SolvMix: Learning Formulation-State Landscapes for Liquid Electrolyte Conductivity Prediction

Kexin Zhang[1,2,3] **Minzhang Li**[1,4] **Guotao Qiu**[5] **Tianqi Zhao**[3] **Yuna Zhang**[5]
Yibing Shan[6,7] **Ye Mei**[3] **Xushan Zhao**[5] **Jingyi Yu**[1,†]

[1]School of Information Science and Technology, ShanghaiTech University, Shanghai, China
[2]Lingang Laboratory, Shanghai, China
[3]Shanghai Smart Logic Technology, Shanghai, China
[4]Cellverse Co., Ltd., Shanghai, China
[5]Contemporary Amperex Technology Co., Limited (CATL), Ningde, Fujian, China
[6]iHuman Institute, ShanghaiTech University, Shanghai, China
[7]School of Life Science and Technology, ShanghaiTech University, Shanghai, China
[†]Corresponding author: yujingyi@shanghaitech.edu.cn

Abstract

A liquid electrolyte is a recipe: a few familiar molecules, mixed in chosen proportions and used at a chosen temperature. Conductivity models already feed the proportions into the network, by concatenating them to atom features or by weighting each molecule's embedding by its molar ratio. We ask which form this conditioning should take. Once a component's token carries an amount embedding, adding a further amount-dependent shift is equivalent to no calibration: the amount moves every molecule of a type in the same way. A learned per-feature scale instead lets the effect of the amount depend on the molecule, and contains molar-ratio weighting as the special case of a uniform scale. SolvMix combines this scale with a direct temperature path to the readout. On two experimental benchmarks it lowers MSE by 30–39% relative to the strongest retrained baseline, and by 49–62% when whole regions of composition or temperature are held out; it also fits a synthetic benchmark almost exactly. Replacing the scale with a shift more than doubles the error, while changing the molecular encoder barely moves it.

1 Introduction

The electrolyte carries lithium ions between the electrodes of a battery, and its ionic conductivity limits how fast the battery can charge, how much power it can deliver, and how well it works in the cold [Xu, 2004, Logan and Dahn, 2020]. Raising conductivity without sacrificing stability is therefore a central target of electrolyte development, and each candidate formulation must be mixed and measured across temperatures, which makes the search slow and costly and motivates data-driven prediction [Ling, 2022]. Electrolytes are designed by choosing solvents and a salt and then tuning their proportions, so most candidate formulations reuse a small set of familiar molecules. What changes from one recipe to the next is how much of each molecule is present. Diluting ethylene carbonate (EC) with dimethyl carbonate (DMC) lowers viscosity but also lowers the dielectric constant, and which effect dominates depends on the ratio, the salt, and the temperature. A molecule's contribution to conductivity is therefore not a fixed property of the molecule.

Machine learning for mixtures has largely followed molecular machine learning: better graph encoders, pretraining, set aggregation, and geometric interaction between components [Gilmer et al., 2017, Hu et al., 2020, Zaheer et al., 2018, Zhang et al., 2024, Li et al., 2025]. Recent electrolyte models do use proportions inside the network. GeoMix concatenates each molecule's proportion

