MixRoute: Rethinking Single-Distribution Training for Generalizable Neural Routing

Hang Yi[1], **Ziwei Huang**[1], **Yining Ma**[2]*, **and Zhiguang Cao**[1]
[1]Singapore Management University, [2]Massachusetts Institute of Technology
hang.yi.2024@phdcs.smu.edu.sg, ziweihuang@smu.edu.sg
yiningma@mit.edu, zgcao@smu.edu.sg

Abstract

Neural Combinatorial Optimization (NCO) is a promising paradigm for solving routing problems, yet its generalization across diverse data distributions remains a critical bottleneck. Existing methods typically tackle cross-distribution generalization through complex multi-distribution training or intricate meta-learning schemes, implicitly presuming that single-distribution training yields a weak zero-shot baseline. In this paper, we demonstrate that the potential of single-distribution training has been substantially underestimated and can be unlocked by a simple architectural inductive bias: Mix Normalization. To approach this, we first construct a comprehensive benchmark spanning three representative routing problems and 179 fine-grained datasets, capturing diverse shifts in node coordinates, customer demands, capacity and time windows. Building on this benchmark, we propose MixRoute, a neural framework that adaptively combines normalization statistics at different granularities to mitigate varying distribution shifts. Trained exclusively on uniform distributions, MixRoute generalizes effectively to a wide range of unseen distributions without any distribution-specific adaptation. Extensive experiments demonstrate that MixRoute consistently achieves state-of-the-art zero-shot generalization. Our work revisits normalization as a parsimonious alternative for zero-shot generalization against complex training-adaptation schemes. Code is available at https://github.com/yihang0410/MixRoute.

1 Introduction

The Vehicle Routing Problem (VRP) is a fundamental class of combinatorial optimization problems that seek cost-minimizing routes for serving geographically distributed customers under various operational constraints [38]. As a cornerstone of modern logistics, VRPs underpin a wide range of real-world applications, including transportation [20], supply chain management [6], and ride-sharing [31]. Owing to their NP-hard nature [24], obtaining exact solutions becomes computationally prohibitive as problem size grows. Classical heuristics, such as LKH3 [14] and HGS [41], can produce high-quality solutions but rely heavily on hand-crafted rules and domain expertise, making them difficult to adapt to new problem variants or operational constraints. To overcome these limitations, Neural Combinatorial Optimization (NCO) has emerged as a promising data-driven alternative [2, 3], leveraging deep neural networks to automatically learn solution policies from instance data. Among existing NCO paradigms, constructive methods [21, 22, 42, 47] which sequentially build a solution by appending one node at a time have become the dominant approach due to their ability to learn solution policies end-to-end without relying on hand-crafted local-search operators.

Despite their empirical success, most NCO methods are trained and predominantly evaluated on instances synthesized from a single canonical distribution, typically with node coordinates sampled

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

uniformly over the unit square [22, 21, 42, 2, 16]. Even works targeting size generalization have largely retained this uniform-distribution setup on both training and testing sides [27, 7, 10, 50, 37]. However, empirical evidence shows that once the node coordinates depart from this idealized regime, the performance of these solvers degrades sharply [4, 19, 12]. This indicates that what such models have learned is, to a large extent, statistical regularities specific to the training distribution rather than the problem-invariant combinatorial structure of routing itself. To address this cross-distribution generalization problem, a growing line of work has resorted to increasingly complex remedies, such as data augmentation across multiple distributions [18], meta-learning over diverse task families [51, 30, 34], multi-teacher distillation [4], and distribution-specific architectural components [46]. A common premise in these approaches is that single-distribution training is fundamentally insufficient.

In this work, we revisit this premise and ask a question that has received little attention: *what is the actual ceiling of single-distribution training?* Rather than further scaling up training data diversity, we look for inspiration outside of NCO. In computer vision, where cross-distribution generalization has been studied extensively, normalization layers have repeatedly emerged as a critical lever for explicitly controlling what statistics a model preserves or discards across distributions [28, 32, 33]. This motivates us to ask whether normalization could serve as an equally important entry point for cross-distribution generalization in NCO. Our preliminary experiments confirm that different normalization schemes indeed exhibit different behaviors across routing distributions, with no single scheme being uniformly optimal across the diverse distributions. Motivated by this observation, we propose **MixRoute**, a neural routing framework that adaptively harnesses the complementary strengths of multiple normalizations through a Mix Normalization (MixNorm) module and demonstrates that with the right normalization design, single-distribution training alone can already achieve strong zero-shot cross-distribution generalization. Our contributions are summarized as follows:

- We introduce MixNorm to NCO for the first time and identify it as a simple yet effective lever for cross-distribution generalization, showing that the ceiling of single-distribution training is substantially higher than previously assumed.
- We construct a comprehensive benchmark of 179 distribution shifts spanning diverse spatial and demand patterns, providing a unified testbed for systematically evaluating cross-distribution generalization in neural routing solvers.
- We extensively validate MixRoute for different routing problems, backbone architectures, and normalization choices, demonstrating its consistent effectiveness.

2 Related Work

Neural Solvers for VRPs. NCO leverages deep neural networks to learn solution policies directly from problem instances, and existing neural solvers for VRPs broadly fall into two paradigms: *construction* methods, which generate a complete solution from scratch, and *improvement* methods, which iteratively refine an initial solution through learned local-search operators [45, 29, 15]. Within the construction paradigm, autoregressive (AR) approaches have become the dominant choice: starting from Vinyals et al. [42] and Bello et al. [2], Kool et al. [21] introduced the Attention Model (AM), which was further advanced by POMO [22] and remains a canonical backbone for neural routing. Subsequent works have extended this line through heavier decoders [27], bisimulation-based reformulations [7], instance-conditioned adaptation [50], mixture-of-experts architectures [52], and vision-assisted multi-modal encoders [13]. Non-autoregressive (NAR) variants generate all edges in parallel via graph neural networks for faster inference [19], but typically trail AR methods in solution quality. Complementary to learned policies, recent work also employs large language models to automatically generate solvers for complex VRP variants [25, 49]. We adopt the AR construction paradigm as our backbone, following the dominant practice in recent neural routing literature.

Cross-Distribution Generalization in Neural VRP Solvers. A growing body of work has investigated cross-distribution generalization in neural VRP solvers. *Data-centric* approaches enrich the training distribution: Jiang et al. [18] formulate routing under a distributionally robust optimization objective, Geisler et al. [12] introduce adversarial perturbations to expose model vulnerabilities, and Wang et al. [43] adopt curriculum-style instance generation guided by policy-space response oracles. *Model-centric* approaches inject distributional priors into the solver itself through meta-learning over diverse task families [51, 30], multi-teacher knowledge distillation that consolidates expertise across distributions [4], or distribution-specific architectural components such as multiple decoders [46]. A

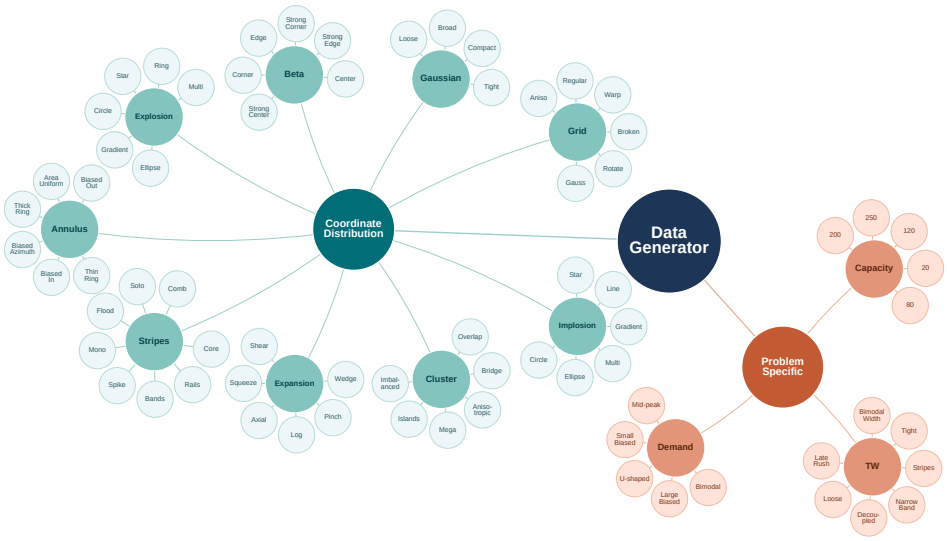

Figure 1: Hierarchical structure of our cross-distribution benchmark. Starting from a central data generator, the benchmark branches into two top-level axes: coordinate distributions and problem-specific shifts. Each family further expands into multiple parametric variants.

common thread across these methods is a shared assumption that single-distribution training is inherently insufficient for cross-distribution generalization, which has driven the field toward increasingly complex solutions. Our work challenges this assumption from a different angle: we show that a single uniform training distribution can already unlock strong cross-distribution generalization ability.

Mix Normalization. Different normalization schemes, including Batch Normalization (BN) [17], Layer Normalization (LN) [1], and Instance Normalization (IN) [40], differ in the granularity at which statistics are aggregated and exhibit complementary strengths across architectures and tasks. Switchable Normalization (SN) [28] introduces the idea of learnable normalizer mixtures, motivated by the empirical observation that no single normalizer is universally optimal across layers or tasks. Subsequent works extend this idea along several directions, including sparse selection among candidates [36], gated combinations restricted to specific normalizer pairs [32, 33], and domain-aware mixtures designed to mitigate distributional shift [35, 8]. However, these developments have remained almost exclusively within image-based settings. To the best of our knowledge, our work is the first to investigate mixture-based normalization in the context of NCO.

3 Benchmark Datasets across Diverse Distributions

To rigorously evaluate cross-distribution generalization, we construct a test suite that systematically varies the four structural attributes of a routing instance most likely to differ between training and deployment: node coordinates, customer demands, capacity and time window. All benchmark instances are used exclusively at test time; training is conducted on a single uniform distribution. Our full benchmark constitutes a structured taxonomy of distribution shifts, as visualized in Figure 1.

Coordinate distributions. Most prior neural VRP solvers train and evaluate exclusively on a single uniform distribution, while a handful of recent works extend evaluation to a small number of mutated patterns derived from Bossek et al. [5], commonly *Explosion*, *Implosion*, *Expansion*, and *Grid*, which serve as the de facto standard for cross-distribution evaluation [4, 51]. While these patterns introduce diversity, they remain a narrow slice of the broader instance space and are typically used in their canonical, fixed-parameter form only. We instead design nine structurally distinct coordinate families: five novel families, namely *Annulus*, *Beta*, *Gaussian*, *Stripe*, and *Cluster*, together with the four traditional ones mentioned above. To stress-test generalization, we further generate parametric variants within each family by varying its control parameters, where each parameter setting yields a substantially different spatial pattern rather than a mere re-sampling of the same distribution.

Demand distributions. Existing neural VRP solvers almost universally adopt a single uniform demand distribution [4, 51], overlooking the substantial demand variability observed in real-world deployments. We design five demand families, all constrained to share the same numerical range $[1, D_{\max}]$ to isolate the effect of distributional *shape* from that of *scale*.

Time window. Prior neural works almost exclusively evaluate on a single uniform time-window distribution identical to the training distribution, which fails to reflect the highly structured temporal patterns that arise in real-world delivery. Service windows in production logistics are rarely independent uniform draws: they are shaped by morning and evening rush hours, fixed departure slots in scheduled fleets, and customer business hours that cluster into a few standard bands. A model trained only on uniformly distributed windows therefore implicitly relies on two strong priors: that every time of day is equally likely, and that the temporal pattern decouples from the spatial layout. To address this, we build a time-window distribution shift benchmark consisting of 7 variants that systematically perturb three orthogonal axes of the temporal structure: the width and tightness of the windows themselves, the temporal location and concentration of when the bulk of service must occur, and the spatial-temporal coupling between customer position and window timing.

Capacity configurations. The first three dimensions vary the *distributional shape* of coordinates and demands; the fourth dimension instead perturbs the *numerical regime* itself. The effective load on each route is jointly determined by demand and vehicle capacity. Because demands enter the model as the normalized ratio d_i/Q, scaling the capacity Q is equivalent to rescaling all demands proportionally. Table 1 provides three illustrative configurations, where varying Q from 20 to 250 produces a more than tenfold change in the numerical range, despite identical demand sampling.

Table 1: Capacity regimes. Demands are sampled from $\{1, \dots, 9\}$ ($\bar{d} = 5$).

Regime	Q	$d_{\min}/Q$	$\bar{d}/Q$	$d_{\max}/Q$
Tight	20	0.050	0.250	0.450
Medium	50	0.020	0.100	0.180
Loose	250	0.004	0.020	0.036

Table 2 compares our benchmark with representative cross-distribution benchmarks for neural routing solvers. Existing benchmarks fall into two largely disconnected categories: AMDKD [4] and INViT [9] vary only node coordinates on TSP/CVRP, while Luo et al. [26] vary only constraint tightness on CVRP/CVRPTW. In contrast, our benchmark unifies both types of shift in a single, comprehensive testbed, spanning three problems (TSP, CVRP, CVRPTW), four shift dimensions (coordinates, demand, capacity, time window), and 179 fine-grained out-of-distribution test settings.

Table 2: Comparison of cross-distribution generalization benchmarks for neural routing solvers. # Train and # Test denote the numbers of training and out-of-distribution test distributions reported in the original works.

Benchmark	**# Distributions**		**Problem Coverage**			**TSP**	**CVRP-specific**		**CVRPTW-specific**			**Open-source**
	# Train	# Test	TSP	CVRP	CVRPTW	Coord.	Demand	Capacity	Width	Timing	Coupling	
AMDKD [4]	3	4	✓	✓	×	✓	×	×	×	×	×	✓
INViT [9]	1	4	✓	✓	×	✓	×	×	×	×	×	✓
Luo et al. [26]	multiple	7	×	✓	✓	×	×	✓	✓	×	×	✓
Ours	**1**	**179**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

4 Methodologies

Notation. Let $\mathcal{X}$ denote the instance space of a routing problem and $\mathcal{D}$ denote a probability distribution over $\mathcal{X}$. A neural solver $\pi_\theta : \mathcal{X} \to \mathcal{Y}$ parameterized by θ maps an instance $x \in \mathcal{X}$ to a feasible solution. Training amounts to minimizing the expected cost $\mathcal{L}(\theta; \mathcal{D}) = \mathbb{E}_{x \sim \mathcal{D}}[\, c(\pi_\theta(x))\,]$, where $c(\cdot)$ denotes the problem-specific cost (e.g., tour length).

Single-distribution training. Following the convention in early NCO works [21, 22], the model is trained on instances drawn from a single source distribution $\mathcal{D}_{\text{train}}$:

$$\theta^* = \arg\min_\theta \; \mathbb{E}_{x \sim \mathcal{D}_{\text{train}}}\big[\, c(\pi_\theta(x))\,\big], \tag{1}$$

where $\mathcal{D}_{\text{train}}$ is typically the Uniform distribution over the unit square $[0, 1]^2$. The trained model is expected to generalize to a target distribution $\mathcal{D}_{\text{test}} \neq \mathcal{D}_{\text{train}}$ at deployment, without further adaptation.

Multi-distribution training. To address the cross-distribution generalization issue, a growing line of work [18, 4, 51] expands the training data to span $K \geq 2$ distinct source distributions $\{\mathcal{D}_k\}_{k=1}^K$.

