} \subseteq \mathcal{E}_{\pi}, \qquad P_{\pi,\mathcal{G}} \leq P_{\pi,\mathcal{G}'} \leq P^{\star}_{\pi}. \tag{10}$$

The remaining error can be controlled through the grid's coverage and resolution. Let R specify the finite range $[-R, R]$ and h the maximum bin width within it, with an unbounded tail bin at each end. Denote the strict bin-order probability by $P_{\pi,h,R}$. For item j, let $r_j(R) = \mathbb{P}(|Z_j| > R)$ be its tail probability and $M_j = \|f_j\|_\infty$ its density bound.

Theorem 3.4 (Grid error for continuous probabilities)**.** *If $Z_1, \ldots, Z_L$ are conditionally independent and absolutely continuous with bounded densities, then*

$$0 \leq P^{\star}_{\pi} - P_{\pi,h,R} \leq \sum_{i=1}^{L} r_{\pi_i}(R) + h \sum_{i=1}^{L-1} \min\{M_{\pi_i}, M_{\pi_{i+1}}\}. \tag{11}$$

Consequently, $P_{\pi,h,R} \to P^{\star}_{\pi}$ as $R \to \infty$ and $h \to 0$. If the grids are also nested, convergence is monotonic.

The first term controls mass outside the finite range, and the second controls adjacent ordered scores sharing an interior bin. Increasing R and decreasing h therefore reduce the two sources of approximation error. Appendices B.3 and B.4 give the proofs. Appendix B.5 extends the analysis to negative log-likelihood error and parameter-gradient convergence under additional conditions.

3.5 Path posterior and learning signals

To understand how a list-level ordering objective trains each item's distribution, we examine which bin assignments explain the observed order. Each feasible path is one such explanation. For $P_{\pi,T} > 0$, normalizing its mass gives the posterior probability of that assignment conditioned on the target order:

$$\rho_{\pi}(m) = \frac{w_{\pi}(m)}{P_{\pi,T}} = \frac{\prod_{i=1}^{L} q_{\pi_i}(t_i)}{P_{\pi,T}}, \qquad m \in \mathcal{M}_{L,T}. \tag{12}$$

Thus $\rho_{\pi}(m) \geq 0$ and $\sum_{m \in \mathcal{M}_{L,T}} \rho_{\pi}(m) = 1$. Marginalizing this posterior over paths yields a distribution over bins for each item, providing the learning signal derived below.

Computing the posterior probability that item π_i occupies bin t requires both the prefixes reaching (i, t) and the suffixes leaving it. The forward mass $A_i(t)$ already includes $q_{\pi_i}(t)$, so we define a suffix mass $B_i(t)$ that excludes this factor:

$$B_L(t) = 1, \qquad t \in [T], \tag{13}$$

$$B_i(t) = \sum_{v=t+1}^{T} q_{\pi_{i+1}}(v) B_{i+1}(v), \qquad i = L-1, \dots, 1. \tag{14}$$

The boundary $B_L(t) = 1$ represents the empty suffix. For $i < L$, the sum includes all next bins $v > t$ and their feasible continuations.

Prefixes and suffixes concatenate uniquely at (i, t), so all complete paths through that state have total mass $A_i(t)B_i(t)$. The posterior occupancy of the state is

$$\gamma_i(t) = \mathbb{P}_{m \sim \rho_\pi}(t_i = t) = \frac{A_i(t)B_i(t)}{P_{\pi,T}}, \qquad \sum_{t=1}^{T} \gamma_i(t) = 1. \tag{15}$$

Each complete path visits one state in column i, so $\gamma_i(1{:}T)$ is a normalized posterior marginal.

For a discrete head parameterized by bin logits $a_j = (a_j(1), \dots, a_j(T)) \in \mathbb{R}^T$, the relationship between predictions and posterior occupancies is explicit. The predicted masses are

$$q_j(t) = \frac{\exp(a_j(t))}{\sum_{u=1}^{T} \exp(a_j(u))}. \tag{16}$$

The gradient with respect to these logits is the difference between the predicted bin distribution and its posterior conditioned on the entire target order:

Proposition 3.5 (Path-posterior gradient)**.** *If $P_{\pi,T} > 0$, then for every $i \in [L]$ and $t \in [T]$,*

$$\frac{\partial \mathcal{L}_{\text{MPL}}(\pi)}{\partial a_{\pi_i}(t)} = q_{\pi_i}(t) - \gamma_i(t). \tag{17}$$

The vector q_{π_i} is the predicted bin distribution, and γ_i describes how its bins participate in complete paths consistent with the order. Gradient descent increases logits where posterior occupancy exceeds predicted mass and decreases them where it falls below. Multiple separated modes can therefore contribute to the learning signal through their posterior probabilities. Appendix B.6 provides the derivation.

3.6 Tied labels

The construction so far assumes a strict target order. Relevance grades provide a coarser form of supervision: they order groups of items but leave each group's internal order unspecified. We accommodate this supervision by summing over bin assignments consistent with the order between groups.

Let $C_1, \dots, C_G$ partition the L items into nonempty blocks in increasing label order, and let $U_i \in [T]$ be item i's bin index with $\mathbb{P}(U_i = t) = q_i(t)$. A nonnegative integer d specifies the minimum bin-index gap between consecutive blocks. Setting $d = 1$ requires strict separation, while $d = 0$ allows a shared boundary bin. The resulting event and its probability are

$$\begin{aligned} \mathcal{S}_{\text{tie}}^{(d)} &= \left\{ (u_1, \dots, u_L) \in [T]^L \;\middle|\; \max_{i \in C_g} u_i + d \le \min_{j \in C_{g+1}} u_j, \quad g = 1, \dots, G-1 \right\}, \\ P_{\text{tie}}^{(d)} &= \mathbb{P}\Big((U_1, \dots, U_L) \in \mathcal{S}_{\text{tie}}^{(d)} \Big). \end{aligned} \tag{18}$$

A block's minimum and maximum bin indices determine which assignments in neighboring blocks are compatible with it. Under conditional independence, an exact recurrence aggregates all within-block assignments by these boundaries and connects consecutive blocks in $\mathcal{O}(LT^2)$ time. This marginalizes their internal orders without enumerating permutations; Appendix C.2 gives the construction and proof. For singleton blocks with $d = 1$, the event reduces to the strict-order setting, and the prefix recurrence above evaluates it in $\mathcal{O}(LT)$ time.

Our real-world LTR experiments use the weak discrete ordering event with $d = 0$. Sharing a boundary bin allows scores from adjacent blocks to occur in either order within that bin. The lower-bound guarantee in theorem 3.3 applies to strict bin order and therefore does not extend to this setting.

4 RESULTS

We evaluate long-list ranking on MultiDigit MNIST, retrieval with tied labels on MSLR-WEB30K and Istella, and multimodal score distribution recovery. Baselines are RankNet (Burges et al., 2005), LambdaRank (Burges et al., 2006), ListMLE (Xia et al., 2008), SoftSort (Prillo & Eisenschlos, 2020), DiffSort (Petersen et al., 2021), LapSum (Struski et al., 2025), and FOB (Tang et al., 2026). All methods share each task's encoder architecture and data construction; the multimodal experiment compares MPL and FOB with identical mixture parameterizations. Appendix D gives configurations, metric definitions, confidence intervals, and additional results.

4.1 MULTIDIGIT MNIST

Each item is a four-digit number image. To increase task difficulty, we reduce the resolution of each digit image by average pooling and concatenate four digits into a 14×56 number image. Lists are ordered by increasing integer value. We train and test at matching lengths $L \in \{4, 6, 8, 16, 32\}$, test unseen lengths after training only at $L_{\text{train}} = 8$, and train jointly on all five lengths. MPL uses $T = 256$ nonparametric discrete distributions and ranks by the predicted mean; FOB follows its original setup (Tang et al., 2026). We also evaluate FOB (nonparametric), which uses the same nonparametric score distribution head as MPL and interprets each bin mass as a uniform continuous density for SRGA cutpoint optimization. Each setting uses five seeds.

Table 1: Matched-length ranking on MultiDigit MNIST. Entries report exact match (element accuracy), both as percentages and with higher values preferred. Bold denotes the best point estimate at each test length.

	Exact match (Element accuracy), %, ↑				
Method	$L = 4$	$L = 6$	$L = 8$	$L = 16$	$L = 32$
RankNet	76.96 (87.51)	55.61 (81.66)	34.88 (75.85)	4.87 (64.58)	0.00 (46.58)
LambdaRank	74.31 (86.02)	57.56 (82.60)	38.49 (77.44)	5.73 (66.66)	0.00 (48.13)
ListMLE	79.24 (88.79)	57.43 (82.57)	37.24 (77.05)	3.49 (61.86)	0.01 (45.32)
SoftSort	79.27 (88.84)	56.92 (82.27)	43.29 (80.00)	4.15 (63.83)	0.01 (50.04)
LapSum	77.83 (87.97)	56.33 (82.00)	43.44 (80.02)	5.21 (65.05)	0.01 (49.28)
DiffSort	81.83 (90.15)	62.55 (84.65)	46.13 (80.93)	5.31 (66.10)	0.01 (51.23)
FOB	81.91 (90.16)	63.00 (84.76)	46.96 (81.01)	15.67 (74.42)	0.04 (50.08)
FOB (nonparametric)	91.70 (95.26)	84.59 (93.32)	75.34 (91.04)	43.03 (84.68)	4.71 (72.63)
MPL	**92.91** (**95.83**)	**87.22** (**94.23**)	**79.68** (**92.14**)	**54.88** (**86.61**)	**17.21** (**76.62**)

The nonparametric head substantially improves FOB: at $L = 16$, exact match rises from 15.67% to 43.03%. MPL reaches 54.88%, exceeding FOB (nonparametric) by 11.85 percentage points; at $L = 32$, the corresponding values are 17.21% and 4.71%. The gap is smaller at $L = 4$ (92.91% versus 91.70%). Thus, flexible score distributions account for substantial gains, while MPL retains an advantage over this rectangle-bound adaptation, especially on long lists. The shared head controls representation capacity; the probability events and optimization procedures differ, as detailed in Appendix D.3.

Generalization across lengths. Training only at $L_{\text{train}} = 8$, MPL has the highest exact-match point estimate at all four unseen lengths. At $L = 16, 32$, it achieves $49.49\%, 10.73\%$, compared with $39.05\%, 4.30\%$ for FOB (nonparametric). Under mixed-length training, the corresponding values are $54.39\%, 15.78\%$ and $43.83\%, 6.40\%$. The advantage over this adaptation extends to both cross-length generalization and mixed training. Appendix D.3 gives full results.

4.2 REAL-WORLD LTR: MSLR-WEB30K AND ISTELLA

Both datasets provide query-grouped documents and graded relevance labels. We use official WEB30K splits and hold out validation queries from the Istella training set. Equal-grade documents have no specified internal order. We evaluate Strict Kendall and pairwise accuracy across different

grades, and NDCG@10 and NDCG@20 for top-of-list quality. MPL and FOB predict logistic locations with the same fixed raw scale. MPL uses $T = 128$ bins and the tied-block objective with state gap $d = 0$, allowing adjacent relevance blocks to share a boundary state. Quality evaluation uses ten seeds per WEB30K fold and ten for Istella. Separate timing experiments measure complete optimization steps; Appendix D.4 gives the protocol.

Table 2: Real-world LTR test quality and full training-step time (mean ± 95% CI). Timing uses five independent seeds. Methods are ordered by increasing NDCG@10; bold denotes the best point estimate.

Dataset	Method	Metrics (%, ↑)				Full step time (ms, ↓)
		NDCG@10	NDCG@20	Strict Kendall	Pairwise acc.	
WEB30K	ListMLE	47.85 ± 0.10	53.16 ± 0.08	30.17 ± 0.15	66.74 ± 0.07	53.53 ± 6.79
	SoftSort	48.04 ± 0.08	53.43 ± 0.07	30.02 ± 0.11	66.66 ± 0.06	15.75 ± 0.92
	DiffSort	48.16 ± 0.12	53.47 ± 0.09	29.88 ± 0.09	66.60 ± 0.05	369.28 ± 43.04
	LapSum	48.44 ± 0.06	53.69 ± 0.04	30.62 ± 0.10	66.96 ± 0.05	218.51 ± 24.03
	RankNet	48.68 ± 0.06	54.06 ± 0.05	31.50 ± 0.07	67.40 ± 0.04	$\mathbf{8.90 \pm 0.20}$
	LambdaRank	49.12 ± 0.07	54.40 ± 0.06	30.33 ± 0.11	66.82 ± 0.05	9.87 ± 0.31
	MPL	49.34 ± 0.05	54.61 ± 0.04	$\mathbf{31.78 \pm 0.07}$	$\mathbf{67.54 \pm 0.04}$	261.05 ± 4.71
	FOB	$\mathbf{49.37 \pm 0.06}$	$\mathbf{54.63 \pm 0.06}$	31.18 ± 0.09	67.24 ± 0.04	680.54 ± 14.46
Istella	ListMLE	72.13 ± 0.49	73.96 ± 0.52	73.16 ± 1.31	91.52 ± 0.63	24.85 ± 2.94
	LapSum	73.14 ± 0.47	74.82 ± 0.52	73.50 ± 1.58	91.69 ± 0.79	80.09 ± 10.59
	SoftSort	74.17 ± 0.33	75.92 ± 0.36	76.62 ± 0.70	93.25 ± 0.33	10.72 ± 0.97
	DiffSort	74.46 ± 0.25	76.52 ± 0.22	79.47 ± 0.28	94.68 ± 0.04	100.97 ± 22.18
	LambdaRank	75.09 ± 0.25	77.09 ± 0.22	80.12 ± 0.33	95.01 ± 0.08	9.68 ± 0.26
	FOB	75.19 ± 0.22	76.97 ± 0.18	78.20 ± 0.57	94.04 ± 0.23	561.32 ± 19.14
	RankNet	75.98 ± 0.13	77.88 ± 0.12	80.87 ± 0.32	95.38 ± 0.05	$\mathbf{8.57 \pm 0.25}$
	MPL	$\mathbf{76.55 \pm 0.17}$	$\mathbf{78.38 \pm 0.15}$	$\mathbf{80.94 \pm 0.32}$	$\mathbf{95.41 \pm 0.05}$	227.97 ± 5.00

On WEB30K, MPL and FOB have close NDCG@10 (49.34% versus 49.37%) and NDCG@20 (54.61% versus 54.63%). MPL has the highest Strict Kendall (31.78%) and pairwise accuracy (67.54%) point estimates, indicating better overall order consistency at similar top-of-list quality. On Istella, it has the highest point estimate on all four quality metrics, with NDCG@10 exceeding FOB by 1.36 percentage points. Full training steps take 261.05 ms on WEB30K and 227.97 ms on Istella, respectively 38.4% and 40.6% of the FOB times.

4.3 Multimodal score distribution recovery

Identical scalar scores can correspond to distinct underlying distributions. To assess whether permutation supervision can recover these distributional differences, we construct three machine types with six-dimensional features. The types have similar linear scalar scores but distinct three-component Gaussian mixture score distributions. Lists contain $L = 6$ items, and training supervision consists of permutations induced by sampled scores. MPL and FOB share the encoder, mixture parameterization, initialization, data splits, and optimization settings across four seeds (Appendix D.5). Validation distributions determine affine score calibration; test ranking accuracies use the true mean-score order (Appendix D.5.5).