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

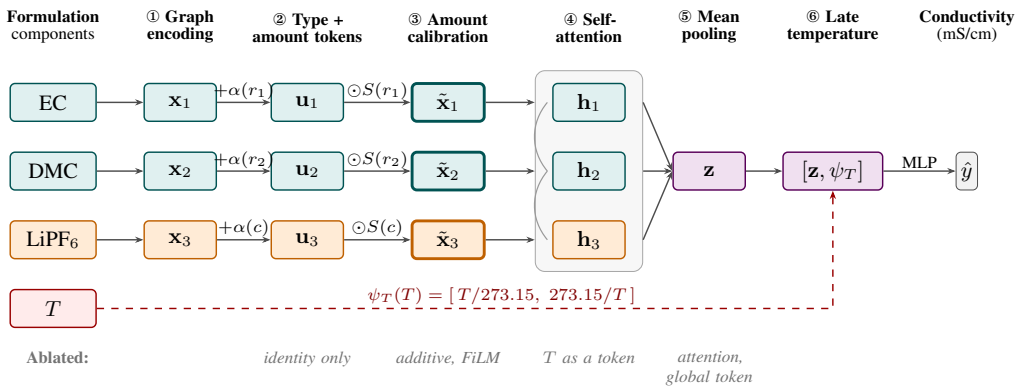


Figure 1: **SolvMix.** Example: EC and DMC (mole ratios r_1, r_2), $LiPF_6$ (salt mole ratio c), temperature T. ① Type-specific graph encoders give descriptors $\mathbf{x}_k$. ② A type embedding and an amount embedding are added. ③ A learned scale $S_{\tau_k}(m_k)$ rescales each token *before* the components interact. ④–⑤ Self-attention and mean pooling give the mixture state $\mathbf{z}$. ⑥ Temperature features bypass mixing and are concatenated to $\mathbf{z}$ at readout. Grey: alternatives compared in Section 4.3.

to its atom features and adds temperature after mixing [Li et al., 2025]; Bamboo-mixer multiplies each molecular embedding by its molar ratio before self-attention and fits a VFT-type law for temperature [Yang et al., 2026]. GeoMix reports that concatenation works better than ratio weighting; Bamboo-mixer uses ratio weighting. Neither is compared with a learned modulation of the component representation, so the form this conditioning should take remains open.

Our answer is that **amounts should scale each feature of a component, before the components interact.** If an amount only adds an offset to a component's embedding, changing the amount moves every molecule of that type in the same way, and any molecule-specific effect of amount must be recovered after mixing (Proposition 1). Ratio weighting rescales all features of a molecule by the same factor. A learned per-feature scale $S_\tau(m)$ contains both as special cases and lets the amount change which features of a molecule dominate. Temperature follows GeoMix and Bamboo-mixer and enters after mixing, as $[T, 1/T]$ features concatenated to the mixture state.

We implement these choices in SolvMix (Figure 1), a model with 64-dimensional hidden states and no dataset-specific tuning, and evaluate it on two experimental datasets, CALiSol-23 [De Blasio et al., 2024] and Bamboo-mixer [Yang et al., 2026], and on the synthetic DiffMix dataset [Zhu et al., 2024].

- SolvMix achieves the lowest test error on two experimental benchmarks and one synthetic benchmark (five-seed means). On the experimental datasets it reduces MSE by 30% and 39% relative to GeoMix-TFN, the strongest prior model (Section 4.2).
- The form of amount conditioning matters more than the molecular encoder. Shifting instead of scaling raises MSE from 0.27 to 0.65, and routing temperature only through attention raises it to 0.62, while changing the graph encoder moves it by at most 0.03 (Section 4.3).
- When contiguous regions of solvent ratio, salt ratio, or temperature are held out, SolvMix's error grows less than GeoMix-TFN's, and its MSE is 49–62% lower (Section 4.4).

2 How should amounts enter the network?

Setup. A formulation is $x = (\mathcal{G}^{\text{solv}}, G^{\text{salt}}, \mathbf{r}, c, T)$: solvent graphs $\{G_i^{\text{solv}}\}_{i=1}^{n_s}$, one salt graph, solvent mole ratios $\mathbf{r} \in \Delta^{n_s-1}$, salt mole ratio c, and temperature T. The target is conductivity $y \geq 0$ in mS/cm. We use mole ratios throughout because they are dimensionless; converting from molality (mol/kg) requires only the solvent composition, while converting from molarity (mol/L) also requires the solution density. Component $k \in \{1, \ldots, n_s{+}1\}$ has a descriptor $\mathbf{x}_k = \phi_{\tau_k}(G_k) \in \mathbb{R}^d$ from a type-specific graph encoder, a type $\tau_k \in \{\text{solv}, \text{salt}\}$, and an amount m_k (r_i or c).