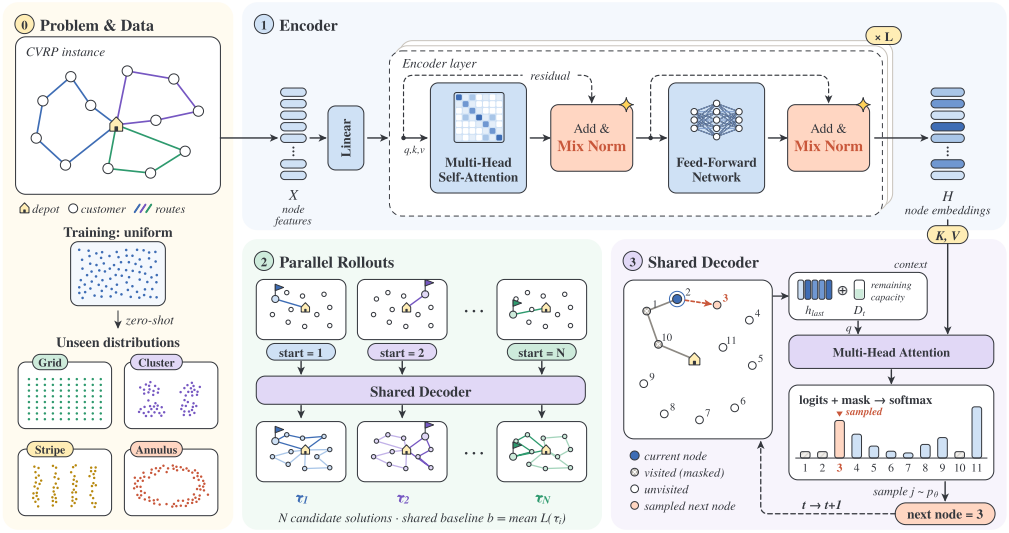

Figure 2: Overview of the MixRoute framework. (1) The encoder consists of L Transformer layers, each comprising multi-head self-attention and a feed-forward network, each followed by Add & Mix Norm. (2) parallel rollouts: each node serves as a starting point, and all trajectories are decoded in parallel. (3) The autoregressive decoder constructs the solution one node at a time.

Concrete instantiations vary in how the K distributions are leveraged, including weighted empirical risk minimization [18], multi-teacher knowledge distillation [4], and meta-learning over distribution-conditioned tasks [51], but they share a common premise: the support of the training distribution is enlarged beyond a single $\mathcal{D}_{\text{train}}$. A canonical realization is the mixture form

$$\theta^* = \arg\min_{\theta} \sum_{k=1}^{K} w_k \, \mathbb{E}_{x \sim \mathcal{D}_k} \big[\, c(\pi_\theta(x)) \, \big], \quad \sum_{k=1}^{K} w_k = 1, \; w_k \geq 0, \tag{2}$$

which subsumes Eq. (1) as the special case $K = 1$.

Remark. The two regimes also differ sharply in their underlying assumption about what is needed for cross-distribution generalization. Existing work has overwhelmingly favored multi-distribution training, building increasingly elaborate machinery on top of it, such as multi-teacher distillation, meta-learning over diverse task families, and distribution-specific architectural components, under the implicit premise that single-distribution training is fundamentally insufficient. This trend has, perhaps inadvertently, led the community to underestimate what single-distribution training alone can achieve. In this work, we revisit this premise and ask how far cross-distribution generalization can be pushed without enlarging $\mathcal{D}_{\text{train}}$, showing that with the right design, the ceiling of single-distribution training is substantially higher than previously assumed.

Overview. Figure 2 illustrates the overall framework of **MixRoute**. We follow the encoder-decoder paradigm of POMO [22], with our key modification being the replacement of the original norm module with the proposed Mix Norm. Given a routing instance with N nodes, MixRoute proceeds in three stages: (i) the encoder embeds all nodes into a shared representation through L stacked Transformer layers, where each layer consists of a multi-head self-attention block and a feed-forward network, both followed by an Add & Mix Norm module; (ii) parallel rollouts leverage the POMO multi-start strategy by treating each of the N nodes as a starting point and decoding N trajectories; (iii) the decoder autoregressively constructs each solution by attending to the encoded node representations and selecting the next node via a softmax over the unvisited candidates.

Add & Mix Norm Module. Given input embeddings $x \in \mathbb{R}^{B \times N \times D}$ (batch size B, N nodes, embedding dim D), Mix Norm produces three independently normalized representations using IN, LN, and BN, each equipped with its own learnable affine parameters:

$$\hat{x}_{\text{IN}} = \text{IN}(x), \quad \hat{x}_{\text{LN}} = \text{LN}(x), \quad \hat{x}_{\text{BN}} = \text{BN}(x). \tag{3}$$

To combine these representations, we introduce an input-adaptive gate that produces per-token mixing weights conditioned on both the token itself and a global instance context. Specifically, we compute the global context as the mean of the input across the node dimension, $c = \frac{1}{N}\sum_{n=1}^{N} x_n \in \mathbb{R}^{B\times 1\times D}$, and broadcast it back to each token. The gate then takes the concatenation $[x_n; c]$ as input and produces a 3-way distribution over the normalizers:

$$g(x_n) = \mathrm{softmax}\big(W_g[x_n; c] + b_g\big) \in \Delta^3, \quad (4)$$

where $W_g \in \mathbb{R}^{3\times 2D}$ and $b_g \in \mathbb{R}^3$ are learnable parameters and Δ^3 denotes the probability simplex. The mixed representation is then a weighted combination of the three normalized outputs:

$$\tilde{x}_n = \sum_{k\in\{\mathrm{IN,LN,BN}\}} g_k(x_n)\,\hat{x}_{k,n}. \quad (5)$$

To stabilize training and preserve the original feature when normalization is unhelpful, we introduce a learnable scalar $\alpha \in [0, 1]$ that interpolates between the input and the mixed output:

$$\mathrm{MixNorm}(x) = (1-\alpha)\,x + \alpha\,\tilde{x}. \quad (6)$$

We initialize the gate bias as $b_g = [0, 0, 1]$, which biases the gate toward BN at the start of training. The gate then adapts these weights on its own; the resulting trajectories are reported in Appendix F.

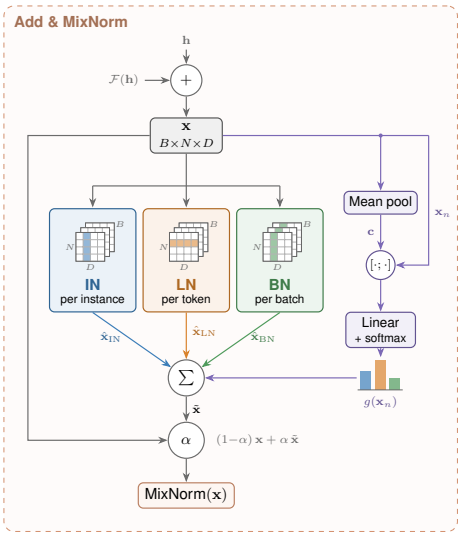

Figure 3: The Add & MixNorm module.

5 Experiments

5.1 Problem-Level Generality

Setup. We evaluate the zero-shot generalization capability of MixRoute on TSP, CVRP and CVRPTW without any fine-tuning or task-specific adaptation. The test benchmark covers 12 categories of distribution shift, comprising at least 54 datasets, all of size 100. The shifts are organized along two dimensions: node coordinate distributions and problem-specific constraint shifts, as summarized in Table 3. Each test set contains 128 instances, and the full specification of every sub-variant is detailed in Appendix C. Crucially, none of these distributions are observed during training.

Training settings. We adopt the standard POMO [22] backbone, with $N = 100$ starting nodes per instance. The encoder consists of 6 Transformer layers with embedding dimension 128, 8 attention heads; logit clipping is applied with $C = 10$. All models are trained on uniformly distributed instances with problem size 100 using the Adam optimizer (learning rate 1×10^{-4}, weight decay 1×10^{-6}), with a learning rate scheduler decaying by $\gamma = 0.1$ at epochs 6401 and 6451. Training is performed for 6500 epochs, each consisting of 10,000 instances with a batch size of 64. All experiments are conducted on a single NVIDIA RTX 3090 GPU with 24GB of memory.

Table 3: Composition of the cross-distribution benchmark across three routing problems (TSP, CVRP, CVRPTW), all of size 100. Vertical bars separate node coordinate shifts (left) from problem-specific shifts (right).

	Node coordinate distributions									Problem-specific shifts			
Task	Annulus	Beta	Cluster	Expansion	Explosion	Gaussian	Grid	Implosion	Stripe	Demand	Capacity	TW	**Total**
TSP	6	6	6	6	6	4	6	6	8	—	—	—	**54**
CVRP	6	6	6	6	6	4	6	6	8	5	5	—	**64**
CVRPTW	6	6	6	6	6	4	6	6	8	—	—	7	**61**

Baselines. We compare MixRoute against three distinct categories of baselines. For all neural baselines, we use their officially released pre-trained checkpoints. *(i) Classical solvers.* We include LKH3 [14] (max trials = 100) and additionally HGS [41] (time limit = 10s) since LKH3 fails to produce feasible solutions on a considerable fraction of CVRPTW instances. *(ii) Single-distribution neural baselines.* This category covers neural solvers trained exclusively on uniformly distributed instances. We include POMO [22], ELG [11], LEHD [27], BQ [7], and INViT [9]. Among these,

Table 4: Zero-shot generalization results across TSP, CVRP, CVRPTW. Gap (%) is computed against LKH3, except for the CVRPTW where LKH3 produces no feasible solution and HGS is used instead. The Average column is the arithmetic mean over all distributions reported. Best results among neural methods are in **bold**; second best are underlined. Time denotes the average inference time per distribution family.

	In-dist.	**Out-of-distribution: Node coordinate shifts**										
Method (TSP)	Uniform	Annulus	Beta	Cluster	Expansion	Explosion	Gaussian	Grid	Implosion	Stripe	**Avg**	**Time**
LKH3	—	—	—	—	—	—	—	—	—	—	—	4.31s
Single-distribution training												
ELG	0.66%	0.79%	1.03%	2.45%	0.79%	0.57%	1.03%	1.12%	0.96%	1.70%	1.11%	3.26s
LEHD	0.57%	0.62%	5.71%	1.62%	0.64%	0.50%	**0.73%**	0.94%	**0.69%**	5.03%	1.71%	3.82s
INViT	3.80%	4.30%	4.15%	5.90%	3.88%	3.85%	4.13%	3.51%	4.37%	5.36%	4.33%	13.99m
Multi-distribution training												
AMDKD-AM	3.84%	6.36%	11.91%	12.74%	5.21%	3.70%	4.75%	4.33%	4.54%	12.07%	6.95%	0.98s
AMDKD-POMO	1.02%	1.49%	**0.63%**	**1.12%**	0.81%	0.84%	0.94%	1.43%	0.89%	1.35%	1.05%	0.97s
POMO	0.88%	1.13%	1.86%	3.89%	1.30%	0.77%	1.55%	1.28%	1.30%	2.99%	1.70%	0.74s
MixRoute	**0.54%**	**0.52%**	0.69%	2.12%	**0.61%**	**0.45%**	0.91%	**0.93%**	0.78%	**1.14%**	**0.87%**	0.75s

	In-dist.	**Out-of-distribution: Node coordinate shifts**									**CVRP-specific shifts**			
Method (CVRP)	Uniform	Annulus	Beta	Cluster	Expansion	Explosion	Gaussian	Grid	Implosion	Stripe	Demand	Capacity	**Avg**	**Time**
LKH3	—	—	—	—	—	—	—	—	—	—	—	—	—	15.41s
Single-distribution training														
ELG	-0.38%	0.31%	2.10%	3.52%	0.85%	0.18%	2.16%	1.65%	0.29%	3.02%	1.94%	4.89%	1.71%	4.18s
LEHD	2.77%	0.61%	3.92%	3.96%	0.45%	0.33%	3.19%	2.60%	0.15%	2.92%	2.88%	3.97%	2.31%	4.44s
BQ	0.02%	1.00%	4.83%	2.90%	1.36%	0.80%	2.91%	2.67%	0.76%	3.02%	2.92%	5.18%	2.36%	2.41m
INViT	6.13%	6.65%	10.76%	10.60%	6.70%	7.15%	9.75%	7.60%	6.38%	10.45%	9.69%	11.73%	8.63%	15.01m
Multi-distribution training														
AMDKD-AM	2.15%	2.59%	5.22%	6.88%	3.34%	2.88%	5.97%	4.14%	3.23%	4.91%	4.78%	7.12%	4.43%	1.28s
AMDKD-POMO	-0.83%	1.13%	1.83%	2.37%	0.13%	-0.01%	1.79%	1.64%	-0.25%	1.77%	1.87%	5.53%	1.41%	1.28s
LEHD-VCT+MEM	0.69%	0.27%	3.68%	3.11%	1.89%	1.81%	2.75%	2.13%	0.96%	2.66%	2.59%	**1.68%**	2.02%	8.06s
POMO	-0.93%	7.11%	2.95%	3.25%	0.65%	-0.19%	1.72%	1.38%	-0.15%	8.30%	1.49%	8.34%	2.83%	2.03s
MixRoute	**-1.03%**	**-0.87%**	**1.11%**	**1.82%**	**-0.13%**	**-0.28%**	**1.36%**	**1.18%**	**-0.30%**	**1.13%**	**1.24%**	3.36%	**0.72%**	2.30s

	In-dist.	**OOD: Node coordinate shifts**									**Time Window shifts**				
Method (CVRPTW)	Uniform	Annulus	Beta	Cluster	Expansion	Explosion	Gaussian	Grid	Implosion	Stripe	Width	Timing	Coupling	**Avg**	**Time**
HGS	—	—	—	—	—	—	—	—	—	—	—	—	—	—	763.79s
LEHD-VCT+MEM	19.39%	23.44%	22.60%	24.16%	22.15%	19.71%	23.44%	19.10%	19.15%	23.53%	12.82%	-4.70%	18.52%	18.72%	1.69s
POMO	3.57%	4.99%	5.37%	5.86%	4.72%	3.77%	4.16%	3.76%	4.01%	5.02%	**7.73%**	0.00%	3.83%	4.37%	3.90s
MixRoute	**3.19%**	**4.01%**	**4.42%**	**2.45%**	**3.61%**	**3.56%**	**2.20%**	**3.47%**	**3.68%**	**3.80%**	12.55%	**-15.98%**	**3.50%**	**2.65%**	4.02s

POMO is the most direct baseline, as MixRoute is built directly on top of POMO. *(iii) Multi-distribution neural baselines.* To contextualize the strength of our single-source training protocol, we further compare against neural solvers that explicitly leverage distributionally diverse training data. We include AMDKD [4] with its two reported variants AMDKD-AM and AMDKD-POMO, and LEHD-VCT+MEM [26]. The first three are trained on multiple node coordinate distributions, while LEHD-VCT+MEM is instead trained on instances with varying capacity and time window values.

Results. Table 4 reports zero-shot generalization results on the TSP, CVRP, and CVRPTW benchmarks, respectively. Across all three problems, MixRoute achieves the lowest average gap among neural methods, and consistently improves over POMO across almost all distribution shifts.

5.2 Architecture-Level Generality

To verify the generality of our framework across different architectures, we apply MIXNORM to MatNet [23] and use it to solve the Asymmetric CVRP (ACVRP). MixRoute retains this exact setup and only replaces the standard Add & Norm sub-layer with our Add & MixNorm module. It is worth noting that all type-A instances are post-processed with the triangle inequality constraint, which sufficiently homogenizes the resulting distance matrices and removes meaningful distributional shift along this axis. We accordingly restrict our evaluation to the demand axis.