Table 3: Multimodal score distribution recovery (mean ± 95% Student-t CI). Score W1 uses score units; permutation-NLL regret uses nats per permutation; other metrics are proportions. Ranking accuracies use the true mean-score order. Bold denotes the best point estimate.

Method	Score W1 (score units, ↓)	Pair-prob. MAE (proportion, ↓)	Perm. NLL regret (nat/perm., ↓)	Exact match (proportion, ↑)	Element acc. (proportion, ↑)
FOB	2.459 ± 0.173	0.031 ± 0.019	0.227 ± 0.057	0.158 ± 0.162	0.528 ± 0.199
MPL	$\mathbf{1.670 \pm 0.058}$	$\mathbf{0.018 \pm 0.004}$	$\mathbf{0.003 \pm 0.036}$	$\mathbf{0.295 \pm 0.124}$	$\mathbf{0.675 \pm 0.094}$

MPL reduces Score W1 from 2.459 to 1.670, pair-probability MAE from 0.031 to 0.018, and permutation-NLL regret from 0.227 to 0.003. Permutation-NLL regret measures excess log loss relative to the ground-truth model, with both probabilities evaluated on finite grids (Appendix D.1.3). Exact match of the mean-score order increases from 0.158 to 0.295 and element accuracy from 0.528 to 0.675. These results show improvements in score distribution recovery, ordering probability estimation, and ranking accuracy.

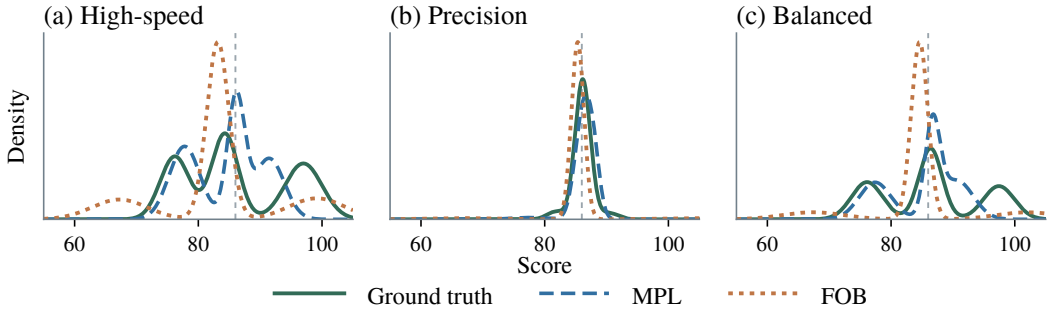

Figure 3: Representative densities for high-speed, precision, and balanced machines (left to right) with ground-truth means of 86.15, 86.11, and 86.26, respectively. Green solid, blue dashed, and orange dotted curves show the ground-truth, MPL, and FOB mixtures. Axes give score and probability density.

The three types have similar expected scores but distinct modal structures. Figure 3 shows that MPL preserves more type-dependent modes, while FOB concentrates on the central mode and spreads remaining mass across wider tails. The feature-space visualization and additional recovery metrics in Appendix D.5 support this observation.

4.4 Ablation Studies

Grid resolution. We assess sensitivity to the number of bins T on MultiDigit MNIST with training and test list length $L = 8$. Across five seeds, mean exact-match accuracy increases from 68.05% at $T = 32$ to 76.05% at $T = 64$, with smaller gains at higher resolutions: 78.59%, 79.59%, and 80.81% at $T = 128, 256, 512$, respectively. Element accuracy follows the same trend, increasing from 88.59% to 92.43% over the full range. These results show that coarse grids limit ranking performance in this setting, while additional bins yield smaller improvements beyond $T = 128$. Appendix D.3.1 reports the protocol and full results.

5 Conclusion

Monotonic Path Likelihood (MPL) learns score distributions from complete ordering events through a common bin-mass interface. Under conditional independence, exact strict grid-event likelihoods are evaluated in $\mathcal{O}(LT)$ time for end-to-end optimization. Our analysis connects grid refinement to continuous approximation and derives supervision from path posteriors. Experiments show improved long-list ranking over the nonparametric FOB control, competitive real-world retrieval, and improved synthetic distribution recovery.

The method assumes conditionally independent scores; approximation accuracy depends on grid resolution, and general tied-block inference costs $\mathcal{O}(LT^2)$. Ranking supervision alone does not uniquely identify absolute score locations, scales, or shapes; the model parameterization and shared parameters further constrain the learned distributions. Modeling correlated scores, adapting the grid, and developing faster partial-order recurrences are promising directions for extending the method's expressiveness and scalability.

AI USE STATEMENT

Generative AI tools were used for language polishing, assisting with code verification and checks of theoretical derivations, document formatting, and improving the clarity and presentation of figures. The research ideas, methodological design, theoretical contributions, and scientific conclusions were developed by the human authors. AI assistance was limited to the supporting tasks described above. The authors take full responsibility for the correctness and integrity of the final manuscript, including all AI-assisted text, code, theoretical claims, and figures.

REPRODUCIBILITY STATEMENT

Section 3 defines the model assumptions and likelihood objectives. Appendix A provides the FOB comparison and Algorithm 1; Appendix B supplies the theoretical proofs. Appendix C describes distribution parameterizations, tied-label inference, and variable-length loss normalization. Appendix D specifies architectures, metrics, and statistical conventions. The task protocols in Appendices D.3, D.4, and D.5 describe data construction, splits, and training. Appendix D.5.5 details the validation-based score calibration used for distribution recovery, and Appendix D.4 describes the training-step timing procedure.

REFERENCES

Hossein Azari Soufiani, David C. Parkes, and Lirong Xia. Random utility theory for social choice. In *Advances in Neural Information Processing Systems*, volume 25, 2012. URL https://papers.nips.cc/paper_files/paper/2012/hash/a512294422de868f8474d22344636f16-Abstract.html.

Rose Baker. New order-statistics-based ranking models and faster computation of outcome probabilities. *IMA Journal of Management Mathematics*, 31(1):33–48, 2020. doi: 10.1093/imaman/dpz001.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 89–96, 2005. doi: 10.1145/1102351.1102363.

Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems*, volume 19, 2006. URL https://proceedings.neurips.cc/paper/2006/hash/af44c4c56f385c43f2529f9b1b018f6a-Abstract.html.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 129–136, 2007. doi: 10.1145/1273496.1273513.

Marco Cuturi and Mathieu Blondel. Soft-DTW: A differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 894–903. PMLR, 2017. URL https://proceedings.mlr.press/v70/cuturi17a.html.

Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d8c24ca8f23c562a5600876ca2a550ce-Abstract.html.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 369–376, 2006. doi: 10.1145/1143844.1143891.

Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1eSS3CcKX.

John Guiver and Edward Snelson. Bayesian inference for Plackett–Luce ranking models. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 377–384, 2009. doi: 10.1145/1553374.1553423.

Jungtaek Kim, Jeongbeen Yoon, and Minsu Cho. Generalized neural sorting networks with error-free differentiable swap functions. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RLSWbk9kPw.

Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009. doi: 10.1561/1500000016.

Jiaqi Ma, Xinyang Yi, Weijing Tang, Zhe Zhao, Lichan Hong, Ed H. Chi, and Qiaozhu Mei. Learning-to-rank with partitioned preference: Fast estimation for the Plackett–Luce model. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 928–936. PMLR, 2021. URL https://proceedings.mlr.press/v130/ma21a.html.

Arthur Mensch and Mathieu Blondel. Differentiable dynamic programming for structured prediction and attention. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3462–3471. PMLR, 2018. URL https://proceedings.mlr.press/v80/mensch18a.html.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Differentiable sorting networks for scalable sorting and ranking supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8546–8555. PMLR, 2021. URL https://proceedings.mlr.press/v139/petersen21a.html.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Monotonic differentiable sorting networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=IcUWShptD7d.

Felix Petersen, Christian Borgelt, Tobias Sutter, Hilde Kuehne, Oliver Deussen, and Stefano Ermon. Newton losses: Using curvature information for learning with differentiable algorithms. In *Advances in Neural Information Processing Systems*, volume 37, pp. 26113–26135, 2024. doi: 10.52202/079017-0821. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2e102d937d094b7211c4d32ce1f1126c-Abstract-Conference.html.

Robin L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975. doi: 10.2307/2346567.

Przemysław Pobrotyn and Radosław Białobrzeski. NeuralNDCG: Direct optimisation of a ranking metric via differentiable relaxation of sorting. In *Proceedings of the ACM SIGIR Workshop on eCommerce*, 2021. URL https://sigir-ecom.github.io/ecom21Papers/paper11.pdf.

Sebastian Prillo and Julian Martin Eisenschlos. SoftSort: A continuous relaxation for the argsort operator. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7793–7802. PMLR, 2020. URL https://proceedings.mlr.press/v119/prillo20a.html.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pp. 452–461, 2009.

Łukasz Struski, Michal B. Bednarczyk, Igor T. Podolak, and Jacek Tabor. LapSum - one method to differentiate them all: Ranking, sorting and top-k selection. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 56990–57007. PMLR, 2025. URL https://proceedings.mlr.press/v267/struski25a.html.

Robin Swezey, Aditya Grover, Bruno Charron, and Stefano Ermon. PiRank: Scalable learning to rank via differentiable sorting. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/b5200c6107fc3d41d19a2b66835c3974-Abstract.html.

Yongxiang Tang, Chao Wang, Jincheng Lu, Yanhua Cheng, Xialong Liu, and Peng Jiang. Learning to rank by directly optimizing full-order probabilities. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=fch6yT64ZH.

Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. SoftRank: Optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pp. 77–86, 2008. doi: 10.1145/1341531.1341544.

Jonathan von Schroeder and Thorsten Dickhaus. Efficient calculation of the joint distribution of order statistics. *Computational Statistics & Data Analysis*, 144:106899, 2020. doi: 10.1016/j.csda.2019.106899.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1192–1199, 2008. doi: 10.1145/1390156.1390306.

John I. Yellott, Jr. The relationship between Luce's choice axiom, Thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology*, 15(2): 109–144, 1977. doi: 10.1016/0022-2496(77)90026-8.

A Method Details

This appendix compares the FOB and MPL probability regions and provides pseudocode and a worked recurrence update. The score model, bin masses, and path likelihood are defined in sections 3.1 and 3.2.

A.1 Comparison with the Full-Order Bound

For cutpoints $-\infty = c_0 \le c_1 \le \cdots \le c_L = +\infty$, the FOB rectangle is

$$\mathcal{A}_{\pi,c} = \left\{ z_{1:L} \in \mathbb{R}^L : c_{i-1} < z_{\pi_i} \le c_i \text{ for all } i \in [L] \right\}. \tag{19}$$

Every vector in this set satisfies the target order, so $\mathcal{A}_{\pi,c} \subseteq \mathcal{E}_\pi$. Conditional independence gives

$$\begin{aligned} P_{\mathrm{FOB}}(\pi, c) &= \mathbb{P}(Z_{1:L} \in \mathcal{A}_{\pi,c}) \\ &= \prod_{i=1}^{L} \left[F_{\pi_i}(c_i) - F_{\pi_i}(c_{i-1}) \right] \le P^\star_\pi. \end{aligned} \tag{20}$$

FOB retains the mass of one rectangle, while MPL sums all strictly ordered grid cells. Both regions lie within $\mathcal{E}_\pi$, but a finite MPL grid and an arbitrary FOB rectangle need not contain one another. Figure 4 compares these probability regions for three items. Appendix D.3 specifies the histogram adaptation used in the controlled MNIST comparison.

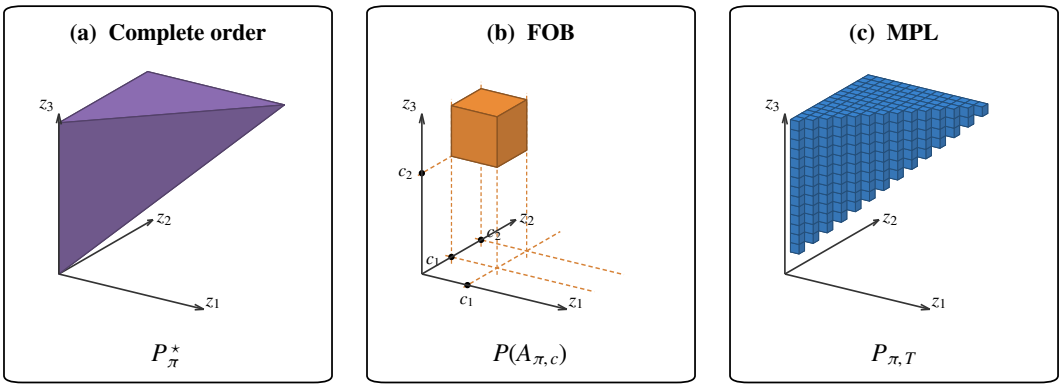

Figure 4: Probability regions for $x_1 \prec x_2 \prec x_3$ ($\pi = (1,2,3)$), shown within $[0,1]^3$. **(a)** The complete-order region $z_1 < z_2 < z_3$. **(b)** The FOB region with $c_1 = 1/3$ and $c_2 = 2/3$; dashed lines project its boundaries onto the coordinate planes. **(c)** The MPL region on a uniform grid with $T = 16$. Strict bin order places its lowest z_3 boundary at $2/T$.

A.2 Pseudocode and a worked update

Algorithm 1 implements equation (9) using two cumulative-mass buffers. It assumes $L \ge 1$ and $T \ge L$; padding and loss normalization are handled as described in Appendix C.3.

For the $L = 3, T = 4$ example in figure 2, assigning the second item to bin 3 extends either first-item bin 1 or 2:

$$A_2(3) = q_{\pi_2}(3)\left[q_{\pi_1}(1) + q_{\pi_1}(2)\right], \tag{21}$$

$$C_2(3) = A_2(2) + A_2(3), \tag{22}$$

$$A_3(4) = q_{\pi_3}(4) C_2(3). \tag{23}$$

The last update combines paths $(1,2,4)$, $(1,3,4)$, and $(2,3,4)$ through their shared prefix states. The remaining complete path, $(1,2,3)$, contributes through $A_3(3)$.

Appendix B.2 proves the time and memory bounds. Appendices B.3, B.4, and B.5 give the refinement, approximation-error, and gradient-convergence proofs. These results concern strict bin order; the $d = 0$ tied-block objective in Appendix C.2 permits shared boundary bins.

Algorithm 1: MPL prefix recurrence

Input: Target permutation $\pi \in \Pi_L$; bin probability mass matrix $q \in [0,1]^{L\times T}$ with $\sum_t q_j(t) = 1$
Output: Target permutation probability $P_{\pi,T}$

```
C(0) ← 0;
for t ← 1 to T do
    C(t) ← C(t − 1) + q_{π_1}(t);
for i ← 2 to L do
    C_new(0) ← 0;
    for t ← 1 to T do
        A(t) ← q_{π_i}(t) C(t − 1);
        C_new(t) ← C_new(t − 1) + A(t);
    C ← C_new;
return C(T);
```

B Supplementary Proofs

We first derive the Plackett–Luce special case, then prove the complexity, grid-refinement, and continuous-error results. We subsequently establish likelihood and gradient consistency and derive posterior occupancies and training gradients.