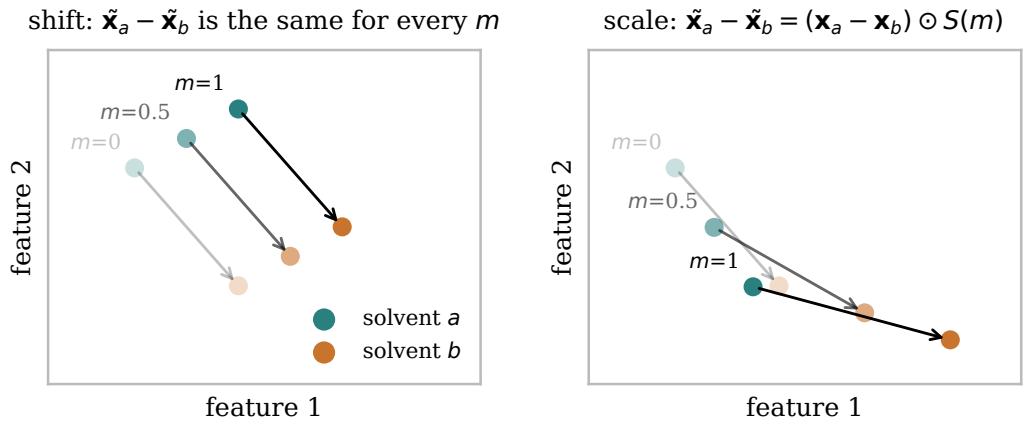


Figure 2: **Proposition 1 in two dimensions** (schematic). Two solvents at the same amount m, before mixing. A shift moves both by the same vector, so the difference between them (arrow) does not depend on m. A scale stretches each feature by an amount-dependent factor, so m can enlarge the features in which the solvents differ and shrink others.

Design space. We write a mixture model as four stages,

$$\mathbf{x}_k \xrightarrow{A} \mathbf{u}_k \xrightarrow{\Gamma} \tilde{\mathbf{x}}_k \xrightarrow{M} \mathbf{z} \xrightarrow{h} \hat{s}, \qquad \hat{y} = e^{\hat{s}} - 1, \tag{1}$$

where A builds a token, Γ calibrates it by amount (we call $\tilde{\mathbf{x}}_k$ the *formulation state* of component k), M mixes the set of calibrated tokens into a mixture state, and h reads out log-conductivity. Temperature can enter M, as an extra token, or h, as a readout input. Existing models differ mainly in ϕ and M, and either omit Γ or fix it; we study Γ and where T enters.

Shifting versus scaling. Let tokens carry type and amount, $\mathbf{u}_k = \mathbf{x}_k + \mathbf{e}_{\tau_k} + \alpha_{\tau_k}(m_k)$, with learned $\mathbf{e}_\tau \in \mathbb{R}^d$ and $\alpha_\tau : \mathbb{R} \to \mathbb{R}^d$. We compare three calibrations with learned maps $S_\tau, \Delta_\tau : \mathbb{R} \to \mathbb{R}^d$:

$$\tilde{\mathbf{x}}_k = \underbrace{\mathbf{u}_k + \Delta_{\tau_k}(m_k)}_{\text{shift}}, \qquad \underbrace{\mathbf{u}_k \odot S_{\tau_k}(m_k)}_{\text{scale}}, \qquad \underbrace{S_{\tau_k}(m_k) \odot \mathbf{u}_k + \Delta_{\tau_k}(m_k)}_{\text{FiLM}}. \tag{2}$$

Proposition 1. (i) *Shift calibration represents exactly the same component states as no calibration ($\tilde{\mathbf{x}}_k = \mathbf{u}_k$).* (ii) *Under shift calibration, changing a component's amount from m to m' changes its state by $\alpha'_\tau(m') - \alpha'_\tau(m)$ with $\alpha'_\tau = \alpha_\tau + \Delta_\tau$ (by $\alpha_\tau(m') - \alpha_\tau(m)$ without calibration); this is the same vector for every molecule of type τ. Under scale calibration the change is $\mathbf{x}_k \odot \big(S_\tau(m') - S_\tau(m)\big) + \mathbf{c}_\tau(m, m')$, where $\mathbf{c}_\tau$ does not depend on the molecule; the first term does.* (iii) *Consequently, two components a, b of type τ at a common amount m satisfy $\tilde{\mathbf{x}}_a - \tilde{\mathbf{x}}_b = \mathbf{x}_a - \mathbf{x}_b$ for every m under shift, and $(\mathbf{x}_a - \mathbf{x}_b) \odot S_\tau(m)$ under scale.*

Proof. (i) $\mathbf{u}_k + \Delta_\tau(m_k) = \mathbf{x}_k + \mathbf{e}_\tau + \alpha'_\tau(m_k)$ with $\alpha'_\tau = \alpha_\tau + \Delta_\tau$, which is an uncalibrated token. (ii) Under shift, $\mathbf{x}_k$ and $\mathbf{e}_\tau$ cancel in $\tilde{\mathbf{x}}_k(m') - \tilde{\mathbf{x}}_k(m)$. Under scale, expanding $(\mathbf{x}_k + \mathbf{e}_\tau + \alpha_\tau(m)) \odot S_\tau(m)$ gives the stated form with $\mathbf{c}_\tau(m, m') = (\mathbf{e}_\tau + \alpha_\tau(m')) \odot S_\tau(m') - (\mathbf{e}_\tau + \alpha_\tau(m)) \odot S_\tau(m)$. (iii) Set $m_a = m_b$ and subtract. □