As shown in Table 5, MixRoute outperforms MatNet on all five demand distributions. The improvement also holds on the in-distribution setting. These gains incur negligible overhead in inference time (0.65s to 0.69s), confirming that MixNorm transfers to a different backbone as a drop-in module.

Table 5: Generalization performance on ACVRP100 across different demand distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**. Time is the total inference time over the test set.

Method	In-dist.	Out-of-distribution: Demand distribution shifts					Average Gap	Time
	Uniform	Small-biased	Large-biased	U-shaped	Mid-peak	Bimodal		
LKH3	—	—	—	—	—	—	—	1.63h
MatNet	-1.49%	1.90%	-3.92%	-0.82%	-1.24%	-0.78%	-1.06%	0.65s
MixRoute	**-2.02%**	**1.06%**	**-4.58%**	**-1.12%**	**-1.74%**	**-1.35%**	**-1.63%**	0.69s

5.3 Normalization-Level Generality

To probe this, we evaluate MixRoute on the CVRP benchmark under three different normalization combinations (IN+LN+NoNorm, IN+LN+BN, and IN+LN+RMSNorm [48]), chosen to span normalizations of qualitatively different characters: identity (no normalization), batch-level statistics, and root-mean-square scaling without mean centering.

As shown in Table 6, this is precisely what we observe. All three MixRoute variants substantially outperform POMO, reducing the average gap from 3.17% to between 0.88% and 1.41%. The IN+LN+BN variant performs marginally best (0.88% Avg), but the differences among the three variants are small relative to the gap between any of them and POMO. This pattern indicates that MIXROUTE's effectiveness is robust to the constituent choice.

Table 6: Robustness of Mix Norm to the choice of constituent normalizations. We compare three MixRoute variants with different normalization combinations against the POMO baseline. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined. Shaded cells indicate improvement over POMO. Time denotes the average inference time per distribution family.

Method	Norms	Node coordinate distributions									CVRP-specific		Avg	Time
		Annulus	Beta	Cluster	Expansion	Explosion	Gaussian	Grid	Implosion	Stripe	Demand	Capacity		
LKH3	—	—	—	—	—	—	—	—	—	—	—	—	—	—
POMO	IN (original)	7.11%	2.95%	3.25%	0.65%	-0.19%	1.72%	1.38%	-0.15%	8.30%	1.49%	8.34%	3.17%	2.03s
MixRoute	IN+LN+NoNorm	**-0.91%**	**1.01%**	1.97%	-0.16%	-0.32%	1.52%	1.20%	-0.33%	**1.08%**	**1.21%**	4.22%	0.95%	2.18s
MixRoute	IN+LN+BN	-0.87%	1.11%	**1.82%**	-0.13%	-0.28%	**1.36%**	**1.18%**	-0.30%	1.13%	1.24%	**3.36%**	**0.88%**	2.30s
MixRoute	IN+LN+RMSNorm	-0.51%	1.58%	2.75%	**-0.18%**	**-0.47%**	1.82%	1.43%	**-0.38%**	1.92%	1.73%	5.79%	1.41%	2.21s

5.4 Generalization to Real World Benchmark

We further evaluate MixRoute on CVRPLIB [39], a widely used real-world benchmark whose instances are drawn from practical routing applications. CVRPLIB instances exhibit heterogeneous customer layouts, demand patterns, and capacity constraints that are not generated from any parametric distribution, and their sizes range from 15 to 1000 nodes. We evaluate on seven sets using the same checkpoints as in Section 5.1 without fine-tuning. As shown in Figure 4, MixRoute improves over POMO on every set, reducing the overall gap from 11.80% to 7.94%.

5.5 Analysis

5.5.1 What Matters is Mixing: Robustness Across Variants

We categorize mixing strategies into two types based on what is being mixed: *output-level mixing*, which combines the outputs of multiple normalization layers via learnable weights, and *statistic-level mixing*, which fuses the underlying normalization statistics before applying a single affine transformation. Full descriptions of each mixing scheme are provided in Appendix B.2.

For each variant, we further examine two combinations of constituent normalizations: `IN+LN+NoNorm` and `IN+LN+BN`. The first combination treats `NoNorm` as a constituent; the second replaces it with the standard Batch Normalization. Table 7 shows that all mixing variants substantially outperform POMO (Avg 3.17%), with the worst-performing variant (Switch Norm IN+LN, 1.68%) still nearly halving POMO's gap. This confirms that the gain originates from the act of mixing itself.

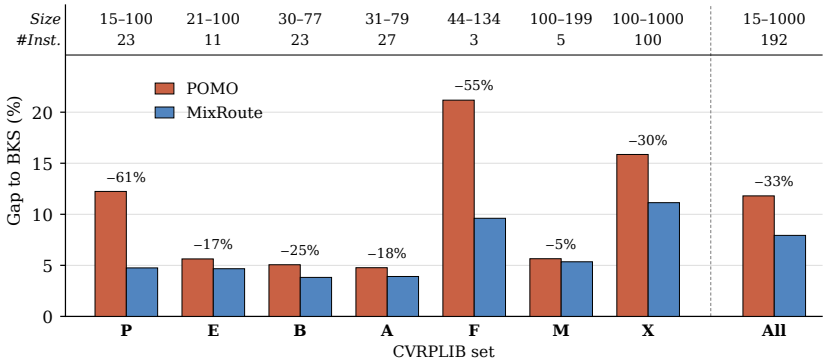

Figure 4: Zero-shot generalization of POMO and MixRoute on seven CVRPLIB sets. Bars report the average optimality gap (%) with respect to the best known solutions (BKS). The number above each pair of bars is the relative gap reduction of MixRoute over POMO.

Table 7: Ablation on Mix Norm variants. Best results are in **bold**; second best are underlined. Color depth indicates improvement magnitude over POMO: light (<1%), medium (1–5%), and dark (≥5%).

Method	Norms	Node coordinate distributions									CVRP-specific		Avg
		Annulus	Beta	Cluster	Expansion	Explosion	Gaussian	Grid	Implosion	Stripe	Demand	Capacity	
LKH3	—	—	—	—	—	—	—	—	—	—	—	—	—
POMO	IN (original)	7.11%	2.95%	3.25%	0.65%	-0.19%	1.72%	1.38%	-0.15%	8.30%	1.49%	8.34%	3.17%
Statistic mixing													
Switch Norm	IN+LN	-0.67%	1.76%	2.47%	0.02%	-0.20%	1.51%	1.24%	-0.17%	1.39%	1.41%	9.72%	1.68%
Switch Norm	IN+LN+BN	-0.68%	1.17%	1.90%	-0.11%	-0.28%	1.45%	**1.18%**	-0.30%	1.57%	1.22%	8.98%	1.46%
Output mixing													
Simple Mixing	IN+LN+NoNorm	-0.70%	1.41%	2.22%	-0.01%	-0.24%	1.45%	1.26%	-0.24%	1.76%	1.32%	5.43%	1.24%
Simple Mixing	IN+LN+BN	-0.52%	1.96%	2.53%	**-0.27%**	**-0.47%**	1.60%	1.31%	**-0.45%**	1.84%	1.49%	4.15%	1.20%
MixRoute (ours)	IN+LN+NoNorm	**-0.91%**	**1.01%**	1.97%	-0.16%	-0.32%	1.52%	1.20%	-0.33%	**1.08%**	**1.21%**	4.22%	0.95%
MixRoute (ours)	IN+LN+BN	-0.87%	1.11%	**1.82%**	-0.13%	-0.28%	**1.36%**	**1.18%**	-0.30%	1.13%	1.24%	**3.36%**	**0.88%**

5.6 Additional Results

Due to space constraints, we defer a number of supplementary experiments and analyses to the appendix. Table 8 summarizes their content and corresponding location for reference.

Table 8: Overview of additional results deferred to the appendix.

Topic	Section	Content
Multi-seed results	Appendix D	Seed variability of MixRoute.
Mechanism of MixNorm	Appendix E	How MixNorm responds to distributional shift.
Behavior of the adaptive gate	Appendix F	Gate weights across layers and distributions.
Related normalizations	Appendix G	Comparison with existing Mix Norm.
Normalizer choice	Appendix H	Effect of pool size and composition.

6 Conclusion

We revisit cross-distribution generalization in neural routing solvers from a perspective that has received little attention: the role of normalization. While prior work has largely assumed single-distribution training to be fundamentally insufficient, pursuing increasingly elaborate multi-distribution machinery, we show that its ceiling is, in fact, substantially higher than previously assumed. By identifying normalization layers as a key source of distribution-specific entanglement and proposing MixNorm, a simple plug-in module combining complementary normalization schemes, we demonstrate that strong cross-distribution generalization is achievable from a single uniform training distribution alone. Extensive experiments across three routing problems, four shift dimensions, and 179 out-of-distribution settings confirm that MixRoute consistently improves diverse

backbones, validating it as an architecture-agnostic and broadly applicable component. We hope this work encourages the community to reconsider single-distribution training not as a limitation to circumvent, but as a stress test that exposes the true inductive biases of a solver.

Acknowledgments

This research is supported by the Ministry of Education, Singapore under its Academic Research Fund (AcRF) Tier 2 Grant Call (Proposal ID: T2EP20225-0056). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore. This research is also supported by the Lee Kong Chian Fellowship awarded to CAO Zhiguang by Singapore Management University.

References

[1] Lei Jimmy Ba, J. Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv.org*, 2016.

[2] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[3] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[4] Jieyi Bi, Yining Ma, Jiahai Wang, Zhiguang Cao, Jinbiao Chen, Yuan Sun, and Yeow Meng Chee. Learning generalizable models for vehicle routing problems via knowledge distillation. *Advances in Neural Information Processing Systems*, 35:31226–31238, 2022.

[5] Jakob Bossek, Pascal Kerschke, Aneta Neumann, Markus Wagner, Frank Neumann, and Heike Trautmann. Evolving diverse tsp instances by means of novel and creative mutation operators. In *Foundations of Genetic Algorithms*, 2019. URL `https://api.semanticscholar.org/CorpusID:195470769`.

[6] Diego Cattaruzza, Nabil Absi, Dominique Feillet, and Jesús González-Feliu. Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, 6(1):51–79, 2017.

[7] Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:77416–77429, 2023.

[8] Xinjie Fan, Qifei Wang, Junjie Ke, Feng Yang, Boqing Gong, and Mingyuan Zhou. Adversarially adaptive normalization for single domain generalization. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8204–8213, 2021. URL `https://api.semanticscholar.org/CorpusID:235313639`.

[9] Han Fang, Zhihao Song, Paul Weng, and Yutong Ban. Invit: A generalizable routing problem solver with invariant nested view transformer. In *International Conference on Machine Learning*, 2024. URL `https://api.semanticscholar.org/CorpusID:267413113`.

[10] Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 7474–7482, 2021.

[11] Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 6914–6922, 2024.

[12] Simon Geisler, Johanna Sommer, Jan Schuchardt, Aleksandar Bojchevski, and Stephan Günnemann. Generalization of neural combinatorial solvers through the lens of adversarial robustness. In *International Conference on Learning Representations*.

[13] Shuangchun Gui, Zhiguang Cao, Wen Song, and Yew-Soon Ong. Vision-assisted foundation model for solving multitask vehicle routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2026.

[14] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European journal of operational research*, 126(1):106–130, 2000.

[15] André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. In *ECAI 2020*, pages 443–450. IOS Press, 2020.

[16] Ziwei Huang, Jianan Zhou, Zhiguang Cao, and Yixin Xu. Rethinking light decoder-based solvers for vehicle routing problems. *arXiv preprint arXiv:2503.00753*, 2025.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 2015. URL `https://api.semanticscholar.org/CorpusID:5808102`.

[18] Yuan Jiang, Yaoxin Wu, Zhiguang Cao, and Jie Zhang. Learning to solve routing problems via distributionally robust optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 9786–9794, 2022.

[19] Chaitanya K Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, 27(1):70–98, 2022.

[20] Grigorios D Konstantakopoulos, Sotiris P Gayialis, and Evripidis P Kechagias. Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification: Gd konstantakopoulos et al. *Operational research*, 22(3):2033–2062, 2022.

[21] W. Kool, H. V. Hoof, and Max Welling. Attention, learn to solve routing problems! 2018.

[22] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in neural information processing systems*, 33:21188–21198, 2020.

[23] Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 34:5138–5149, 2021.

[24] Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[25] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. *arXiv preprint arXiv:2401.02051*, 2024.

[26] Fu Luo, Yaoxin Wu, Zhi Zheng, and Zhenkun Wang. Rethinking neural combinatorial optimization for vehicle routing problems with different constraint tightness degrees. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[27] Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36:8845–8864, 2023.

[28] Ping Luo, Jiamin Ren, and Zhanglin Peng. Differentiable learning-to-normalize via switchable normalization. *ArXiv*, abs/1806.10779, 2018. URL `https://api.semanticscholar.org/CorpusID:49532099`.

[29] Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. *Advances in Neural Information Processing Systems*, 34:11096–11107, 2021.

[30] Sahil Manchanda, Sofia Michel, Darko Drakulic, and Jean-Marc Andreoli. On the generalization of neural combinatorial optimization heuristics. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 426–442. Springer, 2022.

[31] Abood Mourad, Jakob Puchinger, and Chengbin Chu. A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B: Methodological*, 123:323–346, 2019.

[32] Hyeonseob Nam and Hyo-Eun Kim. Batch-instance normalization for adaptively style-invariant neural networks. *ArXiv*, abs/1805.07925, 2018. URL `https://api.semanticscholar.org/CorpusID:29165066`.

[33] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *European Conference on Computer Vision*, 2018. URL `https://api.semanticscholar.org/CorpusID:50781790`.

[34] Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. Dimes: A differentiable meta solver for combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 35: 25531–25546, 2022.

[35] Seonguk Seo, Yumin Suh, Dongwan Kim, Jongwoo Han, and Bohyung Han. Learning to optimize domain specific normalization for domain generalization. *ArXiv*, abs/1907.04275, 2019. URL `https://api.semanticscholar.org/CorpusID:195848051`.

[36] Wenqi Shao, Tianjian Meng, Jingyu Li, Ruimao Zhang, Yudian Li, Xiaogang Wang, and Ping Luo. Ssn: Learning sparse switchable normalization via sparsestmax. *CVPR*, 2019.

[37] Jiwoo Son, Minsu Kim, Hyeonah Kim, and Jinkyoo Park. Meta-sage: Scale meta-learning scheduled adaptation with guided exploration for mitigating scale shift on combinatorial optimization. In *International Conference on Machine Learning*, pages 32194–32210. PMLR, 2023.

[38] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[39] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.

[40] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *ArXiv*, abs/1607.08022, 2016. URL `https://api.semanticscholar.org/CorpusID:16516553`.

[41] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.

[42] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

[43] Chenguang Wang, Zhouliang Yu, Stephen McAleer, Tianshu Yu, and Yaodong Yang. Asp: Learn a universal neural solver! *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46 (6):4102–4114, 2024.

[44] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 2004. URL `https://api.semanticscholar.org/CorpusID:2332513`.

[45] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving routing problems. *IEEE transactions on neural networks and learning systems*, 33(9):5057–5069, 2021.

[46] Yubin Xiao, Di Wang, Xuan Wu, Yuesong Wu, Boyang Li, Wei Du, Liupu Wang, and You Zhou. Improving generalization of neural vehicle routing problem solvers through the lens of model architecture. *Neural Networks*, 187:107380, 2025.

[47] Hang Yi, Ziwei Huang, Yining Ma, and Zhiguang Cao. Radar: Learning to route with asymmetry-aware distance representations. In *International Conference on Learning Representations (ICLR)*, 2026.