B.1 The Plackett–Luce special case

Proposition B.1 (Plackett–Luce special case)**.** *Suppose that*

$$Z_j = s_j + \epsilon_j, \qquad j \in [L], \tag{24}$$

where $\epsilon_1, \dots, \epsilon_L$ are independent Gumbel variables with location 0 and common scale $\beta > 0$,

$$\mathbb{P}(\epsilon_j \le u) = \exp\left[-\exp\left(-\frac{u}{\beta}\right)\right], \tag{25}$$

Then the continuous full-order probability of the ascending permutation π has the analytic form

$$P^\star_\pi = \mathbb{P}(Z_{\pi_1} < \cdots < Z_{\pi_L}) = \prod_{i=1}^{L} \frac{\exp(s_{\pi_i}/\beta)}{\sum_{k=1}^{i} \exp(s_{\pi_k}/\beta)}. \tag{26}$$

The scale β controls random utility fluctuations, and s_j/β is the PL logit. Setting $\beta = 1$ recovers the usual expression. Factor i is the probability that π_i has the largest utility in $\{\pi_1, \dots, \pi_i\}$. Reading the factors in the order $i = L, L-1, \dots, 1$ gives the sequential-choice factorization of PL/ListMLE. We prove this through a Gumbel-to-exponential-race transformation. With different item scales, distribution families, or multiple modes, definition 3.1 still defines the event, but its probability generally lacks this closed-form factorization.

Proof of proposition B.1. We transform the Gumbel random utilities into an exponential race. For each item, define

$$E_j = \exp\left(-\frac{\epsilon_j}{\beta}\right). \tag{27}$$

For any $e > 0$, the Gumbel CDF gives

$$\mathbb{P}(E_j > e) = \mathbb{P}(\epsilon_j < -\beta \log e) \tag{28}$$

$$= \exp\left[-\exp\left(-\frac{-\beta \log e}{\beta}\right)\right] = \exp(-e). \tag{29}$$

Thus E_j is exponentially distributed with rate 1. Since the transformation acts separately on each item, $E_1, \dots, E_L$ remain independent.

Next, let

$$T_j = \exp\left(-\frac{Z_j}{\beta}\right) = \exp\left(-\frac{s_j}{\beta}\right) E_j. \tag{30}$$

For any $t \geq 0$,

$$\mathbb{P}(T_j > t) = \mathbb{P}(E_j > \exp(s_j/\beta)t) \tag{31}$$

$$= \exp[-\exp(s_j/\beta)t]\,. \tag{32}$$

Hence T_j is an exponential random variable with rate

$$\lambda_j = \exp(s_j/\beta) \tag{33}$$

and the T_j are independent. The map $z \mapsto \exp(-z/\beta)$ is strictly decreasing, so

$$Z_{\pi_1} < \cdots < Z_{\pi_L} \quad \Longleftrightarrow \quad T_{\pi_L} < \cdots < T_{\pi_1}. \tag{34}$$

For any nonempty item set S and $j \in S$, independence and the exponential density imply

$$\mathbb{P}\left(T_j = \min_{k\in S} T_k\right) = \int_0^\infty \lambda_j e^{-\lambda_j t} \prod_{k\in S\setminus\{j\}} e^{-\lambda_k t}\,\mathrm{d}t \tag{35}$$

$$= \int_0^\infty \lambda_j \exp\left[-t\sum_{k\in S}\lambda_k\right]\,\mathrm{d}t \tag{36}$$

$$= \frac{\lambda_j}{\sum_{k\in S}\lambda_k}. \tag{37}$$

Exponential variables also satisfy the memoryless property:

$$\mathbb{P}(T_k - t > u \mid T_k > t) = e^{-\lambda_k u}, \qquad u \geq 0. \tag{38}$$

After conditioning on the first arrival time and its item, the remaining waiting times therefore form another independent exponential race with the original rates.

By equation (34), the first arrival must be π_L from the set $S_L = \{\pi_1, \ldots, \pi_L\}$. Removing π_L, the next arrival must be π_{L-1} from $S_{L-1} = \{\pi_1, \ldots, \pi_{L-1}\}$. Applying equation (37) successively until only π_1 remains gives

$$\mathbb{P}(Z_{\pi_1} < \cdots < Z_{\pi_L}) = \prod_{i=1}^{L} \frac{\lambda_{\pi_i}}{\sum_{k=1}^{i}\lambda_{\pi_k}} \tag{39}$$

$$= \prod_{i=1}^{L} \frac{\exp(s_{\pi_i}/\beta)}{\sum_{k=1}^{i}\exp(s_{\pi_k}/\beta)}. \tag{40}$$

This is equation (26). □

B.2 Computational complexity

Proof of theorem 3.2. We count the state operations in a forward evaluation. The first layer computes $A_1(t) = q_{\pi_1}(t)$ and updates $C_1(t) = C_1(t-1) + A_1(t)$ for $t = 1, \ldots, T$, visiting T states in $\mathcal{O}(T)$ time. Each subsequent layer $i = 2, \ldots, L$ also visits T states, with one multiplication

$$A_i(t) = q_{\pi_i}(t) C_{i-1}(t-1) \tag{41}$$

and one cumulative addition

$$C_i(t) = C_i(t-1) + A_i(t). \tag{42}$$

There are LT state visits and at most a constant multiple of that many arithmetic operations, giving total time

$$\mathcal{O}(T) + (L-1)\mathcal{O}(T) = \mathcal{O}(LT). \tag{43}$$

Computing layer i reads only $C_{i-1}(0{:}T)$ and writes $C_i(0{:}T)$ sequentially. Forward-only evaluation can therefore alternate between two buffers of length $T+1$, requiring auxiliary working memory

$$2(T+1) = \mathcal{O}(T). \tag{44}$$

The input bin probability mass matrix itself contains LT values. Standard reverse-mode automatic differentiation retains forward states from each layer; an explicit forward–backward implementation stores equivalent quantities. Both require a constant number of $L \times T$ arrays, giving $\mathcal{O}(LT)$ total memory including the input and backward states. □

B.3 Monotonicity under grid refinement

Proof of theorem 3.3. Let $\kappa_{\mathcal{G}}(z)$ denote the index of the bin containing z on grid $\mathcal{G}$, and define $\kappa_{\mathcal{G}'}$ analogously for the refined grid. Because $\mathcal{G}'$ refines $\mathcal{G}$, each fine bin lies in exactly one coarse bin, and refinement preserves the order of bins on the real line.

First, we show $\mathcal{E}_{\pi,\mathcal{G}} \subseteq \mathcal{E}_{\pi,\mathcal{G}'}$. Take any $z_{1:L} \in \mathcal{E}_{\pi,\mathcal{G}}$. At each adjacent position i, the coarse indices satisfy

$$\kappa_{\mathcal{G}}(z_{\pi_i}) < \kappa_{\mathcal{G}}(z_{\pi_{i+1}}). \tag{45}$$

The entire coarse bin containing z_{π_i} thus precedes the coarse bin containing $z_{\pi_{i+1}}$. Their fine bins lie in these distinct coarse bins, respectively, so the fine indices satisfy

$$\kappa_{\mathcal{G}'}(z_{\pi_i}) < \kappa_{\mathcal{G}'}(z_{\pi_{i+1}}). \tag{46}$$

This holds for all $i = 1, \ldots, L-1$, establishing $z_{1:L} \in \mathcal{E}_{\pi,\mathcal{G}'}$.

Next, we show $\mathcal{E}_{\pi,\mathcal{G}'} \subseteq \mathcal{E}_{\pi}$. Take any $z_{1:L} \in \mathcal{E}_{\pi,\mathcal{G}'}$ and write $z_{\pi_i} \in B'_{u_i}$. The strict index ordering $u_i < u_{i+1}$ and the definition $B'_u = (b'_{u-1}, b'_u]$ give

$$z_{\pi_i} \le b'_{u_i} \le b'_{u_{i+1}-1} < z_{\pi_{i+1}}. \tag{47}$$

The final inequality is strict because the higher bin excludes its left endpoint. Thus $z_{\pi_i} < z_{\pi_{i+1}}$ for every adjacent pair, implying $z_{1:L} \in \mathcal{E}_{\pi}$.

Both event inclusions now follow. Monotonicity of a probability measure under set inclusion yields

$$\mathbb{P}(\mathcal{E}_{\pi,\mathcal{G}}) \le \mathbb{P}(\mathcal{E}_{\pi,\mathcal{G}'}) \le \mathbb{P}(\mathcal{E}_{\pi}), \tag{48}$$

Substituting the notation for these three probabilities gives equation (10). □

B.4 Continuous probability error

Proof of theorem 3.4. Let $\mathcal{E}_{\pi,h,R}$ denote the discrete strict-path event with coverage radius R and maximum interior bin width h. The second inclusion in theorem 3.3 gives

$$\mathcal{E}_{\pi,h,R} \subseteq \mathcal{E}_{\pi}, \tag{49}$$

and hence

$$P_{\pi,h,R} \le P^{\star}_{\pi}, \qquad P^{\star}_{\pi} - P_{\pi,h,R} \ge 0. \tag{50}$$

We bound the omitted event $\mathcal{E}_{\pi} \setminus \mathcal{E}_{\pi,h,R}$. First, define the tail event in which at least one score lies outside the finite range:

$$\mathcal{T}_R = \bigcup_{i=1}^{L} \{|Z_{\pi_i}| > R\}. \tag{51}$$

The union bound gives

$$\mathbb{P}(\mathcal{T}_R) \le \sum_{i=1}^{L} \mathbb{P}(|Z_{\pi_i}| > R) = \sum_{i=1}^{L} r_{\pi_i}(R). \tag{52}$$

Now consider samples in $\mathcal{E}_{\pi} \cap \mathcal{T}_R^c$ that are still excluded by the discrete path event. All scores lie in $[-R, R]$ and satisfy

$$Z_{\pi_1} < Z_{\pi_2} < \cdots < Z_{\pi_L}. \tag{53}$$

Since the scores increase along the real line, their bin indices are nondecreasing. If all adjacent indices increased strictly, the sample would belong to $\mathcal{E}_{\pi,h,R}$. An omitted sample must therefore have some $i \in \{1, \ldots, L-1\}$ for which Z_{π_i} and $Z_{\pi_{i+1}}$ share a finite bin. Its width is at most h, implying

$$0 < Z_{\pi_{i+1}} - Z_{\pi_i} \le h. \tag{54}$$

Define the adjacent collision event

$$\mathcal{D}_i = \{0 < Z_{\pi_{i+1}} - Z_{\pi_i} \le h\}. \tag{55}$$

The preceding argument yields the inclusion

$$\mathcal{E}_\pi \setminus \mathcal{E}_{\pi,h,R} \subseteq \mathcal{T}_R \cup \bigcup_{i=1}^{L-1} \mathcal{D}_i. \tag{56}$$

To bound each $\mathcal{D}_i$, let $X = Z_{\pi_i}$ and $Y = Z_{\pi_{i+1}}$. Conditional independence gives joint density $f_{\pi_i}(x) f_{\pi_{i+1}}(y)$, so

$$\mathbb{P}(\mathcal{D}_i) = \int_{-\infty}^{+\infty} f_{\pi_i}(x) \int_x^{x+h} f_{\pi_{i+1}}(y)\,\mathrm{d}y\,\mathrm{d}x \tag{57}$$

$$\leq \int_{-\infty}^{+\infty} f_{\pi_i}(x) \int_x^{x+h} M_{\pi_{i+1}}\,\mathrm{d}y\,\mathrm{d}x \tag{58}$$

$$= hM_{\pi_{i+1}} \int_{-\infty}^{+\infty} f_{\pi_i}(x)\,\mathrm{d}x \tag{59}$$

$$= hM_{\pi_{i+1}}. \tag{60}$$

Integrating over the same region with respect to x first gives

$$\mathbb{P}(\mathcal{D}_i) = \int_{-\infty}^{+\infty} f_{\pi_{i+1}}(y) \int_{y-h}^{y} f_{\pi_i}(x)\,\mathrm{d}x\,\mathrm{d}y \tag{61}$$

$$\leq hM_{\pi_i}. \tag{62}$$

Both bounds hold, hence

$$\mathbb{P}(\mathcal{D}_i) \leq h \min\{M_{\pi_i}, M_{\pi_{i+1}}\}. \tag{63}$$

Finally, applying the union bound to equation (56) and substituting equations (52) and (63) yields

$$P^\star_\pi - P_{\pi,h,R} = \mathbb{P}(\mathcal{E}_\pi \setminus \mathcal{E}_{\pi,h,R}) \tag{64}$$

$$\leq \sum_{i=1}^{L} r_{\pi_i}(R) + h \sum_{i=1}^{L-1} \min\{M_{\pi_i}, M_{\pi_{i+1}}\}. \tag{65}$$

Together with equation (50), this proves equation (11).

For each fixed item, $\{|Z_j| > R\}$ decreases to the empty set as $R \to \infty$. Continuity from above of the probability measure gives $r_j(R) \to 0$. Since the density bounds M_j are fixed and the list is finite, the second term tends to zero as $h \to 0$. The error bound therefore vanishes, implying $P_{\pi,h,R} \to P^\star_\pi$. For a nested refinement sequence, theorem 3.3 further ensures that these probabilities are nondecreasing. □

B.5 Convergence of the training objective

Corollary B.2 (Negative log-likelihood error)**.** *Let $\varepsilon_{\pi,h,R}$ be the right-hand side of equation* (11)*. If $P^\star_\pi > \varepsilon_{\pi,h,R}$, then*

$$0 \leq -\log P_{\pi,h,R} + \log P^\star_\pi \leq \frac{\varepsilon_{\pi,h,R}}{P^\star_\pi - \varepsilon_{\pi,h,R}}. \tag{66}$$

Proof of corollary B.2. By equation (11),

$$P^\star_\pi - \varepsilon_{\pi,h,R} \leq P_{\pi,h,R} \leq P^\star_\pi. \tag{67}$$

The assumption $P^\star_\pi > \varepsilon_{\pi,h,R}$ makes all three quantities positive. Since the logarithm is increasing,

$$0 \leq -\log P_{\pi,h,R} + \log P^\star_\pi \tag{68}$$

$$= \log \frac{P^\star_\pi}{P_{\pi,h,R}} \tag{69}$$

$$\leq \log \frac{P^\star_\pi}{P^\star_\pi - \varepsilon_{\pi,h,R}} \tag{70}$$

$$= -\log\left(1 - \frac{\varepsilon_{\pi,h,R}}{P^\star_\pi}\right). \tag{71}$$

Set $x = \varepsilon_{\pi,h,R}/P^\star_\pi \in [0,1)$ and use the integral representation

$$-\log(1-x) = \int_0^x \frac{1}{1-u}\,\mathrm{d}u \le \int_0^x \frac{1}{1-x}\,\mathrm{d}u = \frac{x}{1-x}, \tag{72}$$

Substituting the definition of x yields

$$-\log(1-x) \le \frac{\varepsilon_{\pi,h,R}}{P^\star_\pi - \varepsilon_{\pi,h,R}}. \tag{73}$$

Combining this with equation (71) proves equation (66). □

Proposition B.3 (Consistency of grid gradients)**.** *Let $\{\mathcal{G}_n\}_{n\ge1}$ be parameter-independent shared grids with finite coverage radii $R_n \to \infty$ and maximum interior bin widths $h_n \to 0$. Suppose the joint density*

$$g(z_{1:L};\theta) = \prod_{j=1}^{L} f_j(z_j;\theta) \tag{74}$$

is differentiable with respect to θ in a neighborhood of θ_0, and an integrable function $G : \mathbb{R}^L \to \mathbb{R}_{\ge0}$ satisfies

$$\|\nabla_\theta g(z_{1:L};\theta)\| \le G(z_{1:L}) \tag{75}$$

almost everywhere throughout that neighborhood. If $P_{\pi,n}$ denotes the discrete order probability on $\mathcal{G}_n$, then

$$\nabla_\theta P_{\pi,n}(\theta_0) \xrightarrow[n\to\infty]{} \nabla_\theta P^\star_\pi(\theta_0). \tag{76}$$

If $P^\star_\pi(\theta_0) > 0$, then

$$\nabla_\theta \log P_{\pi,n}(\theta_0) \xrightarrow[n\to\infty]{} \nabla_\theta \log P^\star_\pi(\theta_0), \tag{77}$$

The negative log-likelihood gradients consequently converge as well.