With a shift, the amount moves every solvent in the same direction by the same distance, whichever solvent it is (Figure 2). Attention and the readout can still couple identity and amount; the proposition determines where that coupling has to be learned. With a shift, the components enter the mixing layers with an amount signal that ignores their identity, and those layers must learn how amount changes each molecule's role. With a scale, the amount can amplify or suppress specific features of a molecule before mixing, for example features that separate a high-dielectric solvent from a low-viscosity one. Two other forms are special cases of the scale. Weighting each embedding by its molar ratio before attention, as Bamboo-mixer does [Yang et al., 2026], corresponds to $S_\tau(m) = m\mathbf{1}$: it rescales the difference between two components uniformly and cannot change its direction. FiLM adds an offset to the scale, so whether that offset helps is an empirical question (Section 4.3).

Table 1: **Dataset statistics** after deduplication. T in K, σ in mS/cm. Last four columns: formulations with 1, 2, 3, and ≥ 4 solvents.

Dataset	Samples	Salts	Solv.	T range	σ mean$\pm$std	1	2	3	$\geq$4
CALiSol-23	13,269	13	38	194–477	4.13 ± 4.08	2,233	9,719	630	687
DiffMix	24,822	2	6	243–293	2.62 ± 2.35	3,420	14,322	7,080	0
Bamboo-mixer	10,397	12	63	194–393	4.30 ± 4.13	2,025	7,462	351	559

Where temperature enters. Ionic conductivity of liquid electrolytes is commonly described by $\ln \sigma = \ln A - E_a/(k_B T)$ (Arrhenius) or $\ln \sigma = \ln A - B/(T - T_0)$ (VFT), where the prefactor, activation term, and T_0 depend on composition. Both laws separate composition-dependent coefficients from a temperature dependence of known form. Concatenating $[T, 1/T]$ to the mixture state $\mathbf{z}$ gives the readout direct access to temperature and to the Arrhenius variable; this is an inductive bias rather than a constraint, and the VFT dependence on $1/(T - T_0)$ is left to the MLP. Bamboo-mixer instead predicts the coefficients of an empirical VFT-type law explicitly [Yang et al., 2026].

3 SolvMix

SolvMix uses scale calibration and a direct temperature path to the readout; Appendix B.1 gives full details. Solvents and salts are encoded by two edge-conditioned message-passing networks with identical architecture and separate weights (three blocks, mean pooling over atoms). Tokens are built and scaled as in Eq. (2), with S_τ and α_τ implemented as MLPs. The scaled tokens pass through three pre-norm Transformer blocks [Vaswani et al., 2017] with padding masks for variable numbers of solvents and are mean-pooled into $\mathbf{z}$. The readout is $\hat{s} = h([\mathbf{z}, \psi_T(T)])$ with $\psi_T(T) = [T/273.15,\ 273.15/T]$; h is a LayerNorm MLP whose Softplus output keeps $\hat{s} \geq 0$, so $\hat{y} \geq 0$. We minimize $\frac{1}{N}\sum_n (\hat{s}_n - \log(1 + y_n))^2$ because conductivity is right-skewed over several orders of magnitude, and report all metrics on the original scale.

4 Experiments

4.1 Setup

Benchmarks. **CALiSol-23** (13,269 formulations, 13 salts, 38 solvents, 194–477 K) collects experimental measurements from the literature; its salt amounts were converted from mol/L or mol/kg to mole ratios, which can introduce noise (Section 4.5). **DiffMix** (24,822; 2 salts, 6 solvents, 243–293 K) is synthetic: its labels are generated by a physics-based simulation on a regular grid, so they are smooth and nearly noise-free. **Bamboo-mixer** (10,397; 12 salts, 63 solvents, 194–393 K) is experimental and has the widest solvent coverage. We use the processed CALiSol-23 and DiffMix releases of GeoMix [Li et al., 2025] and the data released with Bamboo-mixer [Yang et al., 2026], after deduplication (Appendix A). Figure 3 and Table 1 compare the three datasets. DiffMix's temperatures sit on six grid values and its salts are only $LiPF_6$ and LiTFSI, whereas the two experimental datasets have broad, overlapping temperature distributions and long-tailed salt vocabularies. Binary solvent mixtures are the most common in all three, so the main source of variation within a solvent system is the ratio, not the identity.