[48] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *ArXiv*, abs/1910.07467, 2019. URL https://api.semanticscholar.org/CorpusID:113405151.

[49] Ni Zhang, Zhiguang Cao, Jianan Zhou, Cong Zhang, and Yew-Soon Ong. An agentic framework with llms for solving complex vehicle routing problems. In *International Conference on Learning Representations*, volume 2026, pages 114166–114206, 2026.

[50] C Zhou, X Lin, Z Wang, X Tong, M Yuan, and Q Zhang. Instance-conditioned adaptation for large-scale generalization of neural combinatorial optimization. arxiv. *arXiv preprint arXiv:2405.01906*, 2024.

[51] Jianan Zhou, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Towards omni-generalizable neural methods for vehicle routing problems. In *International conference on machine learning*, pages 42769–42789. PMLR, 2023.

[52] Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Chi Xu. Mvmoe: Multi-task vehicle routing solver with mixture-of-experts. *ArXiv*, abs/2405.01029, 2024. URL https://api.semanticscholar.org/CorpusID:269502224.

MixRoute: Rethinking Single-Distribution Training for Generalizable Neural Routing (Appendix)

A Preliminaries

A.1 Vehicle Routing Problems

We consider three representative VRP variants: the Traveling Salesman Problem (TSP), the Capacitated VRP (CVRP), and the Capacitated VRP with Time Windows (CVRPTW). All are defined on a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_0, v_1, \ldots, v_N\}$ consists of a depot v_0 and N customer nodes (in the case of TSP, v_0 is the starting node and there is no notion of a depot), and each edge is associated with a Euclidean travel cost c_{ij}. For CVRP and CVRPTW, a fleet of vehicles, each with capacity Q, is stationed at the depot.

TSP. A feasible solution $\boldsymbol{\tau}$ is a single Hamiltonian tour that visits every node exactly once and returns to the starting node, with the objective of minimizing the total tour length.

CVRP. Each customer v_i has a demand $d_i \in (0, Q]$. A feasible solution $\boldsymbol{\tau}$ is a set of routes that partitions the customer set, where each route starts and ends at the depot, visits each assigned customer exactly once, and respects the capacity constraint. The objective is to minimize the total travel cost

$$\mathcal{L}(\boldsymbol{\tau}) = \sum_{(v_i, v_j) \in \boldsymbol{\tau}} c_{ij}. \tag{7}$$

CVRPTW. CVRPTW extends CVRP by additionally associating each customer v_i with a service time s_i and a time window $[e_i, l_i]$, within which service must commence. Vehicles arriving before e_i must wait until the window opens, while arrivals after l_i render the route infeasible.

A.2 Reinforcement Learning Formulation

Following the standard practice in neural combinatorial optimization [21, 22], we formulate VRPs as a sequential decision process and learn a construction policy via reinforcement learning. Given an instance $\mathcal{I}$, the policy $\pi_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ constructs a solution $\boldsymbol{\tau}$ autoregressively:

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{\tau} \mid \mathcal{I}) = \prod_{t=1}^{T} \pi_{\boldsymbol{\theta}}(a_t \mid s_t), \tag{8}$$

where at each step t, the action a_t selects the next node to visit based on the current state s_t (i.e., the partial route, remaining capacity, and elapsed time). The reward is defined as the negative tour length, $R(\boldsymbol{\tau}) = -\mathcal{L}(\boldsymbol{\tau})$, so that maximizing expected reward is equivalent to minimizing total travel cost.

The policy is trained with the REINFORCE algorithm [44] using a baseline $b(\mathcal{I})$ to reduce variance:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} \left[\big(R(\boldsymbol{\tau}) - b(\mathcal{I}) \big) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau} \mid \mathcal{I}) \right]. \tag{9}$$

Following POMO [22], we exploit the symmetry of routing solutions by sampling M trajectories with different starting nodes from the same instance and using their mean reward as a shared baseline.

B Additional Experimental Details

This section provides supplementary tables that complement the main paper. Specifically, we include detailed results that could not fit within the main paper due to space constraints, organized to mirror the corresponding sections in the main text. Each table is referenced from the main paper at the relevant point of discussion.

Table 9: Comparison with single normalizations on the CVRP benchmark. We compare POMO equipped with each individual normalization (IN, LN, NoNorm, BN) against MixRoute variants. Best results among neural methods are in **bold**; second best are underlined. Blue cells indicate where MixRoute (IN+LN+NoNorm) outperforms all three single normalizations (IN, LN, NoNorm); orange cells indicate where MixRoute (IN+LN+BN) outperforms all three single normalizations (IN, LN, BN).

		Node coordinate distributions									CVRP-specific		
Method	**Norms**	Annulus	Beta	Cluster	Expansion	Explosion	Gaussian	Grid	Implosion	Stripe	Demand	Capacity	**Avg**
LKH3	—	—	—	—	—	—	—	—	—	—	—	—	—
POMO	IN (original)	7.11%	2.95%	3.25%	0.65%	-0.19%	1.72%	1.38%	-0.15%	8.30%	1.49%	8.34%	3.17%
POMO	LN	-0.38%	4.09%	2.85%	0.03%	-0.20%	2.14%	1.43%	-0.28%	2.25%	1.59%	5.79%	1.76%
POMO	NoNorm	-0.61%	1.68%	3.24%	-0.11%	-0.27%	2.73%	1.34%	-0.30%	2.01%	1.47%	4.35%	1.41%
MixRoute	IN+LN+NoNorm	**-0.91%**	**1.01%**	1.97%	**-0.16%**	**-0.32%**	1.52%	1.20%	**-0.33%**	**1.08%**	**1.21%**	4.22%	0.95%
POMO	BN	-0.76%	1.26%	2.29%	-0.09%	-0.20%	1.46%	**1.16%**	-0.22%	1.24%	1.28%	4.19%	1.06%
MixRoute	IN+LN+BN	-0.87%	1.11%	**1.82%**	-0.13%	-0.28%	**1.36%**	1.18%	-0.30%	1.13%	1.24%	**3.36%**	**0.88%**

B.1 Comparison with Single Normalizations

We further compare MixRoute against single-normalization baselines to isolate the gain of mixing from any particular constituent. We replace the Add & Norm module in POMO with each individual normalization—NoNorm (an identity mapping with affine transformation), Instance Normalization (IN), Layer Normalization (LN), and Batch Normalization (BN)—and compare them against the two MixRoute variants `IN+LN+NoNorm` and `IN+LN+BN`. All models share the same architecture and training protocol; the only difference lies in the normalization layer.

Results. Table 9 shows that no single normalization dominates across all distributions: each of IN, LN, NoNorm, and BN performs well on some categories but poorly on others. In contrast, the two MixRoute variants achieve the lowest average gaps.

B.2 Mixing Types

Two types of mixing. We categorize mixing strategies into two types based on what is being mixed.

(i) Output mixing performs each normalization independently and then linearly combines the outputs:

$$\text{OutputMix}(x) = \sum_{i=1}^{K} \alpha_i \cdot \text{Norm}_i(x), \tag{10}$$

where each $\text{Norm}_i(\cdot)$ produces a fully normalized output, and $\{\alpha_i\}_{i=1}^{K}$ are mixing coefficients with $\sum_i \alpha_i = 1$. Output mixing preserves the complete transformation of every constituent normalization.

(ii) Statistic mixing, in contrast, fuses the normalizations at a more fundamental level by combining their statistics (mean and variance) prior to a single normalization step:

$$\text{StatMix}(x) = \frac{x - \sum_{i=1}^{K} \alpha_i \, \mu_i}{\sqrt{\sum_{i=1}^{K} \alpha_i \, \sigma_i^2 + \epsilon}}, \tag{11}$$

where μ_i and σ_i^2 denote the mean and variance computed under the i-th normalization scheme. Statistic mixing produces a single normalized output whose statistics interpolate between the constituents.

Mixing variants. Within these two paradigms, we evaluate three representative variants:

- **MixRoute (ours)** — output mixing with learnable, input-adaptive weights. The mixing coefficients $\alpha_i(x)$ are produced by a lightweight gating network conditioned on the input.
- **Simple Mixing** — output mixing with fixed equal weights. We set $\alpha_i = 1/K$ for all i, removing all learnable parameters from the mixing module.
- **Switch Norm** [28] — statistic mixing with learnable scalar weights. Originally proposed for image recognition, Switch Norm learns a single set of scalar coefficients (shared across the input) to combine the means and variances of multiple normalization schemes.

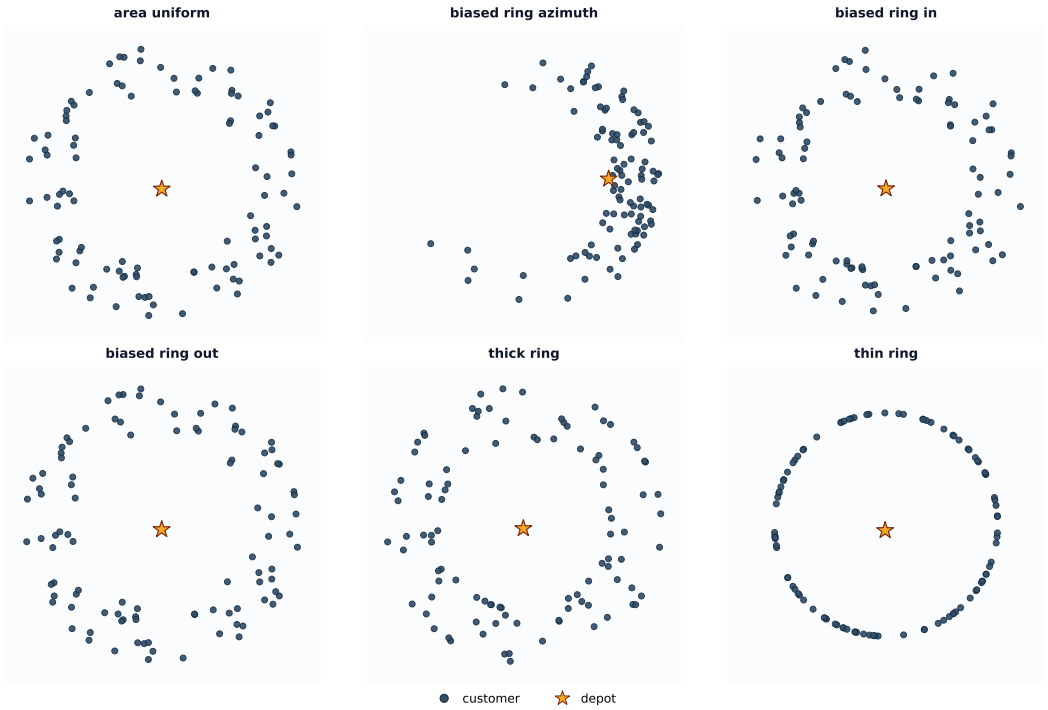

Figure 5: Visualization of the six Annulus distribution variants on CVRP100 instances. The orange star marks the depot at the center.

C Benchmark Details and Full Results

In this section, we provide a detailed description of our cross-distribution generalization benchmark. Given that fully reporting per-variant results across all three problems would consume a prohibitive amount of space, we focus the detailed presentation on the CVRP benchmark, which spans the broadest set of shift dimensions in our evaluation. Training MIXROUTE for 6500 epochs takes approximately 66.97 hours on TSP, 91.15 hours on CVRP, and 106.49 hours on CVRPTW, all measured on a single NVIDIA RTX 3090 GPU. Throughout this appendix, the Average Gap column of each family is computed by pooling all instances of that family.

C.1 Annulus Distribution

The Annulus distribution samples node coordinates from an annular (ring-shaped) region surrounding a central depot, capturing real-world scenarios where customers are distributed around but not too close to a logistics hub—for example, suburban delivery zones surrounding a city center, or service areas around a transportation terminal. We construct six variants that vary in radial thickness, radial density, and angular density. Figure 5 visualizes one representative instance from each variant.

Area-uniform. Coordinates are sampled uniformly over the annular region, serving as the unbiased baseline of this family.

Biased ring azimuth. The angular coordinate is concentrated within a specific sector rather than spread uniformly, modeling one-sided coastal cities or service regions bounded by a natural barrier.

Biased ring in. Radial sampling skewed toward the inner boundary, producing customer density that thins out with distance from the depot.

Biased ring out. The inverse of *Biased ring in*: customers concentrate near the outer perimeter, with sparser nodes near the depot.

Thick ring. A wide annulus that interpolates toward a near-uniform disk distribution with only a small central exclusion zone.

Table 10: Generalization performance on CVRP100 across different Annulus distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined.

Method	Annulus distribution variants						Average Gap	Time
	Area-uniform	Thin ring	Thick ring	Biased-in	Biased-out	Biased-azimuth		
LKH3	—	—	—	—	—	—	—	7.74s
Single-distribution training								
ELG	-0.10%	-2.44%	-0.21%	0.08%	-0.31%	6.40%	0.31%	4.35s
LEHD	0.51%	-0.90%	0.20%	0.07%	0.16%	4.71%	0.61%	4.27s
BQ	0.92%	-1.27%	1.69%	1.37%	1.33%	**1.94%**	1.00%	2.30m
INViT	5.95%	3.42%	6.48%	6.08%	6.58%	12.96%	6.65%	15.20m
Multi-distribution training								
AMDKD-AM	2.76%	-0.74%	2.40%	2.52%	2.28%	7.43%	2.59%	1.01s
AMDKD-POMO	-0.04%	5.06%	0.17%	0.41%	-0.01%	2.01%	1.13%	1.11s
LEHD-VCT+MEM	0.12%	-1.39%	0.04%	-0.07%	0.02%	3.83%	0.27%	8.12s
POMO	-0.50%	43.25%	-0.50%	-0.28%	-0.63%	5.89%	7.11%	2.06s
MixRoute	**-0.88%**	**-3.26%**	**-0.95%**	**-0.74%**	**-0.99%**	2.31%	**-0.87%**	2.27s

Thin ring. A narrow annulus concentrating all nodes along a thin circular band, approximating a one-dimensional ring.

Results. Table 10 reports the per-variant performance on the Annulus family. MIXROUTE achieves the lowest gap on five of the six variants and an average gap of -0.87%, surpassing LKH3 in aggregate. The most striking case is *Thin ring*, where POMO catastrophically degrades to 43.25% while MIXROUTE attains -3.26% under identical training conditions—a 46.5-point improvement that highlights how normalization design alone can recover generalization on extreme one-dimensional geometries.

C.2 Beta Distribution

The Beta distribution family generates node coordinates by independently sampling each axis from a Beta distribution Beta(α, β), with shape parameters tuned to concentrate density near the boundaries, the corners, or the center of the unit square—capturing real-world scenarios where customers are systematically biased toward city peripheries, specific regional sectors, or downtown cores. We construct six variants along two orthogonal axes: *spatial bias type* (edge, corner, center) and *bias intensity* (normal, strong). Figure 6 visualizes one representative instance from each variant.

Edge. A U-shaped marginal density that concentrates customers along the four edges of the unit square, modeling peripheral routes such as ring roads or coastal lines.

Strong edge. A more extreme version of *Edge*, with most nodes confined to a narrow band along the boundaries.

Corner. Asymmetric Beta parameters push density toward one specific corner of the unit square, modeling deliveries localized to a specific geographic sector relative to the depot.

Strong corner. A tighter concentration in the target corner, with only sparse outliers elsewhere.

Center. A bell-shaped marginal density concentrating customers around the unit-square center, modeling downtown-style routing in a central business district.

Strong center. Sharper concentration around the center, with very few outliers along the periphery.