Proof of proposition B.3. Let $\mathcal{E}_{\pi,n}$ denote the discrete strict-path event on $\mathcal{G}_n$. For $L = 1$, the continuous event and every discrete event equal the real line and have probability 1, so the result is immediate. For $L \ge 2$, we first establish pointwise convergence of the discrete event indicators to the continuous event indicator.

If $z_{1:L} \notin \mathcal{E}_\pi$, the inclusion $\mathcal{E}_{\pi,n} \subseteq \mathcal{E}_\pi$ implies $z_{1:L} \notin \mathcal{E}_{\pi,n}$ for every n, giving

$$\mathbf{1}_{\mathcal{E}_{\pi,n}}(z_{1:L}) = 0 = \mathbf{1}_{\mathcal{E}_\pi}(z_{1:L}). \tag{78}$$

If $z_{1:L} \in \mathcal{E}_\pi$, its finitely many adjacent differences are positive. Define

$$\delta(z) = \min_{1\le i<L}(z_{\pi_{i+1}} - z_{\pi_i}) > 0, \qquad R(z) = \max_{1\le j\le L}|z_j| < \infty. \tag{79}$$

Because $h_n \to 0$ and $R_n \to \infty$, there exists n_0 such that $h_n < \delta(z)$ and $R_n > R(z)$ for all $n \ge n_0$. All scores then lie within the finite range, and adjacent scores differ by more than the maximum interior bin width, so they cannot share a bin. Their strict score ordering therefore induces a strict bin ordering, giving $z_{1:L} \in \mathcal{E}_{\pi,n}$. Thus

$$\mathbf{1}_{\mathcal{E}_{\pi,n}}(z_{1:L}) \longrightarrow \mathbf{1}_{\mathcal{E}_\pi}(z_{1:L}). \tag{80}$$

The grid boundaries are independent of θ. The assumptions permit differentiation under the integral, giving

$$\nabla_\theta P_{\pi,n}(\theta_0) = \int_{\mathbb{R}^L} \mathbf{1}_{\mathcal{E}_{\pi,n}}(z)\nabla_\theta g(z;\theta_0)\,\mathrm{d}z, \tag{81}$$

$$\nabla_\theta P^\star_\pi(\theta_0) = \int_{\mathbb{R}^L} \mathbf{1}_{\mathcal{E}_\pi}(z)\nabla_\theta g(z;\theta_0)\,\mathrm{d}z. \tag{82}$$

By equation (80), the first integrand converges pointwise to the second. Also,

$$\left\|\mathbf{1}_{\mathcal{E}_{\pi,n}}(z)\nabla_\theta g(z;\theta_0)\right\| \le G(z), \tag{83}$$

where G is integrable. Applying the dominated convergence theorem to each gradient component proves equation (76).

Similarly, $0 \le \mathbf{1}_{\mathcal{E}_{\pi,n}}(z) g(z;\theta_0) \le g(z;\theta_0)$, and the joint density $g(\cdot;\theta_0)$ is integrable. Another application of dominated convergence gives

$$P_{\pi,n}(\theta_0) \to P^\star_\pi(\theta_0). \tag{84}$$

If $P^\star_\pi(\theta_0) > 0$, then $P_{\pi,n}(\theta_0) > 0$ for sufficiently large n. Using

$$\nabla_\theta \log P = \frac{\nabla_\theta P}{P}, \tag{85}$$

and the convergence of the numerator and denominator proves equation (77). Multiplication by -1 gives convergence of the negative log-likelihood gradient. □

B.6 Derivation of posterior supervision

Proof of proposition 3.5. We first verify the concatenation of forward and suffix masses. For a fixed state (i,t), define the feasible suffixes beginning after this state by

$$\mathcal{S}_{i,t} = \begin{cases} \{(t_{i+1},\dots,t_L) \in [T]^{L-i} : t < t_{i+1} < \cdots < t_L\}, & i < L, \\ \{()\}, & i = L. \end{cases} \tag{86}$$

Here $()$ denotes the empty suffix, whose empty product is 1. We show that the suffix mass has the sum representation

$$B_i(t) = \sum_{s \in \mathcal{S}_{i,t}} \prod_{k=i+1}^{L} q_{\pi_k}(t_k). \tag{87}$$

For $i = L$, the right-hand side is the empty product of the sole empty suffix, equal to $1 = B_L(t)$. Assume the expression holds at layer $i+1$. Every nonempty suffix in $\mathcal{S}_{i,t}$ has a unique first state $v = t_{i+1} > t$; removing that state leaves a sequence in $\mathcal{S}_{i+1,v}$. Conversely, prepending any $v > t$ to a sequence in $\mathcal{S}_{i+1,v}$ produces a suffix in $\mathcal{S}_{i,t}$. Partitioning by this unique first state gives

$$\sum_{s \in \mathcal{S}_{i,t}} \prod_{k=i+1}^{L} q_{\pi_k}(t_k) = \sum_{v=t+1}^{T} q_{\pi_{i+1}}(v) \sum_{s' \in \mathcal{S}_{i+1,v}} \prod_{k=i+2}^{L} q_{\pi_k}(t_k) \tag{88}$$

$$= \sum_{v=t+1}^{T} q_{\pi_{i+1}}(v) B_{i+1}(v) \tag{89}$$

$$= B_i(t). \tag{90}$$

The second equality uses the backward induction hypothesis, and the third uses equation (14). Thus equation (87) holds for all i.

The complete paths through (i,t) form the set

$$\mathcal{M}^{\text{full}}_{i,t} = \{m = (t_1,\dots,t_L) \in \mathcal{M}_{L,T} : t_i = t\}. \tag{91}$$

Let $\mathcal{M}_{i,t}$ denote the feasible prefixes ending at (i,t) in equation (6), and let $w_{1:i}(m^-)$ denote their product of bin masses. Under the strict ordering constraint, each such path corresponds uniquely to a feasible prefix in $\mathcal{M}_{i,t}$ and a feasible suffix in $\mathcal{S}_{i,t}$. Conversely, concatenating any such pair gives a unique path in $\mathcal{M}^{\text{full}}_{i,t}$. The prefix mass $A_i(t)$ includes $q_{\pi_i}(t)$, while the product in equation (87) starts at $k = i+1$, so the current factor appears exactly once. This bijection and distributivity of finite sums imply

$$A_i(t)B_i(t) = \left(\sum_{m^- \in \mathcal{M}_{i,t}} w_{1:i}(m^-) \right) \left(\sum_{m^+ \in \mathcal{S}_{i,t}} \prod_{k=i+1}^{L} q_{\pi_k}(t_k) \right) \tag{92}$$

$$= \sum_{m \in \mathcal{M}^{\text{full}}_{i,t}} w_\pi(m). \tag{93}$$

This also covers $i = L$, when the suffix sum equals 1.

Dividing equation (93) by the total path mass yields

$$\frac{A_i(t)B_i(t)}{P_{\pi,T}} = \sum_{m \in \mathcal{M}_{i,t}^{\text{full}}} \rho_\pi(m) = \gamma_i(t). \tag{94}$$

For fixed i, the sets $\mathcal{M}_{i,1}^{\text{full}}, \ldots, \mathcal{M}_{i,T}^{\text{full}}$ partition $\mathcal{M}_{L,T}$ by the state in column i. Therefore

$$\sum_{t=1}^{T} \gamma_i(t) = \sum_{m \in \mathcal{M}_{L,T}} \rho_\pi(m) = 1. \tag{95}$$

We now derive the gradient. Finite softmax logits ensure $q_{\pi_i}(t) > 0$, so define the log mass

$$\ell_i(t) = \log q_{\pi_i}(t). \tag{96}$$

Using $w_\pi(m) = \exp(\sum_{k=1}^{L} \ell_k(t_k))$, write the path probability as

$$P_{\pi,T} = \sum_{m=(t_1,\ldots,t_L) \in \mathcal{M}_{L,T}} \exp\left(\sum_{k=1}^{L} \ell_k(t_k)\right). \tag{97}$$

When differentiating with respect to $\ell_i(t)$, paths that avoid (i,t) contribute zero. For each path through the state, the derivative of its exponential term equals the path mass itself. Hence

$$\frac{\partial P_{\pi,T}}{\partial \ell_i(t)} = \sum_{m \in \mathcal{M}_{i,t}^{\text{full}}} w_\pi(m) = A_i(t)B_i(t), \tag{98}$$

$$\frac{\partial \log P_{\pi,T}}{\partial \ell_i(t)} = \frac{1}{P_{\pi,T}} \frac{\partial P_{\pi,T}}{\partial \ell_i(t)} = \gamma_i(t). \tag{99}$$

Next, propagate the gradient to the logits. For any $s, t \in [T]$, the log-softmax probability is

$$\ell_i(s) = a_{\pi_i}(s) - \log \sum_{v=1}^{T} \exp a_{\pi_i}(v), \tag{100}$$

Its Jacobian entries are therefore

$$\frac{\partial \ell_i(s)}{\partial a_{\pi_i}(t)} = \mathbf{1}\{s = t\} - \frac{\exp a_{\pi_i}(t)}{\sum_{v=1}^{T} \exp a_{\pi_i}(v)} \tag{101}$$

$$= \mathbf{1}\{s = t\} - q_{\pi_i}(t). \tag{102}$$

Since $\mathcal{L}_{\text{MPL}} = -\log P_{\pi,T}$, the chain rule together with equations (95) and (99) gives

$$\frac{\partial \mathcal{L}_{\text{MPL}}}{\partial a_{\pi_i}(t)} = -\sum_{s=1}^{T} \frac{\partial \log P_{\pi,T}}{\partial \ell_i(s)} \frac{\partial \ell_i(s)}{\partial a_{\pi_i}(t)} \tag{103}$$

$$= -\sum_{s=1}^{T} \gamma_i(s)\big(\mathbf{1}\{s = t\} - q_{\pi_i}(t)\big) \tag{104}$$

$$= -\gamma_i(t) + q_{\pi_i}(t) \sum_{s=1}^{T} \gamma_i(s) \tag{105}$$

$$= q_{\pi_i}(t) - \gamma_i(t). \tag{106}$$

This proves equation (17).

□

Zero-sum property and gradient norm. Since q_{π_i} and γ_i are normalized probability vectors, their difference gives

$$\sum_{t=1}^{T} \frac{\partial \mathcal{L}_{\text{MPL}}(\pi)}{\partial a_{\pi_i}(t)} = 0, \qquad \left\|\nabla_{a_{\pi_i}} \mathcal{L}_{\text{MPL}}(\pi)\right\|_1 \leq 2. \tag{107}$$

The zero-sum identity follows by summing equation (17) over bins, giving $1 - 1 = 0$. The triangle inequality yields

$$\left\|\nabla_{a_{\pi_i}} \mathcal{L}_{\text{MPL}}\right\|_1 = \sum_{t=1}^{T} |q_{\pi_i}(t) - \gamma_i(t)| \tag{108}$$

$$\leq \sum_{t=1}^{T} q_{\pi_i}(t) + \sum_{t=1}^{T} \gamma_i(t) = 2. \tag{109}$$

These identities concern bin logits; continuous parametric models receive gradients through the dependence of their bin masses on distribution parameters.

C Model Implementation

The path recurrence requires only normalized masses for each item on shared ordered states.

C.1 Score distribution representations

We represent score uncertainty with nonparametric discrete distributions, logistic distributions, and Gaussian mixtures. A nonparametric discrete distribution learns masses directly on T ordered states. A logistic distribution is specified by a location and a positive scale, while a Gaussian mixture uses component weights, means, and positive scales. All three supply the masses $q_i(1{:}T)$ required by the recurrence. At test time, the mean of each score distribution determines the permutation.

Nonparametric discrete distributions. We learn probability masses directly on shared ordered support points $\xi_1 < \cdots < \xi_T$ to represent score distribution shapes flexibly. Let $a_i(t) \in \mathbb{R}$ be the unnormalized log mass output for item i at state t. The normalized mass and mean are

$$q_i(t) = \frac{\exp a_i(t)}{\sum_{u=1}^{T} \exp a_i(u)}, \tag{110}$$

$$\widehat{s}_i = \sum_{t=1}^{T} q_i(t)\xi_t, \tag{111}$$

Here ξ_t is the finite representative value of state t. MultiDigit MNIST uses $T = 256$ equally spaced representative values across $[-4, 4]$. Each support point carries a learnable mass. The path loss uses only state indices and probabilities, while test-time rankings use the weighted mean in equation (111).

Logistic distributions. Let item i have logistic location $\mu_i \in \mathbb{R}$ and scale $s_i > 0$. Its CDF, bin masses, and mean are

$$F_i(z) = \frac{1}{1 + \exp(-(z - \mu_i)/s_i)}, \tag{112}$$

$$q_i(t) = F_i(b_t) - F_i(b_{t-1}), \tag{113}$$

$$\mathbb{E}[Z_i] = \mu_i. \tag{114}$$

In the real-world LTR experiments, we fix every item's raw scale to $s_0 = 1$ and apply a shared affine normalization within each query. For L_q valid items with raw locations $\widetilde{\mu}_i$, define

$$\overline{\mu}_q = \frac{1}{L_q} \sum_{i=1}^{L_q} \widetilde{\mu}_i, \tag{115}$$

$$v_q = \frac{1}{L_q} \sum_{i=1}^{L_q} \left[(\widetilde{\mu}_i - \overline{\mu}_q)^2 + s_0^2 \right], \tag{116}$$

$$\mu_i = \frac{\widetilde{\mu}_i - \overline{\mu}_q}{\sqrt{v_q}}, \qquad s_i = \frac{s_0}{\sqrt{v_q}}. \tag{117}$$

This transformation gives all items within a query the same normalized scale. The normalized locations satisfy $L_q^{-1} \sum_i \mu_i = 0$, and the location–scale second moment satisfies $L_q^{-1} \sum_i (\mu_i^2 + s_i^2) = 1$. The quantity v_q is the shared location–scale normalization statistic used in our implementation. Since the actual logistic variance is $\pi^2 s_i^2/3$, v_q is not the exact variance of a logistic mixture. For MPL, we evaluate equation (114) using the effective scale $s_i^{\text{eff}} = \sqrt{s_i^2 + 0.03^2}$ specified in Appendix D.4, on a shared grid of $T = 128$ bins whose finite interior boundaries span $[-4, 4]$. The first and last bins absorb the tails $(-\infty, -4]$ and $(4, +\infty)$, respectively.