Protocol. We use random 70/20/10 splits. Every SolvMix model is trained for a fixed 50,000 full-batch epochs with AdamW and weight EMA, using one set of hyperparameters for all datasets, and evaluated at the last step; no checkpoint is selected on validation or test data. Hyperparameters: $d = 64$, three graph blocks, three Transformer blocks with four heads, no dropout, learning rate 5×10^{-4} with exponential decay, gradient clipping 0.5, EMA decay 0.999 (Appendix B.2). Full-batch training removes mini-batch noise as a source of difference between ablation variants. We report mean $\pm$ std over five seeds. On CALiSol-23 and DiffMix we retrain the full baseline suite of GeoMix on our splits: an MLP, MM-MoLFormer, MolSets, EGNN and TFN with attention or linear mixing, and GeoMix-EGNN/TFN. On Bamboo-mixer we retrain GeoMix-TFN, the strongest of these. We also evaluate the released Bamboo-mixer checkpoint for reference; its training split is not public, so we can only evaluate it on the full dataset.

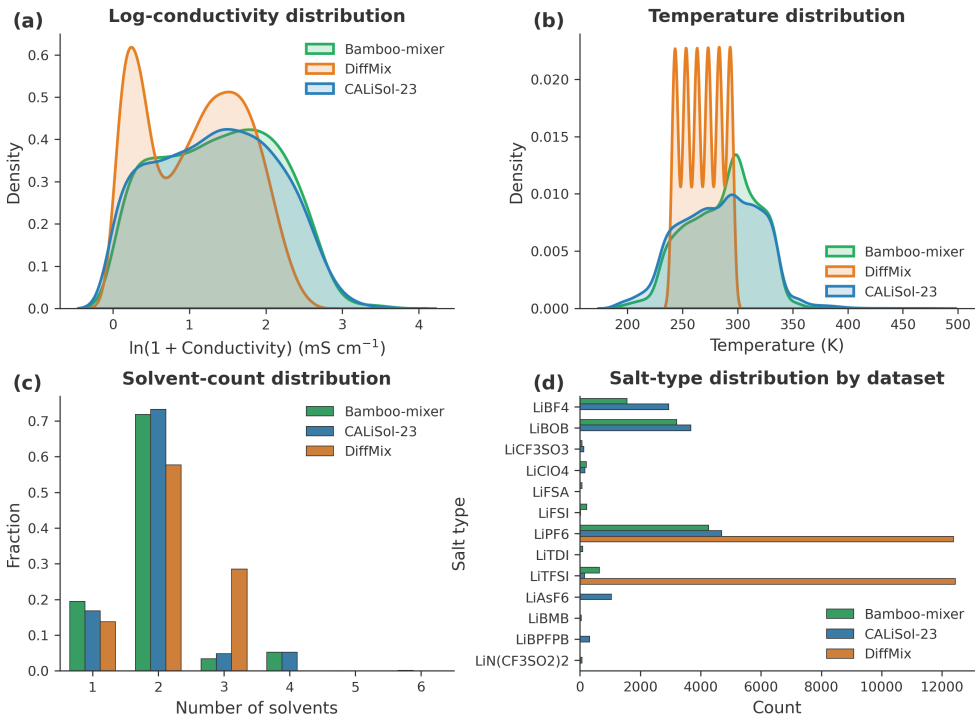


Figure 3: **The three benchmarks.** (a) Log-conductivity, (b) temperature, (c) number of solvents per formulation, (d) salt counts. Green: Bamboo-mixer; orange: DiffMix; blue: CALiSol-23.