Results. Table 11 reports the per-variant performance on the Beta family. MIXROUTE achieves the lowest gap on five of the six variants and an average gap of 1.11%, outperforming the second-best baseline (AMDKD-POMO, 1.83%). The exception is *Strong corner*, where AMDKD-POMO (1.43%) edges out MIXROUTE (2.74%); we attribute this to AMDKD-POMO's exposure to corner-biased distributions during multi-distribution training.

C.3 Cluster Distribution

The Cluster distribution family generates node coordinates from a mixture of Gaussian components, modeling routing scenarios where customers form spatial groupings such as residential neighborhoods, commercial districts, or industrial parks. We construct six variants that vary along distinct cluster-

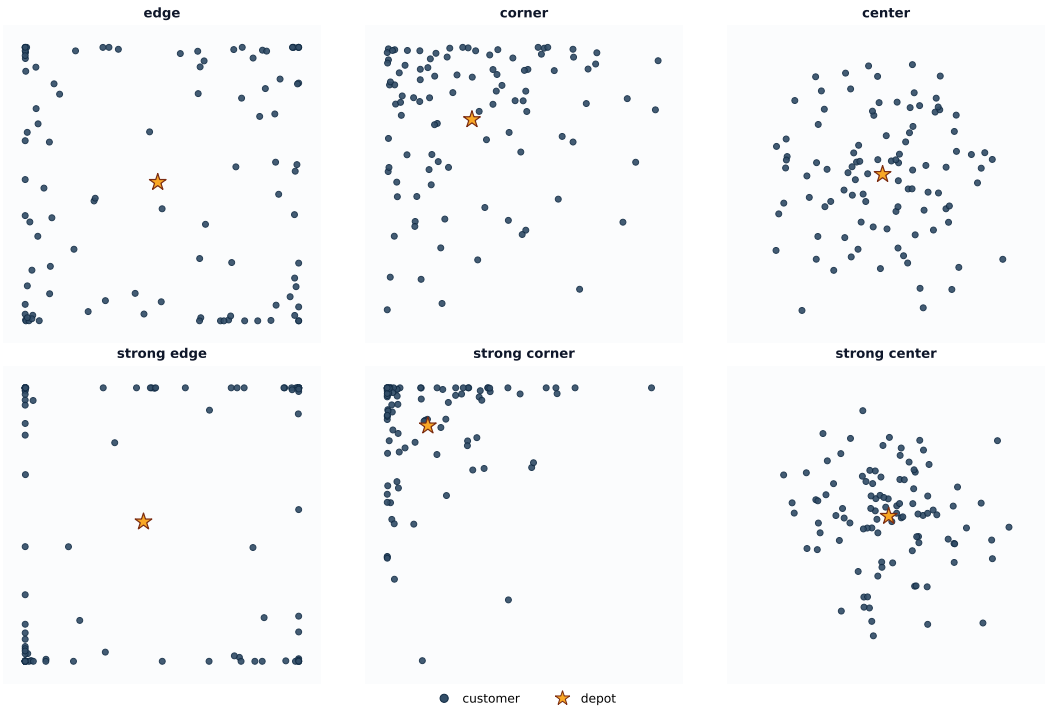

Figure 6: Visualization of the six Beta distribution variants on CVRP100 instances. The orange star marks the depot. Top row: standard intensity; bottom row: strong intensity.

Table 11: Generalization performance on CVRP100 across different Beta distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined.

Method	Beta distribution variants						Average Gap	Time
	Edge	Corner	Center	Strong edge	Strong corner	Strong center		
LKH3	—	—	—	—	—	—	—	10.90s
Single-distribution training								
ELG	1.21%	2.05%	1.87%	1.28%	5.59%	2.28%	2.10%	4.18s
LEHD	2.85%	3.17%	3.06%	5.94%	4.52%	3.45%	3.92%	4.20s
BQ	5.04%	3.25%	2.38%	8.87%	4.09%	2.33%	4.83%	2.30m
INViT	9.70%	11.46%	9.34%	10.44%	15.26%	9.78%	10.76%	15.2m
Multi-distribution training								
AMDKD-AM	3.82%	5.50%	4.70%	4.27%	9.47%	5.84%	5.22%	1.06s
AMDKD-POMO	1.38%	1.64%	1.99%	2.14%	**1.43%**	2.53%	1.83%	1.05s
LEHD-VCT+MEM	2.14%	2.93%	2.17%	6.64%	4.69%	2.61%	3.68%	8.13s
POMO	0.81%	1.81%	1.42%	2.60%	12.29%	1.92%	2.95%	1.98s
MixRoute	**0.49%**	**1.60%**	**1.16%**	**0.33%**	2.74%	**1.44%**	**1.11%**	2.23s

level dimensions: the number, size, balance, overlap, shape, and connectivity of clusters. Figure 7 visualizes one representative instance from each variant.

Islands. A balanced multi-cluster pattern with $K = 8$ tight clusters of equal population ($\sigma = 0.035$), resembling an archipelago of well-separated dense pockets.

Mega. A few large, loose clusters ($K = 3$, $\sigma = 0.09$) with diffuse edges, contrasting sharply with *Islands* (few-and-large vs. many-and-small).

Imbalanced. $K = 7$ centers with weights drawn from Dirichlet($\alpha = 0.25$), producing highly uneven populations—typically one dominant cluster contains 60–80% of all customers.

Overlap. $K = 9$ centers with large noise ($\sigma = 0.12$) contracted toward the unit-square center, forcing nine latent clusters to entangle into a near-uniform point cloud.

Anisotropic. $K = 6$ elongated strands with anisotropic shape ($\sigma_x = 0.025$, $\sigma_y = 0.14$), each randomly rotated, modeling directionally biased clusters along varying axes.



Figure 7: Visualization of the six Cluster distribution variants on CVRP100 instances. The orange star marks the depot. Top row: *Islands*, *Mega*, *Imbalanced*; bottom row: *Overlap*, *Anisotropic*, *Bridge*.

Table 12: Generalization performance on CVRP100 across different Cluster distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined.

Method	Cluster distribution variants						Average Gap	Time
	Islands	Mega	Imbalanced	Overlap	Anisotropic	Bridge		
LKH3	—	—	—	—	—	—	—	13.56s
Single-distribution training								
ELG	2.58%	2.89%	6.42%	3.38%	2.29%	4.59%	3.52%	4.35s
LEHD	3.59%	3.53%	4.55%	3.41%	3.68%	5.30%	3.96%	4.26s
BQ	3.67%	2.30%	3.08%	2.59%	3.20%	2.42%	2.90%	2.30m
INViT	11.79%	9.40%	10.24%	10.48%	11.99%	9.24%	10.60%	15.09m
Multi-distribution training								
AMDKD-AM	5.74%	6.08%	9.28%	7.77%	5.92%	7.57%	6.88%	1.17s
AMDKD-POMO	2.65%	2.56%	**1.73%**	2.43%	2.33%	2.40%	2.37%	1.01s
LEHD-VCT+MEM	3.01%	2.66%	3.40%	2.65%	2.89%	4.23%	3.11%	8.10s
POMO	2.03%	2.19%	7.55%	2.71%	1.99%	4.35%	3.25%	2.00s
MixRoute	**1.62%**	**1.49%**	2.44%	**1.90%**	**1.48%**	**2.31%**	**1.82%**	2.27s

Bridge. $K = 2$ well-separated dense clusters with 12% of customers placed along the line connecting them, modeling bi-modal urban-rural patterns where customers thin out along a connecting highway.

Results. Table 12 reports the per-variant performance on the Cluster family. MIXROUTE achieves the lowest gap on five of the six variants and an average gap of 1.82%, outperforming the second-best baseline (AMDKD-POMO, 2.37%). The exception is *Imbalanced*, where AMDKD-POMO (1.73%) edges out MIXROUTE (2.44%); we attribute this to AMDKD-POMO's exposure to skewed cluster patterns during multi-distribution training.

C.4 Grid Distribution

The Grid distribution family generates node coordinates from a regular 10×10 lattice over the unit square, perturbed by small jitter (± 0.02). Compared to the random distributions in earlier sections, Grid represents a structurally regular regime that differs sharply from the random training distribution. We construct six variants that share the same underlying lattice but apply distinct geometric or stochastic perturbations. Figure 8 visualizes one representative instance from each variant.

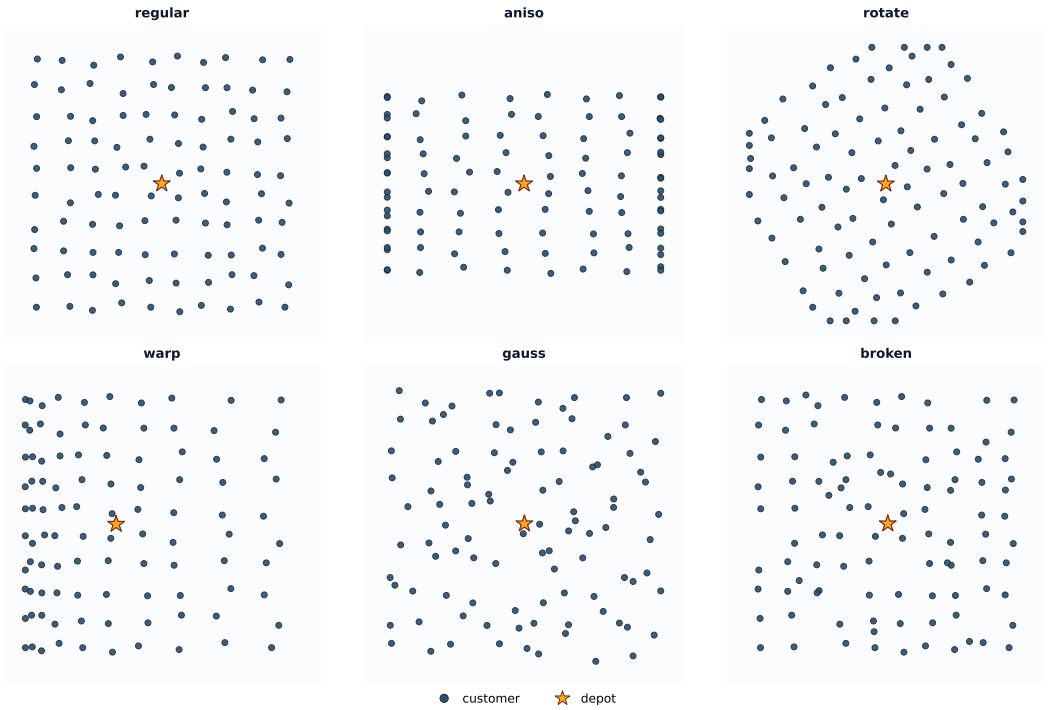

Figure 8: Visualization of the six Grid distribution variants on CVRP100 instances. The orange star marks the depot. *Regular* serves as the structural baseline; the other five variants apply distinct perturbations on top.

Regular. The base 10×10 lattice with only default jitter, yielding an almost ideal grid that an ideal NCO solver should exploit via row/column sweeps.

Aniso. The grid is rescaled around the center with $a_x = 1.5$ and $a_y = 0.7$, producing a horizontally stretched array with spacing ratio ~ 2.14, modeling coordinate-scale anisotropy.

Rotate. The grid is rotated by $\theta \sim \text{Uniform}(0, \pi/2)$ around its center, with edge points clamped back to $[0, 1]^2$. Although VRP solutions are rotation-invariant, models taking absolute coordinates as input may not learn this invariance.

Warp. The x-coordinate is compressed via $x \leftarrow x^2$, densifying the left side and sparsifying the right—reminiscent of a perspective projection.

Gauss. Gaussian noise ($\sigma = 0.03$) is added on top of the jitter, making the lattice structure nearly invisible while the underlying generative process remains anchored at lattice centers.

Broken. 15% of lattice points are replaced with uniform samples, simulating semi-structured logistics networks where the solver must exploit grid regularity while accommodating outliers.

Results. Table 13 reports the per-variant performance on the Grid family. MIXROUTE achieves the lowest gap on five of the six variants and an average gap of 1.18%, outperforming the second-best baseline (POMO, 1.38%). The exception is *Regular*, where POMO (1.43%) marginally edges out MIXROUTE (1.45%); we view this as evidence that MIXROUTE does not sacrifice in-structure performance to gain robustness, since the two methods remain effectively tied on the structural baseline while MIXROUTE dominates across all five perturbed variants.

C.5 Stripe Distribution

The Stripe distribution family generates node coordinates by sampling the x-coordinate from one of K vertical bands of half-width w centered at fixed positions across $[0.1, 0.9]$, with the y-coordinate drawn uniformly from $[0, 1]$. This family captures routing scenarios where customers are organized along structured longitudinal corridors—such as customers along major roads, parallel commercial

Table 13: Generalization performance on CVRP100 across different Grid distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined.

Method	Grid distribution variants						Average Gap	Time
	Regular	Aniso	Rotate	Warp	Gauss	Broken		
LKH3	—	—	—	—	—	—	—	9.64s
Single-distribution training								
ELG	2.01%	0.89%	1.74%	1.70%	1.77%	1.72%	1.65%	4.12s
LEHD	2.64%	1.95%	2.32%	2.86%	2.67%	3.08%	2.60%	4.26s
BQ	2.71%	2.26%	2.48%	2.75%	2.62%	3.20%	2.67%	2.30m
INViT	7.02%	7.50%	7.35%	7.81%	7.62%	8.31%	7.60%	15.16m
Multi-distribution training								
AMDKD-AM	3.92%	4.05%	4.24%	4.29%	4.02%	4.35%	4.14%	1.02s
AMDKD-POMO	1.88%	1.17%	1.69%	1.52%	1.59%	1.98%	1.64%	1.00s
LEHD-VCT+MEM	2.08%	1.85%	2.19%	2.21%	2.05%	2.36%	2.13%	8.09s
POMO	**1.43%**	1.30%	1.40%	1.28%	1.25%	1.64%	1.38%	1.94s
MixRoute	1.45%	**0.49%**	**1.36%**	**1.25%**	**1.05%**	**1.42%**	**1.18%**	2.25s



Figure 9: Visualization of the eight Stripe distribution variants on CVRP100 instances. The orange star marks the depot. The eight variants correspond to a $2 \times 2 \times 2$ design over band width, number of bands, and population balance.

avenues, or multi-lane delivery routes. We construct eight variants spanning a $2 \times 2 \times 2$ design: *band width* (narrow $w = 0.02$ vs. wide $w = 0.10$), *number of bands* ($K = 2$ vs. $K = 8$), and *population balance* (uniform vs. biased). Figure 9 visualizes one representative instance from each variant.

Rails. *(Narrow, $K = 2$, uniform.)* Two narrow vertical lines at $x \in \{0.1, 0.9\}$ each receiving roughly 50 customers, leaving the central region empty.

Solo. *(Narrow, $K = 2$, biased.)* The same two narrow lines as *Rails*, but with population weights $(0.25, 0.75)$, stressing load balancing across asymmetric corridors.

Comb. *(Narrow, $K = 8$, uniform.)* Eight narrow vertical lines evenly spaced across $[0.1, 0.9]$, each containing $\sim$ 12–13 customers, resembling the teeth of a comb.

Spike. *(Narrow, $K = 8$, biased.)* A comb-like layout where the central band absorbs 75% of the population, creating a sharp "spike" protruding from an otherwise sparse comb.

Bands. *(Wide, $K = 2$, uniform.)* The wide-band counterpart of *Rails*: two wide bands ($w = 0.10$) covering roughly 20% of the x-range each, with noticeably more diffuse edges.