Gaussian mixture distributions. Suppose item i has C Gaussian components, with weight w_{ic}, mean μ_{ic}, and scale σ_{ic} for component c, satisfying

$$w_{ic} \geq 0, \qquad \sum_{c=1}^{C} w_{ic} = 1, \qquad \mu_{ic} \in \mathbb{R}, \qquad \sigma_{ic} > 0. \tag{118}$$

Let ϕ and Φ denote the standard normal density and CDF. The mixture density is

$$f_i(z) = \sum_{c=1}^{C} w_{ic} \frac{1}{\sigma_{ic}} \phi\left(\frac{z - \mu_{ic}}{\sigma_{ic}} \right). \tag{119}$$

Weighting each component's probability in $B_t = (b_{t-1}, b_t]$ gives the bin mass needed by the recurrence. The mixture mean serves as the test-time ranking score:

$$q_i(t) = \sum_{c=1}^{C} w_{ic} \left[\Phi\left(\frac{b_t - \mu_{ic}}{\sigma_{ic}} \right) - \Phi\left(\frac{b_{t-1} - \mu_{ic}}{\sigma_{ic}} \right) \right], \tag{120}$$

$$\widehat{s}_i = \mathbb{E}[Z_i] = \sum_{c=1}^{C} w_{ic} \mu_{ic}. \tag{121}$$

We use $C = 3$ in the multimodal recovery experiment. Softmax normalizes the mixture weights, and a positive parameterization ensures $\sigma_{ic} > 0$. The mixture supplies $q_i(t)$ for the recurrence and is also used to compute Wasserstein distances, pairwise probabilities, and density plots.

State gap. The bin count T controls score resolution and the dynamic programming state count. The interior coverage radius R determines the range of finite boundaries, and the state gap $d \in \mathbb{N}_0$ specifies the minimum index difference between adjacent ordered states. Strict total orders use $d = 1$, requiring $t_{i+1} \geq t_i + 1$. The tied-block experiments on real-world LTR use $d = 0$, allowing adjacent relevance blocks to share a boundary state.

C.2 Modeling tied labels

We use the ordered blocks $C_1, \ldots, C_G$, independent bin indices U_i, and valid-assignment set from equation (18). The recurrence below applies to any nonnegative integer gap d; the real-world LTR experiments use $d = 0$.

C.2.1 Within-block boundary masses

Partial-order constraints connect adjacent blocks only through their minimum and maximum states. We therefore first compute the probability that a block lies within specified boundaries. For item i, define

$$p_i[a,b] = \mathbb{P}(a \le U_i \le b) = \sum_{t=a}^{b} q_i(t), \qquad 1 \le a \le b \le T. \tag{122}$$

Set $p_i[a,b] = 0$ when $a > b$. For block C_g, define the event

$$\mathcal{H}_g(a,b) = \left\{ a \le U_i \le b \text{ for all } i \in C_g, \quad \max_{i \in C_g} U_i = b \right\}, \tag{123}$$

and write $H_g(a,b) = \mathbb{P}(\mathcal{H}_g(a,b))$. For $1 \le a \le b \le T$, the event that all members lie in $[a, b-1]$ is contained in the event that all lie in $[a,b]$. Their difference requires at least one member to reach b. Conditional independence gives

$$H_g(a,b) = \prod_{i \in C_g} p_i[a,b] - \prod_{i \in C_g} p_i[a,b-1]. \tag{124}$$

When $b = a$, $p_i[a, a-1] = 0$, so the second product vanishes. Set $H_g(a,b) = 0$ when $a > b$.

Direct subtraction can lose precision when the products are close. To obtain an equivalent non-negative sum, write $n_g = |C_g|$ and fix any ordering $i_1, \dots, i_{n_g}$ of the block's members. For $r = 1, \dots, n_g$, define

$$\mathcal{H}_{g,r}(a,b) = \left\{ \begin{array}{ll} a \le U_{i_s} \le b-1, & s < r, \\ U_{i_r} = b, & s = r, \\ a \le U_{i_s} \le b, & s > r. \end{array} \right\} \tag{125}$$

The event $\mathcal{H}_{g,r}(a,b)$ specifies that i_r is the first item in this fixed member order to reach state b. Every assignment in $\mathcal{H}_g(a,b)$ has exactly one such first item, so

$$\mathcal{H}_g(a,b) = \bigsqcup_{r=1}^{n_g} \mathcal{H}_{g,r}(a,b). \tag{126}$$

Using conditional independence and summing the disjoint events gives

$$H_g(a,b) = \sum_{r=1}^{n_g} q_{i_r}(b) \prod_{s<r} p_{i_s}[a,b-1] \prod_{s>r} p_{i_s}[a,b]. \tag{127}$$

Every term is nonnegative, and the expression computes the same probability as equation (124). Precomputing left and right products along the member order avoids traversing the full block again for each possible first item reaching b.

C.2.2 Block recurrence

Let $\mathcal{E}_{1:g}^{(d)}$ denote the event that the sampled bin indices satisfy the constraints in equation (18) among the first g blocks, and define the dynamic programming state

$$D_g(b) = \mathbb{P}\left(\mathcal{E}_{1:g}^{(d)} \cap \left\{ \max_{i \in C_g} U_i = b \right\} \right). \tag{128}$$

Thus $D_g(b)$ aggregates all assignments satisfying the partial-order constraints among the first g blocks with the current block's maximum state equal to b.

Proposition C.1 (Exact marginalization over tied blocks)**.** *Under conditional independence of item states, the following recurrence computes the partial-order probability in equation* (18) *exactly:*

$$D_1(b) = H_1(1,b), \qquad b \in [T], \tag{129}$$

$$D_g(b) = \sum_{p=1}^{T} D_{g-1}(p) H_g(p+d, b), \qquad g = 2, \dots, G, \quad b \in [T], \tag{130}$$

$$P_{\text{tie}}^{(d)} = \sum_{b=1}^{T} D_G(b). \tag{131}$$

Here $H_g(p+d, b) = 0$ whenever $p + d > b$.

Proof. The first block has no preceding block constraint, and all states are at least 1. The event that C_1 has maximum state b is therefore exactly $\mathcal{H}_1(1, b)$, giving

$$D_1(b) = \mathbb{P}(\mathcal{H}_1(1, b)) = H_1(1, b). \tag{132}$$

This establishes the base case.

Assume that $D_{g-1}(p)$ exactly aggregates the first $g-1$ blocks with the preceding block's maximum fixed at p. To satisfy the gap constraint between blocks $g-1$ and g, every member of the current block must have state at least $p+d$. With its maximum fixed at b, the current block must satisfy precisely $\mathcal{H}_g(p+d, b)$, whose probability is $H_g(p+d, b)$.

The items in the first $g-1$ blocks are disjoint from C_g. By conditional independence, the joint event for fixed p has probability

$$D_{g-1}(p)H_g(p+d, b). \tag{133}$$

Each assignment satisfying the first g block constraints with current maximum b has a unique preceding-block maximum p. These events are disjoint across p, so summing over $p = 1, \ldots, T$ yields

$$D_g(b) = \sum_{p=1}^{T} D_{g-1}(p)H_g(p+d, b). \tag{134}$$

This proves the induction step.

Finally, each assignment satisfying all G block constraints has a unique final-block maximum b. The terminal events partition the full event, giving

$$P_{\text{tie}}^{(d)} = \sum_{b=1}^{T} D_G(b). \tag{135}$$

The base case, transitions, and terminal sum all match the target event, establishing exactness. □

Constructing each item's prefix masses

$$Q_i(t) = \sum_{u=1}^{t} q_i(u), \qquad Q_i(0) = 0, \tag{136}$$

allows constant-time interval-mass queries through $p_i[a, b] = Q_i(b) - Q_i(a-1)$. An explicit interval table for all $1 \le a \le b \le T$ contains $\mathcal{O}(T^2)$ entries per item and takes $\mathcal{O}(LT^2)$ time to build for all items.

For a fixed block C_g and boundary pair (a, b), equation (127) processes its n_g members. Prefix and suffix products along the member dimension give $\mathcal{O}(n_g)$ cost. Summing over all boundary pairs and blocks gives

$$\sum_{g=1}^{G} \mathcal{O}(n_g T^2) = \mathcal{O}\left(T^2 \sum_{g=1}^{G} n_g\right) = \mathcal{O}(LT^2). \tag{137}$$

The block recurrence enumerates T preceding maxima p for each (g, b), requiring $\mathcal{O}(GT^2)$ time. Since $G \le L$, the total time is $\mathcal{O}(LT^2)$.

Forward evaluation retains only T states for each of the preceding and current blocks. If $H_g(a, b)$ is constructed one block at a time, auxiliary memory is $\mathcal{O}(n_{\max} T^2 + T)$, where $n_{\max} = \max_g n_g$. Storing all item interval tables at once requires $\mathcal{O}(LT^2)$ memory. When every block is a singleton and $d = 1$, the strict total-order recurrence in the main text exploits predecessor prefixes to reduce the cost to $\mathcal{O}(LT)$. The quadratic cost here applies to general tied blocks, where both minimum and maximum states must be tracked.

C.3 Variable-length lists

Write P_{order} for the order probability of a list computed by either the strict total-order recurrence or the tied partial-order recurrence. In batched training, a valid-item mask distinguishes real items from padding. For strict total orders, we select valid items and initialize and update the prefix recurrence

at the true list length. For tied partial orders, we first select valid items and then form $C_1, \ldots, C_G$ by relevance label. Padding therefore enters neither bin mass normalization nor path probability computation.

A zero-length batch element contains no ranking instance and contributes zero loss. A length-one list has no adjacent ordering constraint, so its sole item satisfies the empty constraint with probability one: $P_{\text{order}} = 1$ and the negative log-likelihood is zero. For $L > 1$, MPL uses $-\log P_{\text{order}}/(L-1)$, where L counts valid items. At a fixed length, this only rescales the loss. With mixed lengths, it adjusts the training weight of each list by $L - 1$.

D Experimental Details

We summarize the MPL configurations, define evaluation metrics and statistical conventions, and give an overview across settings. The remaining sections detail the protocols and supplementary results for MultiDigit MNIST, real-world LTR, and multimodal score distribution recovery.

Table 4 summarizes each task's encoder, score distribution, discrete state count, and path objective.

Table 4: MPL implementations by task. Network widths denote hidden-layer sizes. The MNIST encoder has two CNN layers with 16,32 channels followed by a 32-dimensional fully connected layer. T is the number of discrete states or score bins used by the recurrence.

Setting	MultiDigit MNIST	WEB30K / Istella	Multimodal recovery
Encoder	Two-layer CNN + FC	Three-layer MLP	Two-layer MLP
Hidden widths	$16, 32;\ 32$	256,128,64	96,96
Score distribution	Nonparametric discrete	Logistic	Gaussian mixture
T	256	128	128
Path objective	Strict total order	Tied-block partial order	Strict total order

Distribution representations and training protocols are detailed in appendices C.1 and D.3 to D.5. The MNIST encoder uses 3×3 convolutions, each followed by 2×2 max pooling. Every hidden layer in the two LTR tasks includes batch normalization.

D.1 Evaluation metrics

We define all metrics with their directions, units, and aggregation conventions. Ranking metrics assess the final predicted order, distribution recovery metrics assess item score distributions, and probability metrics assess pairwise events and complete permutations. The task-specific sections then describe protocols, evaluation configurations, and additional results.

D.1.1 Ranking performance

Suppose the test set contains N lists. List n has length L_n, target permutation $\pi^{(n)}$, and predicted permutation $\widehat{\pi}^{(n)}$. When all lists in a setting have the same length, denote it by L. For MNIST, these permutations sort integer labels and predicted scores in ascending order. For the multimodal ranking metrics, they sort true and predicted score means in descending order, as implemented in the evaluation. Sampled score permutations are used for the probability metrics in Appendix D.1.3.

Permutation accuracy. Exact match requires every position in the predicted permutation to match the target. Element accuracy averages the proportion of correct positions within each list and then across lists:

$$\text{Exact} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}\left\{\widehat{\pi}^{(n)} = \pi^{(n)}\right\}, \tag{138}$$

$$\text{Element} = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{L_n}\sum_{i=1}^{L_n} \mathbf{1}\left\{\widehat{\pi}_i^{(n)} = \pi_i^{(n)}\right\}. \tag{139}$$

The MNIST tables multiply both metrics by 100 to report percentages.

Rank correlation. Let C_n and D_n count concordant and discordant item pairs in list n. Without tied labels, Kendall's τ is

$$\text{Kendall} = \frac{1}{N}\sum_{n=1}^{N}\frac{C_n - D_n}{\binom{L_n}{2}}. \tag{140}$$

Values closer to 1 indicate greater agreement in pairwise order. MNIST tables report 100 times Kendall's τ.

Mean reciprocal rank. Let $\widehat{r}_n(j)$ denote the predicted rank of item j and $j_n^\star = \pi_1^{(n)}$ the target first item. Mean reciprocal rank is

$$\text{MRR} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{\widehat{r}_n(j_n^\star)}. \tag{141}$$

MRR approaches 1 as the target first item moves toward the top of the predicted list. MNIST tables also multiply MRR by 100.

Top-1 accuracy. Top-1 accuracy is the fraction of lists whose predicted first item matches the target first item:

$$\text{Top1} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{1}\left\{\widehat{\pi}_1^{(n)} = \pi_1^{(n)}\right\}. \tag{142}$$

Higher is better. The multimodal recovery experiment reports this metric as a proportion in $[0, 1]$.

Pairwise accuracy. In multimodal recovery, predicted score means determine a complete permutation, and pairwise accuracy is the fraction of item pairs whose order agrees with the target:

$$\text{PairAcc} = \frac{1}{N}\sum_{n=1}^{N}\frac{C_n}{\binom{L_n}{2}} = \frac{1 + \text{Kendall}}{2}. \tag{143}$$

Here the final permutation orders every item pair. We report pairwise accuracy as a proportion, with higher values preferred. Real-world LTR uses the following convention for ties in labels and predicted scores.

Pairwise evaluation with tied labels. For query n, define the comparable pairs with different labels by

$$\mathcal{C}_n = \{(i, j) : 1 \le i < j \le L_n,\ y_i^{(n)} \ne y_j^{(n)}\}. \tag{144}$$

Let c_n, d_n, and t_n count pairs in $\mathcal{C}_n$ whose predicted order agrees with the labels, disagrees with the labels, or has equal predicted scores, respectively. Thus $c_n + d_n + t_n = |\mathcal{C}_n|$. For $|\mathcal{C}_n| > 0$, define the query-level metrics

$$\text{StrictKendall}_n = \frac{c_n - d_n}{|\mathcal{C}_n|}, \tag{145}$$

$$\text{PairAcc}_n = \frac{c_n + \frac{1}{2}t_n}{|\mathcal{C}_n|}. \tag{146}$$

A predicted tie therefore receives half credit in pairwise accuracy and contributes zero to the numerator of Strict Kendall. If $|\mathcal{C}_n| = 0$, our implementation sets $\text{StrictKendall}_n = 0$ and $\text{PairAcc}_n = 1$. We average equally over all nonempty queries:

$$\text{StrictKendall} = \frac{1}{N}\sum_{n=1}^{N}\text{StrictKendall}_n, \qquad \text{PairAcc} = \frac{1}{N}\sum_{n=1}^{N}\text{PairAcc}_n\,. \tag{147}$$

The main real-world LTR table reports both metrics multiplied by 100.