4.2 Main results

Table 2 shows that SolvMix has the lowest error on every benchmark; the only metric it does not lead is Spearman ρ on CALiSol-23, where it ties GeoMix-EGNN (0.9893 vs. 0.9894). On the two experimental datasets, it lowers MSE relative to GeoMix-TFN by 30% (CALiSol-23) and 39% (Bamboo-mixer), and on Bamboo-mixer MAE drops by 44%. GeoMix relies on atom-level geometric interaction and an auxiliary denoising objective; SolvMix uses neither. On DiffMix, SolvMix fits the test set almost exactly (MSE 2.1×10^{-4} against 0.095 for GeoMix-TFN). DiffMix is synthetic: its labels are a smooth simulated function sampled on a dense grid, which explains the near-zero error. It shows that the model can recover a clean response surface, not that it is accurate on measured data. The experimental datasets, CALiSol-23 and Bamboo-mixer, are the test that matters. The seed variation on CALiSol-23 is much larger for MSE (40% of the mean) than for MAE (9%): the few test points with large errors differ from seed to seed, in line with the label noise analyzed in Section 4.5.

4.3 Which design choices matter

Figure 4a changes one design choice at a time on CALiSol-23 (values in Appendix C). We run ablations on CALiSol-23 because it covers the most salts and the widest temperature range of the three datasets.

Scaling is what matters. Replacing the scale with a shift raises MSE from 0.273 to 0.645. By Proposition 1(i), the shift model is equivalent to having no calibration at all, so this gap measures the value of the calibration step itself. The scale does most of the work on its own: removing the type and amount embeddings from the token while keeping the scale gives MSE 0.303, within seed variation of the default. FiLM includes the scale and adds an offset, but reaches 0.456 and does not improve on the scale alone. By Proposition 1(i), the offset duplicates the amount embedding α_τ, giving the amount a second, additive route into the token; its error lies between the two pure forms.

Temperature should reach the readout directly. Routing temperature only through attention as an extra token raises MSE to 0.625, almost as much as removing calibration. In this variant the

Table 2: **Test performance.** Bold: best per column. SolvMix: mean ± std over five seeds; baselines retrained on our splits, with the MSE reported by GeoMix [Li et al., 2025] in parentheses. ⋆Released checkpoint [Yang et al., 2026] evaluated on the full dataset, which overlaps its training data; shown for reference.

Model	Spearman $\rho \uparrow$	MAE $\downarrow$	MSE $\downarrow$
CALiSol-23			
MLP	0.8889	1.5363	3.5502 (3.657)
MM-MoLFormer	0.8682	1.2876	3.7919 (5.488)
MolSets-Conv	0.9436	0.9560	2.1942 (2.230)
MolSets-SAGE	0.9014	1.0239	2.4934 (2.751)
EGNN-att	0.9006	1.0280	2.5121 (2.666)
TFN-att	0.9608	0.8365	1.7095 (1.808)
EGNN-linear	0.9446	0.8043	1.5877 (1.461)
TFN-linear	0.9364	0.6820	1.1598 (1.107)
GeoMix-EGNN	**0.9894**	0.4817	0.5946 (0.552)
GeoMix-TFN	0.9860	0.2135	0.4311 (0.432)
SolvMix	$0.9893_{\pm.0038}$	$\mathbf{0.1660}_{\pm.0152}$	$\mathbf{0.3037}_{\pm.1230}$
DiffMix (synthetic)			
MLP	0.9544	0.8359	1.5674 (1.363)
MM-MoLFormer	0.9384	0.8983	2.1402 (1.901)
MolSets-Conv	0.9499	0.8540	1.5673 (1.440)
MolSets-SAGE	0.9639	0.6832	1.0347 (0.708)
EGNN-att	0.9683	0.7194	0.7671 (0.752)
TFN-att	0.9712	0.8622	0.7810 (0.804)
EGNN-linear	0.9893	0.2133	0.1863 (0.195)
TFN-linear	0.9858	0.3563	0.2765 (0.285)
GeoMix-EGNN	0.9973	0.1239	0.1040 (0.088)
GeoMix-TFN	0.9988	0.0542	0.0950 (0.035)
SolvMix	**0.99998**	$\mathbf{7.8\times10^{-3}}$	$\mathbf{2.1\times10^{-4}}$
Bamboo-mixer			
Bamboo-mixer⋆	0.9965	0.1625	0.2115
GeoMix-TFN	0.9973	0.1427	0.1674
SolvMix	**0.9990**	$\mathbf{0.0801}_{\pm.0070}$	$\mathbf{0.1019}_{\pm.0410}$

temperature token is not pooled and the readout no longer receives $\psi_T(T)$, so temperature reaches the prediction only through its effect on the component tokens. A direct path from temperature to the readout is therefore the better design, as in GeoMix and Bamboo-mixer.