Mono. *(Wide, $K = 2$, biased.)* The wide-band counterpart of *Solo*: two bands with population weights $(0.25, 0.75)$, testing dominant-corridor identification under blurred boundaries.

Table 14: Generalization performance on CVRP100 across different Stripe distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined.

Method	Stripe distribution variants								Average Gap	Time
	Rails	Solo	Comb	Spike	Bands	Mono	Flood	Core		
LKH3	—	—	—	—	—	—	—	—	—	16.39s
Single-distribution training										
ELG	1.58%	2.48%	1.53%	3.40%	1.19%	2.18%	1.80%	2.60%	2.02%	5.57s
LEHD	1.98%	3.37%	2.86%	4.83%	1.44%	3.21%	2.94%	3.59%	2.92%	5.73
BQ	3.61%	4.93%	2.66%	**2.03%**	1.92%	3.92%	3.03%	1.92%	3.02%	3.21m
INViT	11.64%	11.69%	9.31%	13.61%	8.11%	10.13%	8.93%	11.53%	10.45%	20.15m
Multi-distribution training										
AMDKD-AM	3.33%	3.80%	4.76%	9.33%	3.30%	4.16%	4.69%	7.62%	4.91%	1.42s
AMDKD-POMO	1.69%	1.95%	1.57%	3.73%	0.52%	**1.35%**	1.50%	2.65%	1.77%	1.44s
LEHD-VCT+MEM	1.30%	3.66%	2.53%	4.44%	0.97%	2.86%	2.59%	3.08%	2.56%	10.79s
POMO	7.11%	50.55%	1.20%	4.01%	0.63%	2.02%	1.36%	2.21%	8.30%	2.66s
MixRoute	**0.21%**	**1.57%**	**1.00%**	2.13%	**0.39%**	1.46%	**1.13%**	**1.68%**	**1.13%**	3.04s



Figure 10: Visualization of the four Gaussian distribution variants on CVRP100 instances. The orange star marks the depot. From left to right, σ increases from 0.1 to 0.4, traversing a spectrum from highly concentrated clusters to near-uniform with mild boundary clipping.

Flood. *(Wide, $K = 8$, uniform.)* Eight wide bands that heavily overlap due to w exceeding half the inter-center spacing, producing a "pseudo-uniform" marginal whose surface appearance masks an underlying eight-corridor structure.

Core. *(Wide, $K = 8$, biased.)* The wide-band counterpart of *Spike*: eight wide bands with the central band carrying 75% of the population, mirroring main-road plus side-street logistics patterns.

Results. Table 14 reports the per-variant performance on the Stripe family. MIXROUTE achieves the lowest gap on six of the eight variants and an average gap of 1.13%, outperforming the second-best baseline (AMDKD-POMO, 1.77%). The most striking case is *Solo*, where POMO collapses to 50.55%—the largest failure in our benchmark—while MIXROUTE attains 1.57% under identical training conditions, a 49-point improvement that highlights how severely a fixed normalization can overfit to a single density assumption.

C.6 Gaussian Distribution

The Gaussian distribution family generates node coordinates by sampling each axis independently from a Gaussian distribution centered at $(0.5, 0.5)$ with standard deviation σ, then clipping to the unit square. The Gaussian family varies along a single continuous axis—the standard deviation σ—which controls how concentrated the customer population is around the depot, capturing routing scenarios at varying spatial scales: tightly localized urban deliveries, mid-sized city logistics, metropolitan networks, and country-scale distribution. We construct four variants with increasing σ. Figure 10 visualizes one representative instance from each variant.

Tight ($\sigma = 0.1$). A compact blob occupying roughly 30–40% of the unit square with virtually no boundary clipping, mirroring urban deliveries within a single commercial district.

Compact ($\sigma = 0.2$). A moderate cluster spanning ± 0.2 around the center with mild boundary clipping, modeling mid-sized cities with a dense business district and sparser suburbs. This σ matches the most common Gaussian setting in prior NCO literature.

Table 15: Generalization performance on CVRP100 across different Gaussian distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined.

Method	Gaussian distribution variants				Average Gap	Time
	Tight ($\sigma = 0.1$)	Compact ($\sigma = 0.2$)	Loose ($\sigma = 0.3$)	Broad ($\sigma = 0.4$)		
LKH3	—	—	—	—	—	7.73s
Single-distribution training						
ELG	4.35%	2.29%	1.80%	1.61%	2.16%	2.74s
LEHD	4.39%	3.07%	3.20%	2.84%	3.19%	2.80s
BQ	3.55%	2.56%	2.87%	2.98%	2.91%	1.40m
INViT	9.90%	9.78%	9.64%	9.77%	9.75%	10.14m
Multi-distribution training						
AMDKD-AM	12.00%	5.91%	5.10%	4.65%	5.97%	0.73s
AMDKD-POMO	**2.25%**	2.32%	1.62%	1.42%	1.79%	0.72s
LEHD-VCT+MEM	3.37%	2.58%	2.67%	2.72%	2.75%	5.41s
POMO	3.86%	1.87%	1.38%	1.16%	1.72%	1.31s
MixRoute	2.29%	**1.66%**	**1.22%**	**0.93%**	**1.36%**	1.50s

Loose ($\sigma = 0.3$). A broad cluster with the 3σ region nearly filling the unit square; the central density peak remains identifiable while the periphery approaches uniform-like density—a metropolitan-scale pattern with both dense urban core and non-negligible suburban customers.

Broad ($\sigma = 0.4$). A near-uniform appearance with two latent signatures: a mildly elevated central peak, and "walls" of customers along the unit-square boundary due to substantial clipping. This serves as a "pseudo-uniform" diagnostic for whether models genuinely handle uniform structure or have overfit to a strict i.i.d. assumption.

Results. Table 15 reports the per-variant performance on the Gaussian family. MIXROUTE achieves the lowest gap on three of the four variants and an average gap of 1.36%, outperforming the second-best baseline (POMO, 1.72%). The exception is *Tight* ($\sigma = 0.1$), where AMDKD-POMO (2.25%) marginally edges out MIXROUTE (2.29%); we attribute this to AMDKD-POMO's exposure to highly concentrated cluster patterns during multi-distribution training.

C.7 Expansion Distribution

The Expansion distribution family generates node coordinates by first sampling uniformly from $[0, 1]^2$ and then applying a spatial deformation that distorts the geometric layout while preserving the unit-square support. Unlike density-shift families, Expansion varies the underlying *coordinate geometry* itself, producing instances whose marginal density may remain near-uniform but whose spatial structure deviates substantially. We construct six variants, each corresponding to a distinct mode of real-world urban geometry. Figure 11 visualizes one representative instance from each variant.

Axial. Horizontally stretched by a factor of 2 with edge clipping, yielding an elongated configuration with sparse interior, mirroring coastal cities like Chile or Norway compressed along a major road or river.

Log. The x-coordinate is nonlinearly stretched via a logarithmic-style mapping, producing a denser concentration around the center, corresponding to the classical *monocentric city* pattern with a dominant CBD.

Wedge. The angular coordinate is folded so that all customers lie on one side of the depot, capturing single-sided service regions bounded by a natural barrier such as a coastline, mountain range, or international border.

Pinch. A radial outward push produces a center-evacuated configuration with customers accumulating near the corners, modeling *donut-shaped urban patterns* where the depot is centrally placed but customers cluster in suburban belts.

Shear. An affine shear deforms the square into a parallelogram while preserving local density, capturing non-orthogonal road networks such as Manhattan's $29°$-tilted grid.

Squeeze. Horizontally stretched by $2.5\times$ and vertically compressed by $0.4\times$, yielding a 6:1 aspect-ratio strip, modeling linear corridor cities along major transit infrastructure such as the Tokaido Shinkansen or California's Highway 1.

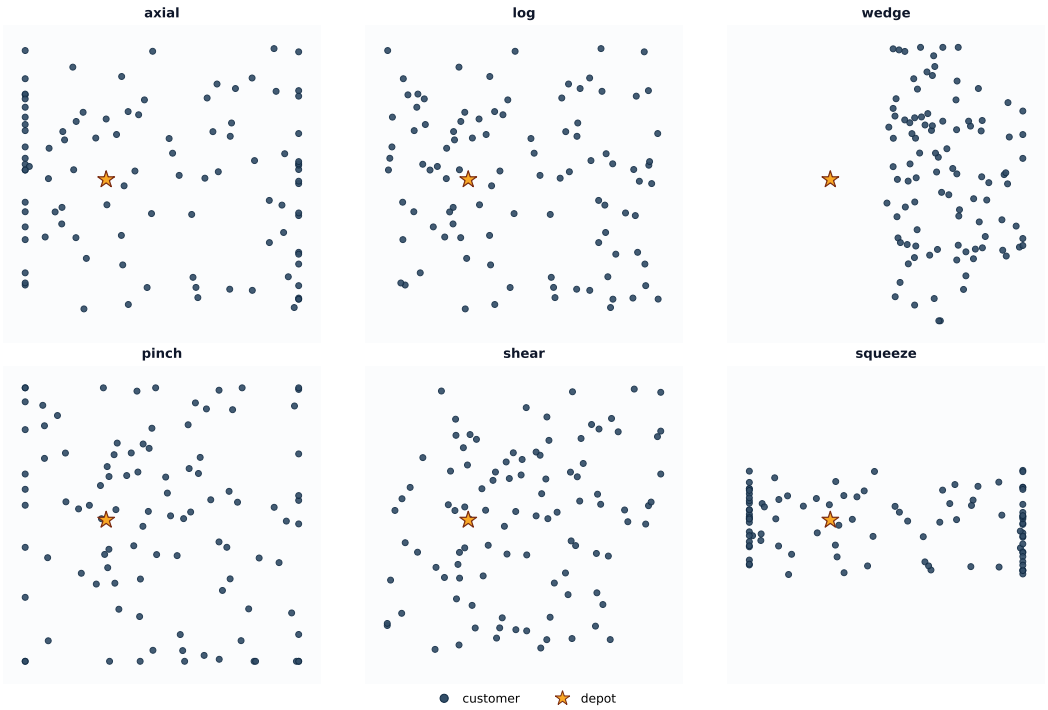

Figure 11: Visualization of the six Expansion distribution variants on CVRP100 instances. The orange star marks the depot. Each variant applies a distinct spatial deformation to a uniformly sampled point set.

Table 16: Generalization performance on CVRP100 across different Expansion distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined.

Method	Expansion distribution variants						Average Gap	Time
	Axial	Log	Wedge	Pinch	Shear	Squeeze		
LKH3	—	—	—	—	—	—	—	23.14s
Single-distribution training								
ELG	0.59%	0.16%	1.35%	-0.22%	-0.28%	4.10%	0.85%	4.17s
LEHD	0.47%	0.14%	0.29%	0.02%	-0.57%	2.67%	0.45%	4.25s
BQ	2.05%	1.02%	0.91%	0.64%	0.11%	3.74%	1.36%	2.30m
INViT	6.86%	6.46%	5.88%	6.67%	5.86%	8.67%	6.70%	15.12m
Multi-distribution training								
AMDKD-AM	3.74%	2.75%	3.55%	2.46%	2.45%	5.47%	3.34%	1.13s
AMDKD-POMO	0.37%	-0.08%	**-0.08%**	-0.31%	-0.52%	1.61%	0.13%	1.00s
LEHD-VCT+MEM	2.73%	1.43%	1.27%	1.65%	0.82%	3.61%	1.89%	8.12s
POMO	0.12%	-0.23%	2.83%	-0.54%	-0.56%	2.88%	0.65%	1.99s
MixRoute	**-0.10%**	**-0.40%**	-0.01%	**-0.75%**	**-0.86%**	**1.60%**	**-0.13%**	2.25s

Results. Table 16 reports the per-variant performance on the Expansion family. MIXROUTE achieves the lowest gap on five of the six variants and an average gap of -0.13%, surpassing LKH3 in aggregate and outperforming the second-best baseline (AMDKD-POMO, 0.13%). The exception is *Wedge*, where AMDKD-POMO (-0.08%) marginally edges out MIXROUTE (-0.01%); we attribute this to AMDKD-POMO's exposure to one-sided spatial patterns during multi-distribution training.

C.8 Implosion Distribution

The Implosion distribution family combines a uniform background with a high-density hotspot, with 30% of customers drawn from a structured concentrated region and the rest uniformly distributed. This models partial-concentration patterns—a dense commercial core coexisting with scattered surrounding customers—while varying the hotspot's geometry across six variants. Figure 12 visualizes one representative instance from each variant.



Figure 12: Visualization of the six Implosion distribution variants on CVRP100 instances. The orange star marks the depot.

Table 17: Generalization performance on CVRP100 across different Implosion distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined. Tied best results are both bolded.

Method	Implosion distribution variants						Average Gap	Time
	Circle	Star	Multi	Line	Ellipse	Gradient		
LKH3	—	—	—	—	—	—	—	22.19s
Single-distribution training								
ELG	0.46%	0.16%	0.31%	0.40%	0.41%	0.01%	0.29%	4.12s
LEHD	0.25%	-0.12%	0.20%	0.38%	0.24%	-0.03%	0.15%	4.25s
BQ	0.85%	0.40%	1.11%	0.84%	0.74%	0.63%	0.76%	2.30m
INViT	7.29%	7.29%	6.92%	7.03%	8.21%	6.24%	7.15%	15.12m
Multi-distribution training								
AMDKD-AM	3.48%	3.42%	3.21%	3.14%	3.50%	2.65%	3.23%	1.02s
AMDKD-POMO	**-0.22%**	**-0.52%**	-0.09%	-0.05%	**-0.34%**	-0.30%	-0.25%	1.05s
LEHD-VCT+MEM	1.36%	0.70%	1.18%	1.07%	0.64%	0.79%	0.96%	8.12s
POMO	0.08%	-0.16%	-0.16%	**-0.14%**	0.00%	-0.47%	-0.15%	2.02s
MixRoute	**-0.22%**	-0.41%	**-0.24%**	-0.13%	-0.16%	**-0.63%**	**-0.30%**	2.24s

Circle. A single Gaussian hotspot at a randomly chosen center, modeling a single-center city with a prominent commercial district surrounded by sparser residential customers.

Star. A non-convex five-pointed star hotspot with sharp protrusions, modeling commercial cores carved into jagged shapes by rivers, railways, or highways.

Multi. Three non-overlapping hotspots, modeling polycentric metropolitan regions such as Tokyo–Yokohama–Kawasaki or the Pearl River Delta.

Line. A horizontal narrow band hotspot, modeling linear-corridor patterns where customers concentrate along a subway line, riverside strip, or coastal boulevard.

Ellipse. An anisotropic Gaussian hotspot ($\sigma_x = 0.10$, $\sigma_y = 0.03$), modeling directional commercial cores elongated along a main avenue or waterfront.

Gradient. A smooth radial density with no discrete hotspot boundary, modeling continuous monocentric gradients where real-world population densities decay smoothly from city to suburb.



Figure 13: Visualization of the six Explosion distribution variants on CVRP100 instances. The orange star marks the depot.

Results. Table 17 reports the per-variant performance on the Implosion family. MIXROUTE achieves the lowest gap on three variants and ties on *Circle*, with an average gap of -0.30% that surpasses LKH3 in aggregate and outperforms the second-best baseline (AMDKD-POMO, -0.25%). The exceptions are *Star* and *Ellipse*, where AMDKD-POMO marginally edges out MIXROUTE; we attribute this to its exposure to anisotropic and irregular hotspots during multi-distribution training.