Discounted cumulative gain. For real-world LTR, let $y_j^{(n)}$ be the graded relevance of item j, and let k be the cutoff. Write $\widehat{\pi}^{(n),\downarrow}$ for the permutation that sorts predicted scores in descending order. We compute discounted cumulative gain $\mathrm{DCG@}\,k^{(n)}$ from this retrieval order. Its maximum, obtained by sorting true relevance in descending order, is $\mathrm{IDCG@}\,k^{(n)}$. Dataset-level normalized discounted cumulative gain is

$$\mathrm{DCG@}\,k^{(n)} = \sum_{r=1}^{\min(k,L_n)} \frac{2^{y^{(n)}_{\widehat{\pi}^{(n),\downarrow}_r}} - 1}{\log_2(r+1)}, \tag{148}$$

$$\mathrm{NDCG@}\,k = \frac{1}{N}\sum_{n=1}^{N} \frac{\mathrm{DCG@}\,k^{(n)}}{\mathrm{IDCG@}\,k^{(n)}}. \tag{149}$$

If $\mathrm{IDCG@}\,k^{(n)} = 0$, our implementation sets that query's NDCG to 1: all gains are zero, so the predicted ranking is equivalent to any ideal ranking. The main text reports $100 \times \mathrm{NDCG@}\,k$.

D.1.2 Score distribution recovery

The following metrics assess overall distribution shape, mixture weights, and recovery for items with nearly equal expected scores in the multimodal task. Predicted distributions use the validation-fitted affine calibration in Appendix D.5.5.

We use $J = 512$ equal-width evaluation bins over $[45, 115]$. Their boundaries are $e_j = 45 + 70j/J$ for $j = 0, \ldots, J$, and their width is $\Delta = 70/J$. Let $\widehat{F}_i$ and $F_i^\star$ denote the predicted and ground-truth CDFs for item i. We compute and renormalize masses within the evaluation range:

$$\widehat{m}_i(j) = \frac{\widehat{F}_i(e_j) - \widehat{F}_i(e_{j-1})}{\widehat{F}_i(e_J) - \widehat{F}_i(e_0)}, \tag{150}$$

$$m_i^\star(j) = \frac{F_i^\star(e_j) - F_i^\star(e_{j-1})}{F_i^\star(e_J) - F_i^\star(e_0)}, \qquad j = 1, \ldots, J. \tag{151}$$

Define cumulative masses through bin j as $\widehat{M}_i(j) = \sum_{r=1}^{j} \widehat{m}_i(r)$ and $M_i^\star(j) = \sum_{r=1}^{j} m_i^\star(r)$. For M test items, Score W1 is the discrete first Wasserstein distance

$$\mathrm{ScoreW1} = \frac{\Delta}{M}\sum_{i=1}^{M}\sum_{j=1}^{J}\left|\widehat{M}_i(j) - M_i^\star(j)\right|. \tag{152}$$

Distribution diagnostics use the evaluation masses in equation (151). Let

$$u_i(j) = \frac{\widehat{m}_i(j) + m_i^\star(j)}{2} \tag{153}$$

be the average predicted and ground-truth mass for item i in bin j. Score Jensen–Shannon divergence (Score JS) and mixture-weight mean absolute error are

$$\mathrm{ScoreJS} = \frac{1}{2M}\sum_{i=1}^{M}\sum_{j=1}^{J}\left[\widehat{m}_i(j)\log\frac{\widehat{m}_i(j)}{u_i(j)} + m_i^\star(j)\log\frac{m_i^\star(j)}{u_i(j)}\right], \tag{154}$$

$$\mathrm{WeightMAE} = \frac{1}{3M}\sum_{i=1}^{M}\sum_{c=1}^{3}|\widehat{w}_{ic} - w_{ic}^\star|, \tag{155}$$

Zero-mass terms take their continuous-limit value of zero, and $\widehat{w}_{ic}$ and $w_{ic}^\star$ are the predicted and ground-truth weights of corresponding components. Let

$$\mathcal{B} = \{i : \mathbb{E}[Z_i^\star] \in [79, 81]\} \tag{156}$$

be the set of items whose true means fall in the nearly equal-score band. Then

$$\mathrm{EqualBandW1} = \frac{\Delta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\sum_{j=1}^{J}|\widehat{M}_i(j) - M_i^\star(j)|. \tag{157}$$

Lower values are better for all these recovery errors. Score W1 and Equal-band W1 use ground-truth score units; Score JS and Weight MAE are dimensionless.

D.1.3 Order-probability prediction

Pairwise probability error and permutation likelihood assess the probabilities assigned to ordering events, complementing evaluation of final rankings and marginal distributions.

Let $\mathcal{P}$ be the evaluated set of item pairs. Draw $\widehat{Z}_i, \widehat{Z}_j$ independently from the predicted score distributions of items i, j, and draw $Z_i^\star, Z_j^\star$ independently from their ground-truth distributions. The predicted and true pairwise order probabilities are

$$\widehat{p}_{ij} = \Pr(\widehat{Z}_i < \widehat{Z}_j), \qquad p_{ij}^\star = \Pr(Z_i^\star < Z_j^\star). \tag{158}$$

Their mean absolute error is

$$\mathrm{PairProbMAE} = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} |\widehat{p}_{ij} - p_{ij}^\star|. \tag{159}$$

Pair probabilities are evaluated analytically by summing the Gaussian comparison probabilities over all component pairs. For a test permutation $\pi^{(n)}$ sampled from the ground-truth model, let $\widehat{P}_J(\pi^{(n)})$ and $P_J^\star(\pi^{(n)})$ denote finite-grid approximations under the predicted and true distributions, respectively. For each model and query, we center all component means by the query mixture mean and divide means and scales by the query mixture standard deviation. We then apply the strict-order recurrence on $J = 512$ bins with finite interior boundaries spanning $[-6, 6]$ and unbounded tail bins. Permutation-NLL regret is

$$\mathrm{PermRegret} = \frac{1}{N} \sum_{n=1}^{N} \left[-\log \widehat{P}_J(\pi^{(n)}) + \log P_J^\star(\pi^{(n)}) \right]. \tag{160}$$

Pair-probability MAE is dimensionless, and permutation-NLL regret is measured in nats per permutation. Lower values are better for both. With exact continuous probabilities, the population excess log loss equals a KL divergence. The reported statistic uses sampled permutations and finite-grid approximations, so it need not be nonnegative.

D.1.4 Statistical conventions

Across experiments, quality metrics and independent training-step timings are summarized as means across seeds with two-sided 95% Student-t confidence intervals. For n independent seeds with values $v_1, \ldots, v_n$, we report

$$\overline{v} \pm t_{0.975,n-1} \frac{s_v}{\sqrt{n}}, \qquad s_v^2 = \frac{1}{n-1} \sum_{r=1}^{n} (v_r - \overline{v})^2, \tag{161}$$

Here $\overline{v}$ is the seed mean, s_v the sample standard deviation, and $t_{0.975,n-1}$ the Student-t quantile with $n-1$ degrees of freedom. MNIST, multimodal recovery, and independent timing use $n = 5, 4, 5$, respectively; Istella uses $n = 10$. Each timing seed first averages its measured complete optimization steps before aggregation across seeds.

For equal fold weighting, we first average the five test-fold metrics within each seed, then compute the overall mean and two-sided 95% Student-t interval from these ten seed means. This interval measures seed variability after averaging over folds. We separately describe fold variability by the standard deviation of the ten-seed mean for each fold. For example, the across-fold NDCG@10 standard deviations are 0.125 percentage points for MPL and 0.139 for FOB.

D.1.5 Runtime

Training-step time, measured in milliseconds, is the time required for one complete model optimization update; lower is better. Section D.4 gives the measurement procedure, and Section D.1.4 specifies aggregation across seeds.

D.1.6 Ranks across settings

The main tables retain each task's original metrics, units, and confidence intervals. Only figure 5 converts results with different units into within-setting ranks to show relative performance across

lengths and tasks. The best point estimate receives rank 1, and equal point estimates share a rank. A method's mean rank is the arithmetic average over its evaluated settings; unevaluated settings are excluded from the denominator. This descriptive overview discards the magnitude of performance differences and does not replace the original tables or significance analysis.

D.2 Comparison across tasks

Figure 5 summarizes the original benchmark methods (the additional FOB (nonparametric) control is reported in the MNIST tables) using the within-setting ranks defined in appendix D.1.6, providing an overview of the experiments below.

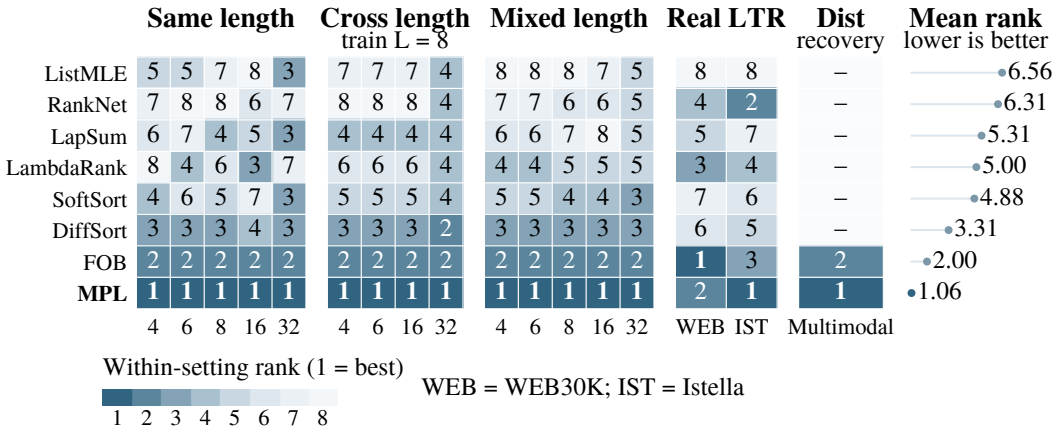

Figure 5: Performance ranks of the original benchmark methods, excluding the additional FOB (nonparametric) control, within experimental settings and mean ranks. Columns cover matched-length, fixed-$L_{\text{train}} = 8$ cross-length, and mixed-length MultiDigit MNIST tests, the two real-world LTR datasets, multimodal score distribution recovery (Multimodal), and a mean-rank dot plot. Ranking uses exact match for MNIST, five-fold aggregate NDCG@10 for WEB30K, NDCG@10 for Istella, and Score W1 for multimodal recovery. Ranks are computed independently from frozen-model test point estimates in each setting, with 1 best and darker colors indicating better ranks. Equal point estimates share a rank; dashes denote unevaluated settings. The right panel shows each method's arithmetic mean rank over evaluated settings, excluding missing entries. Methods are ordered by decreasing mean rank, placing the best overall method at the bottom.

D.3 MultiDigit MNIST

The first 55000 images of the official MNIST training split supply training digits, and the remaining 5000 supply validation digits; the official 10000 test images supply test digits. Within each split, digit images are sampled to form four-digit numbers, and repeated number labels within a list are resampled to ensure strict orders. We average-pool the original 28×28 digit images with kernel size and stride 2 to increase task difficulty, then concatenate four 14×14 digits horizontally into a 14×56 grayscale image per item. Convolutional neural networks (CNNs) with the same architecture independently produce raw scores or distribution parameters for each item. Matched-length models train for 30000 steps at $L \in \{4, 6, 8\}$ and 60000 steps at $L \in \{16, 32\}$, with batch size 500 and learning rate 10^{-3}. FOB follows its original configuration (Tang et al., 2026). MPL uses a nonparametric discrete score distribution on $T = 256$ equally spaced support points:

$$\xi_t = -4 + \left(t - \frac{1}{2}\right)\frac{8}{256}, \qquad t = 1, \ldots, 256, \tag{162}$$

The strict state gap is $d = 1$. Initialization uses scale 0.5 to form a smooth Gaussian-shaped logit profile over these representative values, concentrating early mass near the initial location predicted by the backbone. Gradient norms are clipped at 1 to limit the total gradient magnitude in each update.

FOB **(nonparametric).** To assess the role of the score representation, FOB (nonparametric) uses the same CNN and 256-mass output head as MPL, with the same initialization scheme and distribution-mean prediction. With bin edges $b_t = -4 + 8t/T$, it assigns density $q_i(t)/(b_t - b_{t-1})$ within bin t. Consequently, its continuous score mean equals the discrete grid mean. Given ordered cutpoints $-4 = c_0 < c_1 < \cdots < c_L = 4$, the training bound is $\prod_i [F_{\pi_i}(c_i) - F_{\pi_i}(c_{i-1})]$. Interval probabilities are evaluated from bin overlaps in log space. This retains the FOB rectangle-bound objective and adapts SRGA to a piecewise-uniform density; it does not use discrete boundary search.

Each model update uses 40 simultaneous SRGA steps with constant step size 0.01. Cutpoint gradients are clipped to $[-1, 1]$, and each displacement is restricted to 0.45 times the neighboring interval width. Initial cutpoints are midpoints of sorted predicted means, blended with 0.001 weight on equally spaced cutpoints to ensure positive interval widths. The right-hand density is used at internal bin boundaries. The final cutpoints are detached before differentiating the negative log bound, normalized by $L - 1$. The loss is not clipped, and no additional Gaussian score normalization is applied. The comparison thus controls the network and mass parameterization, while also changing the probability calculation and solver: continuous histograms permit strict order within one bin, whereas the discrete MPL event excludes shared-bin assignments. It is not a comparison of two exact solvers for an identical probability event.

The adaptation uses seeds 0–4, Adam with learning rate 10^{-3}, batch size 500, and the matched-length budgets above. Mixed training runs for 60000 total steps, choosing one of the five lengths uniformly per step. Every 1000 steps, validation uses up to 50 batches per training length; final evaluation uses up to 100 test batches per test length. The model trained at $L = 8$ is reused for unseen-length tests. Confidence intervals summarize the five independent seeds.

Table 5 adds confidence intervals, Kendall's τ, and MRR for matched-length experiments. Table 6 gives unseen-length results after training at $L_{\text{train}} = 8$, and table 7 gives results at each length after joint training on all five lengths.

Table 5: Full metrics for matched-length MultiDigit MNIST training and testing (mean ± 95% CI, percentages). The compact main-text table 1 arranges methods by rows and settings by columns; this table adds confidence intervals, Kendall's τ, and MRR. Within each L, methods are ordered by increasing exact match, then element accuracy. Bold denotes the best point estimate.