The rest is second-order. Global-token pooling is within seed variation of mean pooling, and attention pooling is somewhat worse (0.419). Table 3 varies the components that do not concern amount or temperature. Changing the graph encoder to GCN, GAT, GIN, or GraphSAGE moves MSE by at most 0.03. Adding GeoMix's atom-level interaction layers, the core of that model, worsens results (0.324), and adding its denoising objective worsens them further (0.383). Removing LayerNorm or the Softplus from the readout costs about as much as the atom-level layers. Within SolvMix, the largest effects come from how amount and temperature enter, not from the molecular representation.

4.4 Unseen regions of composition and temperature

Formulation design reuses familiar molecules at new amounts and temperatures. We test this setting on Bamboo-mixer by holding out contiguous regions of one coordinate. Temperature and salt ratio are split into quantile bins, solvent ratio is clustered with K-means within each solvent combination, and whole bins or clusters go to validation or test. Table 4 gives the resulting split sizes. Because whole regions are assigned at once, the proportions deviate from 70/20/10; we keep regions intact rather than move samples back into training. Solvent combinations with a single solvent or too few samples fall back to a random split (Appendix A.3), so the solvent-ratio test set also contains some randomly held-out formulations. All models share the same indices for each split.

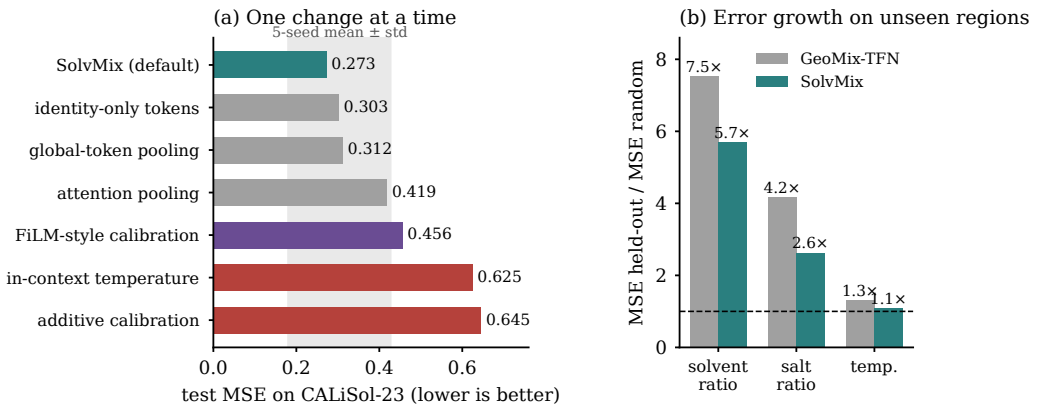


Figure 4: **(a)** Design ablations on CALiSol-23 (seed 42); each bar changes one choice from the default. The grey band is the five-seed mean ± std of the default. Red: replacing the scale by a shift, and removing the direct temperature path. **(b)** Ratio of test MSE on a held-out region to test MSE on the random split, on Bamboo-mixer (random split: five-seed mean for SolvMix).

Table 3: **Implementation choices** on CALiSol-23 (seed 42). Each row changes one component of the default. Full results in Appendix C.2.

Component	Variant	MAE↓	MSE↓
—	SolvMix (edge-conditioned MPNN)	**0.1502**	**0.2727**
Graph encoder	GCN / GAT / GIN / GraphSAGE	0.152–0.159	0.291–0.303
Atom-level interaction	+ GeoMix GIL (3 layers)	0.1626	0.3239
	+ GIL with denoising objective	0.1792	0.3833
Readout	w/o LayerNorm	0.1700	0.3551
	w/o Softplus	0.1751	0.3632
Regularization	dropout $p = 0.1$	0.1643	0.3152

The held-out regions lie inside the overall data range, for example test temperatures of 244–324 K against 194–393 K in training, so these splits test generalization across gaps in the data, not extrapolation beyond it. We do not use CALiSol-23 for this experiment because its converted salt ratios are unreliable, and DiffMix's grid is too narrow for held-out regions to be informative.