C.9 Explosion Distribution

The Explosion distribution family is the structural counterpart of Implosion: instead of injecting a hotspot, Explosion *removes* a structured low-density region (a "hole") from an otherwise uniform sample. This captures scenarios where certain areas are inaccessible or systematically depopulated—parks, lakes, military bases, airports, or protected zones. We construct variants whose hole geometry varies across distinct types. Figure 13 visualizes one representative instance from each variant.

Circle. A single circular hole at a randomly chosen center, modeling a single inaccessible zone such as a park, lake, military base, or airport.

Star. A non-convex five-pointed star hole, modeling forbidden regions with jagged outlines carved by rivers, railways, or airport perimeters.

Multi. Three non-overlapping circular holes, modeling distributed forbidden zones such as several parks or scattered restricted areas, requiring inter-hole navigation.

Ring. An annular exclusion zone separating an inner core from an outer periphery, modeling *moat-and-greenbelt* patterns such as Beijing's second-ring belt or airport runway exclusions, where the optimal path must loop around the entire ring.

Ellipse. An anisotropic elliptical hole with $2.5{:}1$ aspect ratio, modeling directionally elongated forbidden regions such as runways, canals, or highway median strips.

Gradient. A soft hole with no hard boundary, where the rejection probability decays smoothly from the center, modeling soft low-attraction zones such as high-risk districts or fringe areas.

Table 18: Generalization performance on CVRP100 across different Explosion distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined.

Method	Explosion distribution variants						Average Gap	Time
	Circle	Star	Multi	Ring	Ellipse	Gradient		
LKH3	—	—	—	—	—	—	—	23.41s
Single-distribution training								
ELG	0.26%	0.19%	0.07%	0.12%	0.22%	0.25%	0.18%	4.11s
LEHD	0.51%	0.26%	0.08%	0.35%	0.35%	0.41%	0.33%	4.26s
BQ	0.65%	0.69%	0.78%	1.01%	0.86%	0.81%	0.80%	2.30m
INViT	6.41%	6.47%	6.25%	6.21%	6.49%	6.45%	6.38%	15.14m
Multi-distribution training								
AMDKD-AM	3.01%	2.83%	2.68%	2.89%	3.04%	2.80%	2.88%	1.08s
AMDKD-POMO	-0.01%	-0.05%	-0.17%	0.02%	0.17%	-0.02%	-0.01%	1.05s
LEHD-VCT+MEM	2.21%	1.84%	1.23%	2.36%	1.45%	1.76%	1.81%	8.14s
POMO	-0.13%	**-0.30%**	-0.32%	-0.14%	-0.10%	-0.18%	-0.19%	1.99s
MixRoute	**-0.27%**	-0.26%	**-0.47%**	**-0.23%**	**-0.21%**	**-0.23%**	**-0.28%**	2.25s

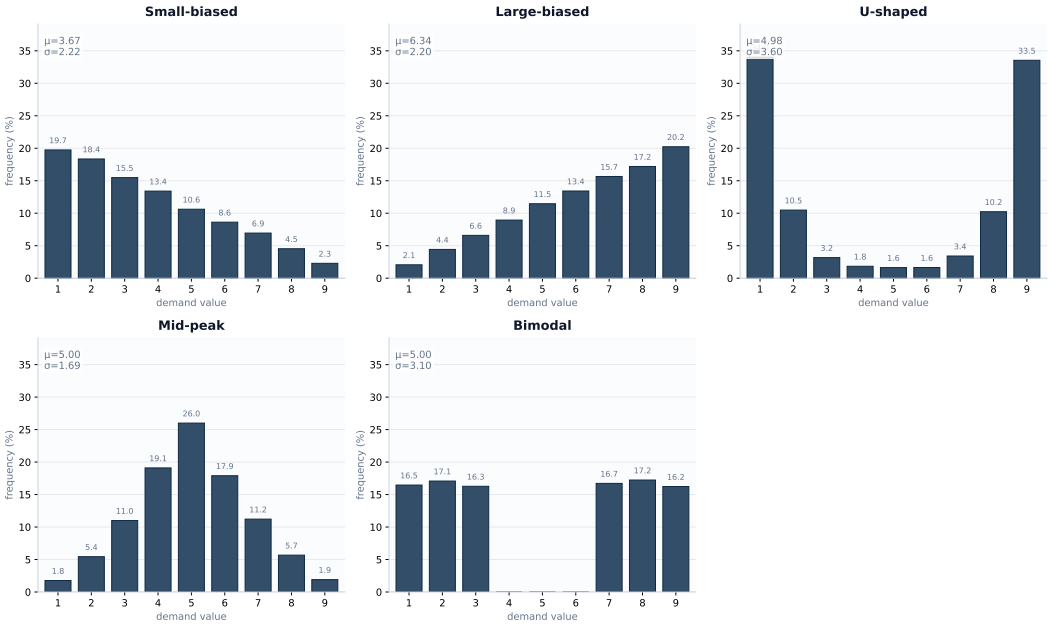

Figure 14: The five Demand distribution variants. Each panel shows the probability mass function over demand values $\{1, 2, \ldots, 9\}$. From left to right: *Small-biased*, *Large-biased*, *U-shaped*, *Mid-peak*, and *Bimodal*.

Results. Table 18 reports the per-variant performance on the Explosion family. MIXROUTE achieves the lowest gap on five of the six variants and an average gap of -0.28%, surpassing LKH3 in aggregate and outperforming the second-best baseline (POMO, -0.19%). The exception is *Star*, where POMO (-0.30%) marginally edges out MIXROUTE (-0.26%); given the small magnitude, we view this as effectively tied performance.

C.10 Demand Distribution

The Demand distribution family departs from preceding families by varying *customer demand values* rather than spatial coordinates. Standard CVRP samples demands uniformly from $\{1, 2, \ldots, 9\}$, but real-world ordering patterns exhibit diverse structures: retail last-mile is dominated by small orders, B2B by large orders, and mixed-channel warehouses by bimodal patterns. We construct five demand variants, each motivated by a distinct real-world ordering pattern. Figure 14 visualizes the demand probability mass function for each variant.

Small-biased. Probability mass decreases monotonically from 1 to 9 ($\mu \approx 3.67$), modeling retail last-mile patterns dominated by small orders such as food delivery or e-commerce final-mile.

Table 19: Generalization performance on CVRP100 across different Demand distributions. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined.

Method	Demand distribution variants					Average Gap	Time
	Small-biased	Large-biased	U-shaped	Mid-peak	Bimodal		
LKH3	—	—	—	—	—	—	10.48s
Single-distribution training							
ELG	2.21%	1.82%	2.23%	1.60%	1.89%	1.94%	4.48s
LEHD	2.67%	2.96%	3.08%	2.61%	3.05%	2.88%	4.22s
BQ	2.25%	3.45%	3.29%	2.28%	3.16%	2.92%	2.30m
INViT	9.60%	10.60%	10.00%	8.76%	9.35%	9.69%	15.21m
Multi-distribution training							
AMDKD-AM	5.08%	4.34%	4.96%	4.63%	5.00%	4.78%	2.35s
AMDKD-POMO	2.20%	1.83%	2.00%	1.48%	1.90%	1.87%	1.01s
LEHD-VCT+MEM	1.92%	2.84%	2.98%	2.44%	2.98%	2.66%	8.88s
POMO	1.77%	1.36%	1.69%	1.17%	1.53%	1.49%	1.97s
MixRoute	**1.25%**	**1.27%**	**1.43%**	**0.96%**	**1.29%**	**1.24%**	2.25s

Large-biased. The mirror of *Small-biased*, with mass increasing monotonically from 1 to 9 ($\mu \approx 6.34$), modeling B2B and bulk distribution such as wholesale supply chain or trunk-line logistics.

U-shaped. Mass concentrates at the two extremes ($\sim 34\%$ each at 1 and 9, with intermediate values sparse but non-zero), modeling mixed-channel warehouses serving B2C retail and B2B wholesale customers from the same depot.

Mid-peak. A symmetric bell-shape centered at demand 5 with the smallest variance ($\sigma \approx 1.69$), modeling standardized procurement such as case-packed FMCG or routine institutional orders.

Bimodal. A discontinuous distribution with mass at $\{1, 2, 3\}$ and $\{7, 8, 9\}$ but exactly zero at $\{4, 5, 6\}$, modeling dichotomous order regimes such as tiered packaging or retail-vs-wholesale partitions. The absence of intermediate values is a structural property that no smooth training distribution exposes the model to.

Results. Table 19 reports the per-variant performance on the Demand family. MIXROUTE achieves the lowest gap on every variant, with an average gap of 1.24% that outperforms the second-best baseline (POMO, 1.49%). Notably, the second-best method on this family is POMO rather than any multi-distribution baseline, suggesting that exposure to additional spatial distributions transfers poorly to demand-distribution shifts.

C.11 Capacity Configuration

The Capacity dimension perturbs the vehicle capacity Q while keeping node coordinates and customer demands fixed at the standard uniform setting. Since demands enter the model only through the normalized ratio d_i/Q, scaling Q is equivalent to rescaling all demands proportionally, providing a controlled probe of how solvers handle changes in the *numerical regime* of the input.

We evaluate five capacity settings, $Q \in \{20, 80, 120, 200, 250\}$, inducing a more than tenfold variation in the per-customer load $\bar{d}/Q$ (from 0.250 at $Q = 20$ to 0.020 at $Q = 250$). All five settings lie outside the standard CVRP training capacity ($Q = 50$) and thus constitute out-of-distribution evaluation. Table 20 reports the per-setting performance.

C.12 Time Window Shifts

Among the three constraint dimensions in CVRPTW, time windows are arguably the most multi-faceted: a time-window assignment involves not only *how wide* the window is, but also *when* during the operating day it falls, and *how* it correlates with the customer's spatial position relative to the depot. Standard CVRPTW benchmarks conflate these three aspects, making it difficult to attribute failures to any specific axis. We decompose time-window shifts into three orthogonal dimensions—*Width*, *Timing*, and *Coupling*—each isolating one aspect while holding the others at training-distribution values.

Width: varying the time-window length. This dimension perturbs `tw_length` while leaving the starting-time distribution unchanged:

Table 20: Generalization performance on CVRP100 across different Capacity settings. Gap (%) is computed against LKH3. Best results among neural methods are in **bold**; second best are underlined.

Method	Capacity setting $Q = 20$	$Q = 80$	$Q = 120$	$Q = 200$	$Q = 250$	Average Gap	Time
LKH3	—	—	—	—	—	—	24.30s
Single-distribution training							
ELG	2.79%	2.66%	5.57%	8.42%	11.67%	4.89%	3.44s
LEHD	4.37%	3.23%	3.25%	2.80%	5.60%	3.97%	3.55s
BQ	7.66%	1.61%	2.57%	2.97%	6.24%	5.18%	2.51m
INViT	12.10%	8.02%	10.97%	13.60%	14.60%	11.73%	12.30m
Multi-distribution training							
AMDKD-AM	4.22%	5.50%	8.37%	12.52%	13.41%	7.12%	0.88s
AMDKD-POMO	4.12%	2.45%	5.62%	8.99%	11.69%	5.53%	0.88s
LEHD-VCT+MEM	**1.97%**	2.75%	**1.58%**	**0.11%**	**0.82%**	**1.68%**	6.79s
POMO	3.88%	1.98%	5.86%	18.04%	27.46%	8.34%	1.76s
MixRoute	2.47%	**1.31%**	2.66%	5.13%	8.75%	3.36%	1.86s

- *tw_loose*: `tw_length` $\sim$ Uniform$([0.5, 1.0])$, with windows wide enough to render time-window constraints nearly vacuous.
- *tw_tight*: `tw_length` $\sim$ Uniform$([0.05, 0.10])$, simulating precise appointments that allow virtually no scheduling flexibility.
- *tw_bimodal*: a $50/50$ mixture of tight and loose, modeling scenarios where some customers require precise scheduling (e.g., VIP deliveries) while others tolerate flexible service.

Timing: varying when service occurs during the day. This dimension fixes the window length but perturbs `tw_start`, changing *when* during the operating horizon the customer load is concentrated:

- *tw_late_rush*: `tw_start` $\sim$ Uniform$([0.7 \cdot \text{MAX}, \text{MAX}])$, simulating an evening rush with morning idleness.
- *tw_narrow_band*: `tw_start` $\sim \mathcal{N}(\text{MAX}/2, 0.2)$, forcing parallel dispatch within a single mid-day window.
- *tw_stripes_2*: two narrow bands at $0.25 \cdot \text{MAX}$ and $0.75 \cdot \text{MAX}$ with $\sigma = 0.05$, modeling morning and afternoon dispatch waves.

The standard training generator induces a mild positive correlation between `tw_start` and the depot-to-customer distance, which all three variants break in different ways.

Coupling: varying the spatial–temporal correlation. This dimension fixes both window length and starting-time distribution but severs the implicit dependency between `tw_start` and the depot-to-customer distance:

- *tw_decoupled*: `tw_start` $\sim$ Uniform$([\text{lo}, \text{hi}])$, sampled independently of d_{0i}.

The standard training distribution generates `tw_start` via a formula that scales with d_{0i}, inducing a strong "far implies late" prior that models may overfit to as a shortcut. *tw_decoupled* severs this prior, testing whether the model has learned genuine spatial–temporal reasoning rather than a distance-based heuristic.

D Multi-seed results

We retrained MixRoute four times with different seeds and report the mean $\pm$ standard deviation across runs in Table 21, where run 1 corresponds to the model reported in the main paper. We report a representative subset of six distributions, including Implosion, Explosion, and Demand, on which the margin of MixRoute over POMO is among the smallest in our benchmark.

The variation across runs is small: the standard deviation is at most 0.15% on five of the six distributions, and 0.33% on Capacity, where the margin over POMO (4.69%) remains more than an order of magnitude larger. More importantly, the variation does not affect the conclusion, as each of the four runs outperforms POMO on all six distributions.

Table 21: Results of MixRoute across four independent training runs on CVRP100. Gap (%) is computed against LKH3. The mean ± standard deviation is computed over the four runs.

Method (CVRP)	Annulus	Expansion	Implosion	Explosion	Demand	Capacity
POMO	7.11%	0.65%	-0.15%	-0.19%	1.49%	8.34%
MixRoute (run 1)	-0.87%	-0.13%	-0.30%	-0.28%	1.24%	3.36%
MixRoute (run 2)	-0.59%	-0.25%	-0.50%	-0.48%	1.36%	4.05%
MixRoute (run 3)	-0.56%	-0.31%	-0.48%	-0.51%	1.30%	3.79%
MixRoute (run 4)	-0.57%	-0.30%	-0.51%	-0.55%	1.33%	3.40%
MixRoute (mean ± std)	**-0.65±0.15%**	**-0.25±0.08%**	**-0.45±0.10%**	**-0.46±0.12%**	**1.31±0.05%**	**3.65±0.33%**

E Mechanism of MixNorm

To understand how MixNorm responds to distributional shift, we measure how far the encoder representation moves when the input distribution changes. We define the *mean shift* at each encoder layer as the L_2 distance between the mean node embedding under a given OOD distribution and that under the in-distribution setting, and group OOD distributions by the improvement of MixRoute over POMO.

Table 22: Mean embedding shift across encoder layers, grouped by the improvement of MixRoute over POMO. Δ is the difference in the layer-averaged shift between the two models; the last column reports the average gap reduction within each group.