L	Method	Exact match (%, ↑)	Element acc. (%, ↑)	Kendall τ (%, ↑)	MRR (%, ↑)
4	LambdaRank	74.31 ± 5.97	86.02 ± 3.37	89.62 ± 2.76	95.66 ± 0.96
	RankNet	76.96 ± 3.59	87.51 ± 2.03	90.84 ± 1.56	96.25 ± 0.73
	LapSum	77.83 ± 5.03	87.97 ± 2.78	91.13 ± 2.17	96.45 ± 0.79
	ListMLE	79.24 ± 3.33	88.79 ± 1.84	91.79 ± 1.41	96.61 ± 0.68
	SoftSort	79.27 ± 3.67	88.84 ± 2.01	91.88 ± 1.51	96.56 ± 0.79
	DiffSort	81.83 ± 2.49	90.15 ± 1.43	92.80 ± 1.09	97.19 ± 0.32
	FOB	81.91 ± 0.74	90.16 ± 0.46	92.75 ± 0.34	96.87 ± 0.30
	FOB (nonparametric)	91.70 ± 0.91	95.26 ± 0.55	96.41 ± 0.43	98.46 ± 0.12
	MPL	$\mathbf{92.91 \pm 0.33}$	$\mathbf{95.83 \pm 0.17}$	$\mathbf{96.76 \pm 0.12}$	$\mathbf{98.56 \pm 0.10}$
6	RankNet	55.61 ± 9.97	81.66 ± 4.83	91.25 ± 2.54	94.92 ± 2.12
	LapSum	56.33 ± 6.09	82.00 ± 2.95	91.37 ± 1.59	96.08 ± 0.72
	SoftSort	56.92 ± 4.81	82.27 ± 2.28	91.56 ± 1.24	95.82 ± 0.50
	ListMLE	57.43 ± 3.88	82.57 ± 1.79	91.74 ± 0.91	95.55 ± 0.60
	LambdaRank	57.56 ± 6.85	82.60 ± 3.07	91.75 ± 1.54	95.54 ± 1.54
	DiffSort	62.55 ± 6.45	84.65 ± 2.86	92.68 ± 1.50	96.09 ± 0.53
	FOB	63.00 ± 1.01	84.76 ± 0.46	92.70 ± 0.22	95.99 ± 0.90
	FOB (nonparametric)	84.59 ± 0.44	93.32 ± 0.16	96.65 ± 0.07	98.16 ± 0.10
	MPL	$\mathbf{87.22 \pm 1.56}$	$\mathbf{94.23 \pm 0.78}$	$\mathbf{97.03 \pm 0.43}$	$\mathbf{98.52 \pm 0.19}$
8	RankNet	34.88 ± 7.73	75.85 ± 4.22	91.17 ± 1.85	93.92 ± 0.88
	ListMLE	37.24 ± 3.21	77.05 ± 1.50	91.62 ± 0.60	94.88 ± 0.67
	LambdaRank	38.49 ± 10.58	77.44 ± 5.69	91.69 ± 2.53	95.13 ± 0.89
	SoftSort	43.29 ± 4.09	80.00 ± 1.52	92.84 ± 0.49	95.38 ± 0.63
	LapSum	43.44 ± 4.90	80.02 ± 2.15	92.83 ± 0.82	95.57 ± 0.66
	DiffSort	46.13 ± 4.23	80.93 ± 1.87	93.08 ± 0.84	95.77 ± 0.62
	FOB	46.96 ± 4.56	81.01 ± 2.10	93.03 ± 0.91	96.16 ± 0.60
	FOB (nonparametric)	75.34 ± 0.73	91.04 ± 0.40	96.58 ± 0.18	97.96 ± 0.34
	MPL	$\mathbf{79.68 \pm 2.48}$	$\mathbf{92.14 \pm 1.21}$	$\mathbf{96.90 \pm 0.53}$	$\mathbf{98.27 \pm 0.55}$
16	ListMLE	3.49 ± 3.31	61.86 ± 6.51	91.78 ± 1.88	91.55 ± 3.51
	SoftSort	4.15 ± 2.11	63.83 ± 4.60	92.27 ± 1.38	93.11 ± 1.24
	RankNet	4.87 ± 3.28	64.58 ± 5.30	92.56 ± 1.42	92.73 ± 1.72
	LapSum	5.21 ± 3.50	65.05 ± 6.34	92.66 ± 1.75	93.17 ± 1.49
	DiffSort	5.31 ± 2.38	66.10 ± 3.29	92.93 ± 0.94	93.98 ± 1.01
	LambdaRank	5.73 ± 3.05	66.66 ± 3.63	93.11 ± 0.86	93.58 ± 0.88
	FOB	15.67 ± 2.31	74.42 ± 1.02	94.74 ± 0.26	95.44 ± 0.66
	FOB (nonparametric)	43.03 ± 3.50	84.68 ± 1.31	96.79 ± 0.33	97.50 ± 0.48
	MPL	$\mathbf{54.88 \pm 3.28}$	$\mathbf{86.61 \pm 1.14}$	$\mathbf{97.05 \pm 0.26}$	$\mathbf{97.87 \pm 0.34}$
32	RankNet	0.00 ± 0.01	46.58 ± 1.53	92.75 ± 0.26	88.08 ± 3.71
	LambdaRank	0.00 ± 0.01	48.13 ± 1.57	92.94 ± 0.47	90.25 ± 2.22
	ListMLE	0.01 ± 0.02	45.32 ± 7.88	92.08 ± 2.20	89.12 ± 2.29
	LapSum	0.01 ± 0.02	49.28 ± 4.36	93.15 ± 1.01	90.25 ± 1.42
	SoftSort	0.01 ± 0.01	50.04 ± 3.03	93.39 ± 0.68	90.02 ± 2.15
	DiffSort	0.01 ± 0.01	51.23 ± 1.22	93.50 ± 0.37	89.83 ± 4.11
	FOB	0.04 ± 0.05	50.08 ± 3.01	92.55 ± 0.91	88.88 ± 3.37
	FOB (nonparametric)	4.71 ± 1.57	72.63 ± 1.44	96.67 ± 0.21	$\mathbf{97.23 \pm 0.28}$
	MPL	$\mathbf{17.21 \pm 2.98}$	$\mathbf{76.62 \pm 1.61}$	$\mathbf{96.99 \pm 0.28}$	97.12 ± 0.37

Table 6: Generalization to unseen lengths after training at $L = 8$ (mean $\pm$ 95% CI, percentages). The matched-length $L = 8$ results appear in table 1; only lengths absent from training are shown here. Bold denotes the best point estimate in each column.

	Exact match (%, ↑)			
Method	$L = 4$	$L = 6$	$L = 16$	$L = 32$
RankNet	78.04 ± 4.37	55.22 ± 7.13	2.07 ± 1.49	0.00 ± 0.00
ListMLE	78.97 ± 1.92	57.39 ± 2.43	2.28 ± 0.71	0.00 ± 0.01
LambdaRank	79.15 ± 6.52	58.19 ± 9.64	3.18 ± 2.23	0.00 ± 0.01
SoftSort	81.86 ± 1.89	62.63 ± 2.60	4.28 ± 1.37	0.00 ± 0.01
LapSum	81.90 ± 2.84	62.95 ± 3.72	4.52 ± 2.18	0.00 ± 0.01
DiffSort	82.92 ± 2.44	64.60 ± 2.86	5.56 ± 2.07	0.01 ± 0.01
FOB	83.16 ± 2.26	65.38 ± 3.44	6.37 ± 2.27	0.01 ± 0.02
FOB (nonparametric)	92.34 ± 0.37	84.44 ± 0.18	39.05 ± 1.25	4.30 ± 0.57
MPL	$\mathbf{93.24 \pm 1.11}$	$\mathbf{87.05 \pm 1.20}$	$\mathbf{49.49 \pm 4.39}$	$\mathbf{10.73 \pm 2.98}$
	Element accuracy (%, ↑)			
Method	$L = 4$	$L = 6$	$L = 16$	$L = 32$
RankNet	88.12 ± 2.48	81.52 ± 3.46	59.02 ± 5.60	39.65 ± 5.25
ListMLE	88.64 ± 1.10	82.55 ± 1.17	60.58 ± 2.30	41.54 ± 2.36
LambdaRank	88.69 ± 3.68	82.82 ± 4.54	61.47 ± 7.65	42.79 ± 7.16
SoftSort	90.23 ± 1.01	84.97 ± 1.06	64.81 ± 2.15	46.15 ± 2.54
LapSum	90.21 ± 1.54	85.04 ± 1.60	65.21 ± 3.49	46.59 ± 3.84
DiffSort	90.70 ± 1.38	85.54 ± 1.31	66.34 ± 2.98	48.01 ± 3.11
FOB	90.76 ± 1.31	85.77 ± 1.63	66.88 ± 3.10	48.99 ± 3.78
FOB (nonparametric)	95.58 ± 0.24	93.18 ± 0.17	83.48 ± 0.77	71.78 ± 0.93
MPL	$\mathbf{95.98 \pm 0.70}$	$\mathbf{94.10 \pm 0.62}$	$\mathbf{85.78 \pm 1.84}$	$\mathbf{75.22 \pm 2.82}$

Table 7: Test performance at each length after joint mixed-length training (mean $\pm$ 95% CI, percentages). All evaluated methods and test lengths are shown. Bold denotes the best point estimate in each column.

	Exact match (%, ↑)				
Method	$L = 4$	$L = 6$	$L = 8$	$L = 16$	$L = 32$
ListMLE	79.05 ± 6.36	58.20 ± 10.60	38.71 ± 12.46	3.60 ± 3.64	0.00 ± 0.00
LapSum	80.46 ± 2.31	59.96 ± 4.30	40.38 ± 4.60	3.45 ± 1.77	0.00 ± 0.00
RankNet	80.35 ± 5.71	59.61 ± 10.83	40.43 ± 12.57	4.07 ± 4.20	0.00 ± 0.01
LambdaRank	81.88 ± 2.87	62.55 ± 5.06	43.88 ± 6.33	4.70 ± 2.50	0.00 ± 0.01
SoftSort	81.67 ± 3.80	62.48 ± 6.60	43.98 ± 7.77	5.00 ± 3.55	0.01 ± 0.02
DiffSort	82.71 ± 3.44	64.40 ± 5.56	46.54 ± 6.52	6.29 ± 3.06	0.01 ± 0.02
FOB	86.32 ± 0.79	71.49 ± 1.91	55.79 ± 1.98	12.12 ± 2.06	0.05 ± 0.02
FOB (nonparametric)	93.06 ± 0.61	85.99 ± 1.50	77.61 ± 2.21	43.83 ± 4.53	6.40 ± 2.38
MPL	$\mathbf{93.88 \pm 0.90}$	$\mathbf{88.12 \pm 1.45}$	$\mathbf{81.55 \pm 2.28}$	$\mathbf{54.39 \pm 4.09}$	$\mathbf{15.78 \pm 3.32}$
	Element accuracy (%, ↑)				
Method	$L = 4$	$L = 6$	$L = 8$	$L = 16$	$L = 32$
ListMLE	88.57 ± 3.47	82.61 ± 4.74	77.37 ± 5.79	61.45 ± 8.21	43.25 ± 8.45
LapSum	89.41 ± 1.24	83.58 ± 1.83	78.42 ± 2.15	63.12 ± 2.98	44.87 ± 2.86
RankNet	89.31 ± 3.13	83.26 ± 4.88	78.18 ± 5.99	62.54 ± 8.32	44.40 ± 8.87
LambdaRank	90.16 ± 1.55	84.76 ± 2.15	80.06 ± 2.77	65.18 ± 4.20	46.90 ± 4.72
SoftSort	90.06 ± 2.07	84.73 ± 2.90	79.98 ± 3.49	65.24 ± 5.38	47.05 ± 5.67
DiffSort	90.55 ± 2.03	85.28 ± 2.56	80.83 ± 3.11	66.65 ± 4.40	49.08 ± 5.09
FOB	92.45 ± 0.48	88.32 ± 0.90	84.61 ± 0.91	72.50 ± 1.61	55.80 ± 1.99
FOB (nonparametric)	95.97 ± 0.34	93.89 ± 0.71	91.85 ± 0.81	85.18 ± 1.19	74.24 ± 2.08
MPL	$\mathbf{96.30 \pm 0.54}$	$\mathbf{94.46 \pm 0.72}$	$\mathbf{92.65 \pm 1.08}$	$\mathbf{86.66 \pm 1.57}$	$\mathbf{76.83 \pm 2.57}$

D.3.1 Grid-resolution ablation

We train and evaluate MPL at $L = 8$ with $T \in \{32, 64, 128, 256, 512\}$ equally spaced states for a nonparametric discrete distribution on $[-4, 4]$. Each setting uses seeds 0–4, 30000 training steps, batch size 500, learning rate 10^{-3}, temperature 1, and state gap $d = 1$. The encoder and remaining training settings are held fixed. All entries, including $T = 256$, come from the ablation runs collected separately from the main comparison. Changing T also changes the output dimension of the nonparametric discrete distribution; this experiment measures the trained model's sensitivity to grid resolution. The continuous grid-refinement guarantees concern a fixed underlying distribution.

Table 8: Grid-resolution ablation on MultiDigit MNIST with training and test list length $L = 8$. Entries are percentages, reported as mean $\pm$ 95% Student-t confidence interval across five seeds. Higher is better for all metrics; bold denotes the best point estimate in each column.

T	Exact match	Element acc.	Kendall's τ	MRR	Pairwise acc.
32	68.05 ± 1.57	88.59 ± 0.59	95.73 ± 0.26	97.38 ± 0.20	97.87 ± 0.13
64	76.05 ± 1.97	91.18 ± 0.77	96.62 ± 0.32	98.04 ± 0.32	98.31 ± 0.16
128	78.59 ± 2.80	91.84 ± 1.49	96.78 ± 0.70	98.18 ± 0.31	98.39 ± 0.35
256	79.59 ± 1.80	92.03 ± 1.01	96.84 ± 0.48	98.29 ± 0.45	98.42 ± 0.24
512	$\mathbf{80.81 \pm 1.50}$	$\mathbf{92.43 \pm 0.63}$	$\mathbf{96.98 \pm 0.26}$	$\mathbf{98.30 \pm 0.39}$	$\mathbf{98.49 \pm 0.13}$

D.4 Real-world learning to rank

For both real-world LTR datasets, we standardize each feature within its query using finite values and replace missing entries with zero after normalization. Queries with more than 64 documents are subsampled without replacement using a query- and epoch-seeded generator; shorter queries are padded, with a valid-item mask excluding padding. Multilayer perceptrons (MLPs) with the same architecture process document features independently and output raw logistic locations. We fix the raw scale to $s_0 = 1$ and jointly normalize locations and scales using valid items within each query, as in equation (117). To prevent very small normalized scales from concentrating mass in a few bins, we use a numerical scale floor of 0.03 and define the effective scale for bin mass computation as

$$s_i^{\text{eff}} = \sqrt{s_i^2 + 0.03^2}. \tag{163}$$

The shared grid has $T = 128$ bins with finite interior boundaries spanning $[-4, 4]$. The partial-order recurrence uses $d = 0$ and all cross-label constraints in proposition C.1.

We evaluate all five WEB30K folds. Folds 1, 4, and 5 each contain 18919 training, 6306 validation, and 6306 test queries. Fold 2 contains 18918, 6306, and 6307, respectively, and Fold 3 contains 18918, 6307, and 6306. Istella full provides training and test sets with 23219 and 9799 queries. We construct a query-level validation set from the training set: NumPy's `default_rng` permutes query indices in file order using the run seed $(0, \ldots, 9)$, and the first $\text{round}(0.1 \times 23219) = 2322$ queries form the validation set. The remaining 20897 queries form the training set, and the official test set is unchanged. Methods with the same seed share the same partition; partitions vary across seeds, so the reported seed variability also reflects this split variation. Following the training setup of prior work, both tasks train for one full epoch with batch size 32, learning rate $10^{-3.5} \approx 3.1623 \times 10^{-4}$, zero weight decay, and gradient-norm clipping at 5. Each WEB30K method uses seeds $0, \ldots, 9$ on every fold, for $5 \times 8 \times 10 = 400$ runs. Istella uses ten seeds per method.

Training-step time is measured in a separate experiment. Each method uses five timing seeds, each measuring ten complete optimization steps with device synchronization at timing boundaries. We average step times within each seed, then compute the mean and confidence interval across the five seeds.

Five-fold aggregation and confidence intervals follow appendix D.1.4. Figure 6 compares the four primary metrics.