Group	**Model**	L0	L1	L2	L3	L4	L5	**Δ**	**Improv.**
>5%	POMO	0.08	0.11	0.11	0.10	0.09	0.06		
	MixRoute	4.72	4.94	3.48	4.57	3.44	2.31	3.82	+22.6%
1–5%	POMO	0.04	0.05	0.04	0.03	0.02	0.01		
	MixRoute	1.35	1.45	1.01	1.15	0.91	0.67	1.06	+3.7%
<1%	POMO	0.03	0.03	0.03	0.02	0.02	0.02		
	MixRoute	0.21	0.97	0.94	0.89	0.72	0.67	0.70	+0.2%

As shown in Table 22, the encoder representation of POMO barely moves under distributional shift, whereas that of MixRoute responds substantially, and the magnitude of the response grows with the improvement over POMO. Extending the analysis to 16 additional distributions, the embedding shift of MixRoute is positively associated with its improvement over POMO (Pearson $r = 0.651$, Spearman $\rho = 0.685$, $n = 16$). We emphasize that this association is descriptive rather than causal: a larger mean shift does not by itself lead to better performance. We next turn to the gate itself, whose behavior is analyzed in Appendix F.

F Behavior of the Adaptive Gate

We examine the learned gate weights from four perspectives: across encoder layers, over the course of training, across distribution families, and in response to specific instance-level characteristics. Recall that MixNorm is applied after both the multi-head self-attention (MHA) and the feed-forward (FFN) sublayers of each encoder layer.

Layer-wise patterns. Table 23 reports the mean gate weights at each layer. The two insertion points exhibit distinct patterns. After MHA, BN receives the largest weight at every depth, while the contribution of IN is concentrated in the shallowest layer. After FFN, BN dominates only at L0, and LN becomes the largest component in all subsequent layers.

Training dynamics. Table 24 tracks the gate weights of the post-MHA MixNorm at layer L5 over four checkpoints. The weights evolve steadily throughout training, with BN gradually absorbing weight from IN.

Distribution-level adaptation. To test whether the gate adapts to distributional structure, we compare the gate weights under Uniform, Annulus (thin ring), and Stripe (spike) instances using

Table 23: Mean gate weights of each normalizer across encoder layers.

	Post-MHA			Post-FFN		
Layer	IN	LN	BN	IN	LN	BN
L0	0.43	0.01	0.57	0.03	0.26	0.71
L1	0.19	0.40	0.41	0.30	0.47	0.23
L2	0.28	0.15	0.58	0.00	0.99	0.01
L3	0.20	0.13	0.67	0.00	0.70	0.30
L4	0.03	0.05	0.91	0.05	0.64	0.32
L5	0.05	0.19	0.77	0.38	0.53	0.09

Table 24: Gate weights of the post-MHA MixNorm at layer L5 over training.

Epoch	IN	LN	BN
500	0.21	0.23	0.56
2000	0.14	0.21	0.65
4000	0.11	0.18	0.71
6500	0.05	0.19	0.77

Welch's t-test for all three pairwise comparisons, with Bonferroni correction ($\alpha = 0.05/3 \approx 0.0167$). As shown in Table 25, the gate weights differ significantly between distribution families in the vast majority of comparisons, indicating that the gate adapts its weighting over normalizers to the input distribution.

Table 25: p-values of Welch's t-test on gate weights between distribution families (U: Uniform, A: Annulus, S: Stripe). Values in parentheses are not significant after Bonferroni correction ($\alpha = 0.0167$).

	IN			LN			BN		
Layer	U vs A	U vs S	A vs S	U vs A	U vs S	A vs S	U vs A	U vs S	A vs S
L0-attn	(6e-02)	4e-37	4e-70	3e-33	4e-25	2e-89	(5e-02)	1e-31	3e-44
L0-ffn	7e-07	6e-29	1e-39	2e-50	8e-90	5e-195	4e-79	1e-105	2e-181
L1-attn	(7e-02)	1e-07	2e-155	5e-19	8e-10	2e-149	3e-06	9e-04	2e-13
L1-ffn	2e-55	1e-92	6e-189	3e-05	6e-04	(2e-02)	4e-61	4e-65	7e-162
L2-attn	2e-04	4e-58	2e-109	3e-70	3e-58	7e-119	1e-40	3e-13	2e-58
L2-ffn	1e-69	3e-55	3e-16	3e-20	5e-29	3e-41	1e-20	8e-29	3e-41
L3-attn	2e-28	1e-45	1e-102	2e-114	2e-55	6e-137	2e-73	5e-58	1e-136
L3-ffn	3e-39	1e-41	(9e-02)	4e-121	9e-28	1e-196	8e-132	1e-22	5e-197
L4-attn	3e-43	3e-19	6e-84	8e-33	4e-42	9e-04	9e-68	(8e-02)	2e-81
L4-ffn	1e-115	8e-87	6e-12	2e-53	1e-64	3e-06	2e-07	9e-25	2e-18
L5-attn	3e-55	2e-59	(3e-02)	3e-29	7e-41	4e-98	2e-47	5e-13	3e-98
L5-ffn	8e-28	8e-29	(2e-01)	8e-17	7e-42	4e-98	4e-142	2e-06	9e-200

Instance-level adaptation. Beyond distinguishing distribution families, we examine how the gate responds to two instance-level characteristics that vary continuously within a family: the spatial dispersion of nodes (Gaussian family) and the tightness of the capacity constraint (Capacity family). Tables 26 and 27 report the mean and standard deviation of the gate weights at layer L5, expressed as ratios relative to the corresponding in-distribution baseline. Both reveal systematic trends.

Spatial dispersion. Under the Gaussian shift (Table 26), the standard deviation of the gate weights is uniformly lower than under Uniform, indicating a more consistent weighting across tokens. Moreover, as the nodes become progressively more dispersed, the mean weight of IN declines steadily while that of BN rises, a trend that holds at both insertion points.

Capacity tightness. The effect of capacity diverges between the two insertion points (Table 27). After MHA, the mean weight of IN rises sharply as capacity loosens, nearly doubling at $Q = 250$, while that of BN declines. After FFN, the trend reverses: BN gains substantial weight as capacity loosens, reaching $1.43\times$ its baseline at $Q = 250$. Together, these results suggest that the gate responds not only to which distribution family an instance belongs to, but also to continuous variations in its structure.

Table 26: Gate weight statistics at layer L5 under the Gaussian shift, as ratios relative to the Uniform baseline. From *tight* to *broad*, nodes become progressively more dispersed.

	Post-MHA				Post-FFN			
Statistic	Tight	Compact	Loose	Broad	Tight	Compact	Loose	Broad
IN mean	1.11	1.10	1.08	1.04	1.10	1.09	1.06	1.03
LN mean	1.05	0.95	0.91	0.87	0.92	0.93	0.95	0.96
BN mean	0.98	1.00	1.01	1.02	0.94	0.97	1.01	1.07
IN std	0.76	0.78	0.74	0.73	0.66	0.61	0.64	0.62
LN std	0.48	0.42	0.42	0.40	0.75	0.67	0.67	0.63
BN std	0.54	0.50	0.52	0.50	0.67	0.70	0.79	0.82

Table 27: Gate weight statistics at layer L5 under the Capacity shift, as ratios relative to the $Q = 50$ baseline. Larger Q corresponds to a looser capacity constraint.

	Post-MHA					Post-FFN				
Statistic	20	80	120	200	250	20	80	120	200	250
IN mean	0.53	1.39	1.71	1.90	1.93	0.84	0.99	0.95	0.90	0.88
LN mean	1.17	1.01	1.03	1.04	1.05	1.13	0.99	1.01	1.02	1.03
BN mean	1.01	0.95	0.91	0.89	0.88	1.07	1.08	1.21	1.37	1.43
IN std	0.49	1.36	1.47	1.60	1.65	1.31	0.89	0.97	1.17	1.23
LN std	0.81	1.07	1.13	1.13	1.11	1.48	0.93	0.95	1.08	1.14
BN std	0.83	1.12	1.22	1.31	1.33	1.01	1.06	1.19	1.37	1.42

G Comparison with Existing Mixing Normalizations

To position MixRoute relative to existing learning-to-normalize approaches, we compare against Switchable Normalization [28]—the most widely adopted mixing normalization in the literature—together with a stronger variant of SN that we design as an additional baseline. We adopt `IN+LN+BN` as the constituent set, following the standard configuration in the original SN paper.

Switchable Normalization (SN). SN performs *statistic mixing*: it computes the mean and variance of each constituent normalizer, takes a learnable weighted average of these statistics, and applies a single shared affine transformation (γ, β) to the result:

$$\mathrm{SN}(x) = \gamma \cdot \frac{x - \sum_i w_i \mu_i}{\sqrt{\sum_i w_i' \sigma_i^2 + \epsilon}} + \beta, \tag{12}$$

where w_i and w_i' are softmax-normalized scalar weights learned end-to-end, and (γ, β) are shared across all constituent normalizers.

SN-affine (our designed baseline). To stress-test whether SN's statistic-mixing paradigm can be pushed further, we design a stronger variant that we refer to as **SN-affine**, in which each constituent normalizer is additionally assigned its own affine parameters (γ_i, β_i). The effective affine applied to the mixed-statistic output is itself a weighted combination using the same mixing weights:

$$\text{SN-affine}(x) = \Big(\sum_i w_i \gamma_i\Big) \cdot \frac{x - \sum_i w_i \mu_i}{\sqrt{\sum_i w_i' \sigma_i^2 + \epsilon}} + \sum_i w_i \beta_i. \tag{13}$$

Table 28: Comparison with existing mixing normalizations on the CVRP benchmark. All variants use `IN+LN+BN` as the constituent set. Best results among neural methods are in **bold**.

	Node coordinate distributions									CVRP-specific		
Method	Annulus	Beta	Cluster	Expansion	Explosion	Gaussian	Grid	Implosion	Stripe	Demand	Capacity	**Avg**
LKH3	—	—	—	—	—	—	—	—	—	—	—	—
SN	-0.68%	1.17%	1.90%	-0.11%	-0.28%	1.45%	1.18%	**-0.30%**	1.57%	1.22%	8.98%	1.46%
SN-affine	-0.86%	1.12%	1.83%	-0.07%	-0.28%	1.40%	**1.12%**	-0.26%	**1.06%**	1.31%	8.01%	1.31%
MixRoute (ours)	**-0.87%**	**1.11%**	**1.82%**	**-0.13%**	**-0.28%**	**1.36%**	1.18%	**-0.30%**	1.13%	**1.24%**	**3.36%**	**0.88%**

Table 28 shows that MixRoute achieves the lowest average gap (0.88%), outperforming both SN (1.46%) and SN-affine (1.31%). Notably, although SN-affine pushes SN's design to its limit by additionally switching the affine parameters, it still falls short of MixRoute by a clear margin.

H Normalizers Choice

The constituent set of MixNorm follows Switchable Normalization [28], which combines statistics estimated instance-wise, layer-wise, and minibatch-wise through IN, LN, and BN. This appendix examines how the choice of constituent set affects performance, training dynamics, and inference cost. All experiments follow the CVRP setting of Section 5.1, and we report a representative subset of five distributions.

We first vary the number of constituent normalizers while keeping IN+LN as the base pair. As shown in Table 29, expanding the pool from two to three normalizers yields a clear gain, reducing the average gap from 1.93% to 0.77%. Adding a fourth component (No Norm) brings no further benefit.

Table 29: Effect of pool size on the CVRP benchmark. Gap (%) is computed against LKH3 and averaged over the five distributions reported here. Best results are in **bold**.

Method	Uniform	Annulus	Stripe	Demand	Capacity	**Avg**
POMO	-0.93%	7.11%	8.30%	1.49%	8.34%	4.86%
MixRoute-IN+LN	-0.23%	0.36%	2.77%	1.58%	5.16%	1.93%
MixRoute-IN+LN+BN	-1.03%	**-0.87%**	**1.13%**	**1.24%**	3.36%	**0.77%**
MixRoute-IN+LN+BN+No Norm	**-1.04%**	-0.53%	1.29%	**1.18%**	**3.18%**	0.82%

To test whether the benefit depends on a particular combination, we further evaluate pools that exclude the IN+LN base pair. As shown in Table 30, these variants still substantially outperform POMO, reducing the average gap from 4.86% to 1.12% and 1.07% respectively. This indicates that the gain stems from adaptively mixing complementary normalizers rather than from any specific combination, although IN+LN+BN remains the best overall.

Table 30: Effect of pool composition on the CVRP benchmark. Gap (%) is computed against LKH3. Best results are in **bold**.

Method	Uniform	Annulus	Stripe	Demand	Capacity	**Avg**
POMO	-0.93%	7.11%	8.30%	1.49%	8.34%	4.86%
MixRoute-IN+LN+BN	**-1.03%**	-0.87%	1.13%	**1.24%**	**3.36%**	**0.77%**
MixRoute-IN+BN+No Norm	-0.98%	**-0.88%**	**1.03%**	1.19%	5.24%	1.12%
MixRoute-LN+BN+No Norm	-1.01%	-0.68%	1.81%	1.52%	3.71%	1.07%

I Limitations

While MIXROUTE achieves strong cross-distribution generalization across diverse routing problems and backbones, a limitation is worth noting. Our MIXNORM module is integrated within the encoder layers' Add & Norm sub-blocks, which makes its applicability tied to architectures with sufficiently deep encoder stacks. Modern lightweight or decoder-only neural routing solvers [7, 27], which place most of the modeling capacity in the decoder and rely on shallow or even single-layer encoders, offer fewer normalization sites for MIXNORM to take effect, potentially limiting the gain that mixing-based normalization can deliver. Extending MIXNORM to the decoder side, or designing decoder-friendly mixing schemes, is a natural direction for future work.

J Broader Impacts

MIXROUTE aims to improve the cross-distribution generalization of neural routing solvers, which carries both positive and negative societal implications. On the positive side, more robust routing solvers

can be deployed across diverse real-world conditions without retraining, directly translating to shorter vehicle routes, reduced fuel consumption, and lower carbon emissions in logistics operations. Our single-distribution training paradigm also requires substantially less compute than multi-distribution alternatives, modestly reducing the environmental footprint of training itself. On the potential negative side, increased automation in routing and dispatch may reduce demand for manual planning labor in the logistics sector, and routing optimization is inherently dual-use, with possible applications beyond civilian logistics. As our work is methodological in nature and evaluated on synthetic benchmarks, we do not foresee a direct path to malicious use, but we encourage practitioners to weigh these implications when integrating such methods into deployed systems.

K Asset and Dataset

Table 31 lists all third-party codebases and datasets used in our work, along with their licenses and usage.

Table 31: Used assets, licenses, and their usage.

Type	Asset	License	Usage
Code	LKH3 [14]	BSD 3-Clause License	Evaluation (reference solver)
	HGS [41]	MIT License	Evaluation (reference solver, CVRPTW)
	POMO [22]	MIT License	Backbone & Baseline
	MatNet [23]	MIT License	Backbone (ACVRP)
	ELG [11]	MIT License	Baseline
	LEHD [27]	MIT License	Baseline
	BQ-NCO [7]	Apache-2.0 License	Baseline
	INViT [9]	MIT License	Baseline
	AMDKD [4]	MIT License	Baseline
	LEHD-VCT+MEM [26]	MIT License	Baseline
Dataset	CVRPLIB [39]	Publicly available for research use	Real-world evaluation (Sec. 5.4)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's contributions, which are supported by the experiments and analyses presented in the main text.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed limitations in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides full details on the architecture, training hyperparameters, and optimizer configuration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code, benchmark generators, and pre-trained checkpoints are released at github.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes. We include details in experiment section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix D reports mean ± std over four training seeds on six representative distributions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report this in experiment section and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics in all respects. The work is purely methodological, uses only synthetic routing instances.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discussed this in Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper releases neural routing solvers and synthetic VRP benchmark instances, neither of which poses a high risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We include this in Appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets introduced in this paper are documented in the main text and Appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We include this in Appendix.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.