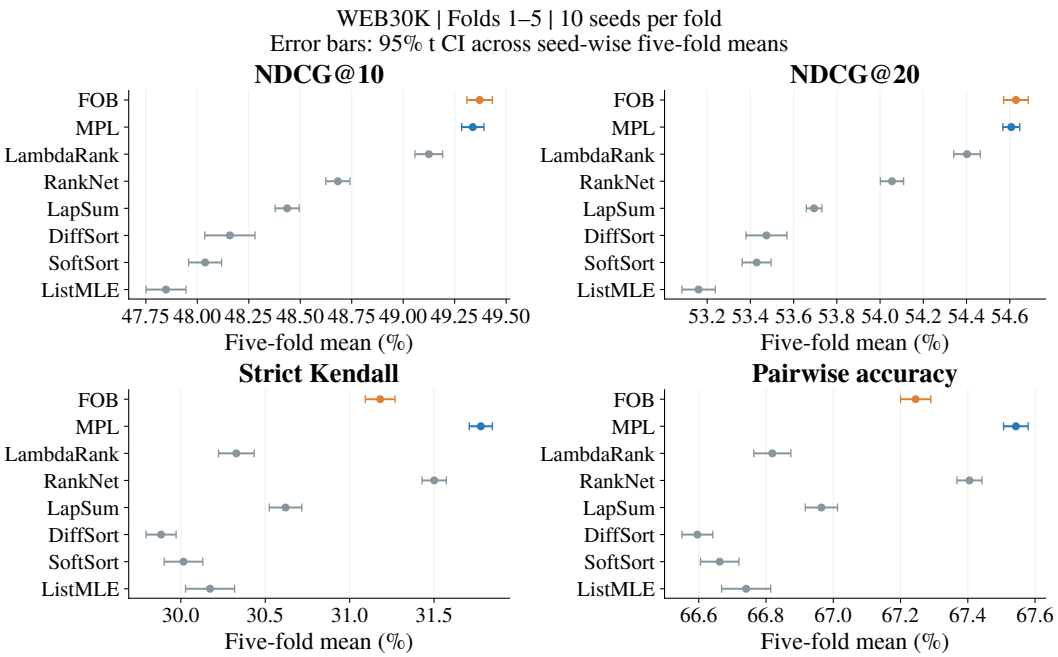

Figure 6: Aggregated five-fold WEB30K test quality with ten seeds per fold. Points are equally weighted five-fold means, and error bars are 95% confidence intervals computed from the ten per-seed five-fold means. Higher is better for all four metrics. Blue denotes MPL and orange denotes FOB.

D.5 Multimodal score distribution recovery

D.5.1 Supplementary results

Table 9: Supplementary ranking and distribution recovery metrics for multimodal score distribution recovery (mean $\pm$ 95% Student-t CI). The left table evaluates predicted orders, and the right evaluates distribution recovery. Arrows indicate preferred directions; bold denotes the best point estimate. Ranking accuracies are proportions, with Kendall's $\tau \in [-1, 1]$. Equal-band W1 uses score units; other distribution errors are dimensionless. Weight MAE measures mixture-weight error, and Equal-band W1 is Score W1 within the nearly equal-score band.

(a) Ranking performance

Metric	FOB	MPL
Pairwise acc. ↑	0.85 ± 0.10	$\mathbf{0.92 \pm 0.03}$
Kendall τ ↑	0.71 ± 0.20	$\mathbf{0.84 \pm 0.06}$
Top-1 acc. ↑	0.74 ± 0.20	$\mathbf{0.75 \pm 0.09}$

(b) Distribution recovery (interpretability)

Metric	FOB	MPL
Score JS ↓	0.14 ± 0.01	$\mathbf{0.10 \pm 0.01}$
Weight MAE ↓	0.14 ± 0.02	$\mathbf{0.11 \pm 0.01}$
Equal-band W1 ↓	2.72 ± 0.11	$\mathbf{1.67 \pm 0.16}$

Table 9 adds six metrics to those in table 3: pairwise accuracy, Kendall's τ, top-1 accuracy, Score JS, mixture-weight error, and Equal-band W1. MPL has better point estimates on all six.

D.5.2 Data generation

This task gives three machine types similar expected scores but distinct multimodal shapes, testing whether permutation supervision can learn distributional structure beyond a scalar mean. Each item's type $k \in \{1, 2, 3\}$ is sampled independently and uniformly, corresponding to high-speed, precision, and balanced machines. Its six features represent speed, precision, stability, energy efficiency, vibration control, and durability. The fixed scoring weights are

$$a = (0.25, 0.25, 0.20, 0.15, 0.10, 0.05)^\top. \tag{164}$$

We specify initial type centers, with row k of the following matrix equal to $\widetilde{c}_k^\top$:

$$\widetilde{C} = \begin{pmatrix} 1.50 & -0.30 & -0.35 & -0.25 & -0.70 & 0.00 \\ -0.45 & 1.35 & 0.90 & 0.05 & 0.65 & 0.30 \\ 0.35 & 0.35 & 0.45 & 0.75 & 0.35 & 0.75 \end{pmatrix}. \tag{165}$$

We then adjust each center along the scoring direction so that all three centers have the same linear score:

$$c_k = \widetilde{c}_k - \frac{a^\top \widetilde{c}_k - 0.35}{\|a\|_2^2} a, \qquad a^\top c_k = 0.35. \tag{166}$$

An item's features combine its type center, dimension-wise noise, and a shared perturbation along a fixed direction:

$$x = c_k + \varepsilon + uv, \tag{167}$$

$$\varepsilon \sim \mathcal{N}(0, 0.34^2 I_6), \qquad u \sim \mathcal{N}(0, 0.18^2), \tag{168}$$

$$v = (0.50, -0.35, 0.25, 0.15, -0.20, 0.30)^\top. \tag{169}$$

Here u is shared only across the six feature dimensions of one item. Types and noise terms are sampled independently, as are different items. The ground-truth expected score follows the same rule for every item: $m^\star(x) = 80 + 7.5 a^\top x$.

Ground-truth scores follow a three-component Gaussian mixture. Type determines baseline mixture weights and peak separations, while features further adjust the components. The true mixture weights are

$$w_c^\star(x, k) = \frac{\exp\big(A_{kc} + 0.35(x^\top B)_c\big)}{\sum_{j=1}^3 \exp\big(A_{kj} + 0.35(x^\top B)_j\big)}, \qquad c \in \{1, 2, 3\}, \tag{170}$$

where the type-specific base logits and feature modulation matrix are

$$A = \begin{pmatrix} 0.10 & 0.55 & 0.35 \\ -1.20 & 1.60 & -1.20 \\ 0.25 & 0.65 & 0.25 \end{pmatrix}, \qquad B = \begin{pmatrix} 0.30 & -0.20 & 0.12 \\ -0.10 & 0.22 & -0.08 \\ 0.08 & 0.14 & -0.10 \\ -0.12 & 0.18 & 0.06 \\ 0.18 & -0.12 & 0.14 \\ -0.05 & 0.10 & -0.04 \end{pmatrix}. \tag{171}$$

Let x_j denote component j of feature vector x. The baseline left and right peak gaps are $\ell = (8.5, 4.0, 10.5)$ and $r = (12.0, 4.0, 10.5)$, with small feature-dependent adjustments:

$$g_-^\star = \ell_k + 0.7 \tanh(x_5 - 0.4 x_3), \tag{172}$$

$$g_+^\star = r_k + 0.7 \tanh(x_1 - 0.3 x_4), \tag{173}$$

$$o = (-g_-^\star, 0, g_+^\star), \tag{174}$$

$$\mu_c^\star = m^\star(x) + o_c - \sum_{j=1}^3 w_j^\star o_j. \tag{175}$$

The last line centers the component offsets by their weights, preserving the mixture mean $m^\star(x)$. The true component scales are

$$\sigma_c^\star = S_{kc}\big[1 + 0.10 \tanh(x_4 - x_5)\big], \qquad S = \begin{pmatrix} 2.2 & 2.0 & 2.8 \\ 1.5 & 1.35 & 1.6 \\ 2.4 & 1.8 & 2.4 \end{pmatrix}. \tag{176}$$

For each item, we draw a component according to $w^\star$, then independently draw a score from the corresponding $\mathcal{N}(\mu_c^\star, (\sigma_c^\star)^2)$. Sorting the sampled scores within each list in ascending order produces the training permutation.

Each seed generates 10000 training lists, 800 validation lists, and 1600 test lists of length $L = 6$, containing 60000, 4800, and 9600 items, respectively. For run seed $s \in \{11, 22, 33, 44\}$, the splits use random seeds $10000 + s$, $20000 + s$, and $30000 + s$. They are generated independently under the same rule, and MPL and FOB share all data within each run seed.

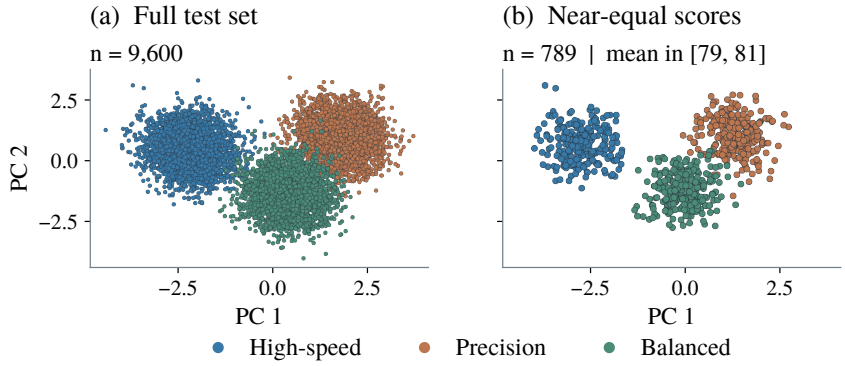

Figure 7: Type structure obscured by a scalar score. The left panel shows the first two principal components of all test instances; the right retains only instances with ground-truth expected scores in $[79, 81]$. Colors distinguish high-speed, precision, and balanced machines, with subset sizes below the panel titles. Even at similar one-dimensional expected scores, the six-dimensional features form three distinct clusters, revealing type-dependent distributional structure that scalar supervision does not explicitly represent.

D.5.3 Prediction model

A shared two-layer MLP with hidden width 96 and Gaussian error linear unit (GELU) activations processes each item's six-dimensional features. For item i, it outputs a raw location r_i^{base}, two raw gaps r_{ik}^{gap} for $k \in \{1, 2\}$, three raw scales r_{ic}^{scale} for $c \in \{1, 2, 3\}$, and three raw mixture weights η_{ic}. We normalize the weights as

$$w_{ic} = \frac{\exp(\eta_{ic})}{\sum_{\ell=1}^{3} \exp(\eta_{i\ell})}, \qquad c \in \{1, 2, 3\}, \tag{177}$$

and write $\text{sigmoid}(u) = 1/(1+\exp(-u))$. Component means and scales are then parameterized by

$$\beta_i = 80 + 18 \tanh(r_i^{\text{base}}), \tag{178}$$

$$g_{ik} = 1 + 14\,\text{sigmoid}(r_{ik}^{\text{gap}}), \qquad k \in \{1, 2\}, \tag{179}$$

$$\widetilde{\delta}_i = (-g_{i1}, 0, g_{i2}), \tag{180}$$

$$\delta_{ic} = \widetilde{\delta}_{ic} - \sum_{k=1}^{3} w_{ik} \widetilde{\delta}_{ik}, \tag{181}$$

$$\mu_{ic} = \beta_i + \delta_{ic}, \tag{182}$$

$$\sigma_{ic} = 0.75 + 3.75\,\text{sigmoid}(r_{ic}^{\text{scale}}) \tag{183}$$

The range $\beta_i \in (62, 98)$ covers the main ground-truth score locations. Gaps $g_{ik} \in (1, 15)$ for $k \in \{1, 2\}$ preserve component ordering while allowing different peak separations, and $\sigma_{ic} \in (0.75, 4.5)$ guarantees positive scales spanning the ground-truth component widths. Weighted centering ensures $\sum_c w_{ic}\delta_{ic} = 0$, so the mixture mean is exactly β_i, while the three components can still represent distinct peaks.

D.5.4 Training protocol

Within each optimization batch, MPL and FOB apply the same global affine normalization to their predicted mixtures. For N_b items in the batch, item i has mixture mean and variance

$$m_i = \sum_{c=1}^{3} w_{ic}\mu_{ic}, \tag{184}$$

$$v_i = \sum_{c=1}^{3} w_{ic}(\mu_{ic}^2 + \sigma_{ic}^2) - m_i^2. \tag{185}$$

We then define

$$\overline{m} = \frac{1}{N_b}\sum_{i=1}^{N_b} m_i, \tag{186}$$

$$s_b^2 = \frac{1}{N_b}\sum_{i=1}^{N_b}(m_i^2 + v_i) - \overline{m}^2, \tag{187}$$

$$\mu'_{ic} = \frac{\mu_{ic} - \overline{m}}{s_b}, \qquad \sigma'_{ic} = \frac{\sigma_{ic}}{s_b}. \tag{188}$$

MPL computes mixture CDF differences over $T = 128$ bins, with finite interior boundaries spanning $[-5, 5]$ and the first and last bins absorbing tail mass. Models train for 50 epochs with batch size 128, learning rate 10^{-3}, AdamW weight decay 10^{-4}, and gradient-norm clipping at 5. Runs use four seeds $\{11, 22, 33, 44\}$. FOB initializes cutpoints at midpoints of sorted predicted mixture means and performs at most 40 SRGA updates, stopping when the root-mean-square cutpoint gradient falls below 1. The step size is 0.01, gradients are clipped to $[-1, 1]$, and updates are restricted to 0.45 times the adjacent interval width. Final cutpoints are detached; interval log probabilities are clipped to $[-5, -0.01]$ before their sum is normalized by $L - 1$.

D.5.5 Evaluation and score calibration

Ordering supervision is invariant to a shared positive affine transformation of all scores. To evaluate distributions in ground-truth score units, we fit one positive affine map per model checkpoint using all validation items. Let $\widehat{m}_i$ and $m_i^\star$ be the raw predicted and true validation means, and let bars denote their averages. The calibration coefficients are

$$\alpha_{\mathrm{cal}} = \max\left\{10^{-6}, \frac{\sum_i(\widehat{m}_i - \overline{\widehat{m}})(m_i^\star - \overline{m}^\star)}{\max\{10^{-12}, \sum_i(\widehat{m}_i - \overline{\widehat{m}})^2\}}\right\}, \tag{189}$$

$$\beta_{\mathrm{cal}} = \overline{m}^\star - \alpha_{\mathrm{cal}}\overline{\widehat{m}}. \tag{190}$$

The two numerical floors enforce a positive slope and prevent division by zero. We transform component means and scales by $\mu_{ic}^{\mathrm{cal}} = \alpha_{\mathrm{cal}}\mu_{ic} + \beta_{\mathrm{cal}}$ and $\sigma_{ic}^{\mathrm{cal}} = \alpha_{\mathrm{cal}}\sigma_{ic}$, leaving mixture weights unchanged. The fitted map is held fixed for all test metrics and density plots. Training gradients use permutation labels; ground-truth validation distributions determine the calibration.

The ranking metrics compare predicted and true mean-score orders, with larger means ranked first. Pair-probability MAE compares the two models' analytic pair probabilities, and permutation-NLL regret uses the sampled test permutations and the finite-grid evaluation in Appendix D.1.3.