SolvMix is better on all three splits (Table 5), and the gap is larger when the held-out coordinate is an amount: MSE falls by 54% for solvent ratio and 62% for salt ratio, compared with 49% for temperature. Figure 4b shows the same pattern as error growth. Moving from the random split to an unseen salt-ratio region multiplies GeoMix-TFN's MSE by 4.2 and SolvMix's by 2.6. The ordering of formulations degrades even less than the error. On solvent-ratio holdout, SolvMix keeps Spearman $\rho = 0.989$ against 0.961 for GeoMix-TFN, which matters for screening, where candidate recipes are ranked rather than predicted exactly. Unseen solvent ratios remain the hardest case for both models, since a held-out cluster can change which solvent dominates the mixture.

4.5 When it fails

Sensitivity to amounts helps when amounts are measured accurately and hurts when they are not. Figure 5 shows the same chemical system, $LiBF_4$ in propylene carbonate, learned from two datasets. From Bamboo-mixer, SolvMix learns a smooth surface that rises with temperature and salt ratio and fits the test points. From CALiSol-23, the surface has sharp ridges along c, and the fit is poor even on the training points ($R^2 = 0.62$; test 0.48); the PC/DEC system shows the same ridges. CALiSol-23 merges sources that report salt amounts in mol/L or mol/kg. Converting molality to a mole ratio requires only the solvent composition, but converting molarity also requires the solution density, which depends on temperature and composition and is rarely measured. Errors in that conversion, together with differences between laboratories and protocols, can make nearby points

Table 4: **Split sizes** on Bamboo-mixer (10,397 formulations) and the held-out test range of each coordinate.

Split	Train	Val.	Test	Test region
Random	7,277 (70.0%)	2,079 (20.0%)	1,041 (10.0%)	—
Temperature	7,761 (74.6%)	955 (9.2%)	1,681 (16.2%)	$T \in [244, 324]$ K
Salt ratio	7,291 (70.1%)	1,031 (9.9%)	2,075 (20.0%)	$c \in [0.022, 0.163]$
Solvent ratio	7,424 (71.4%)	1,121 (10.8%)	1,852 (17.8%)	held-out ratio clusters

Table 5: **Held-out regions** on Bamboo-mixer. Each split removes contiguous regions of one coordinate from training; train and test mostly share the same molecules.

Held-out	Model	$\rho \uparrow$	MAE↓	MSE↓
Solvent ratio	GeoMix-TFN	0.9613	0.4567	1.2620
	SolvMix	**0.9892**	**0.2286**	**0.5817**
Salt ratio	GeoMix-TFN	0.9689	0.3891	0.6964
	SolvMix	**0.9953**	**0.1964**	**0.2677**
Temperature	GeoMix-TFN	0.9856	0.2914	0.2188
	SolvMix	**0.9990**	**0.1828**	**0.1114**

in the converted coordinate carry conflicting labels. No smooth function of c fits such labels, and because SolvMix scales component states by c, the misfit appears as ridges along that coordinate. Trained on Bamboo-mixer, the same architecture learns a smooth and accurate surface for the same chemical system.

On the cleaner Bamboo-mixer data, error does not grow with mixture complexity (Figure 6). MAE for one- to four-solvent mixtures stays between 0.036 and 0.054 mS/cm; the five-solvent group has only three samples. Across temperature, the 253–283 K bin has an MSE of 0.145, three times the overall 0.045, while the coldest bin is the most accurate, so the remaining error is not simply a matter of low temperature.

5 Discussion

Allocating capacity in mixture models. Our ablations separate two kinds of change. Changes to the molecular side (four graph encoders, atom-level interaction layers, a denoising objective) move CALiSol-23 MSE by at most 0.11, and none of them helps. Changes to where formulation variables enter move it by up to 0.37. On these benchmarks, how formulation variables enter matters more than how molecules are encoded. The form of conditioning adds almost no parameters, so it is a cheap axis to tune before adding molecular capacity.

Matching the structure of the physics. Both of SolvMix's choices are motivated by structure in conductivity models without fixing a functional form. Late temperature mirrors the separation in Arrhenius and VFT laws between composition-dependent coefficients and a temperature dependence of known form. Per-feature scaling lets the effect of an amount depend on the molecule, as a component's contribution to dielectric screening or viscosity does. DiffMix and Bamboo-mixer impose a VFT-type law explicitly; SolvMix keeps only the separation the law implies and learns the rest.

Data quality as the bottleneck. The largest remaining errors occur where the amount coordinate is least reliable (Section 4.5). A model that uses amounts precisely also reproduces errors in them. Reporting the original unit together with the solution density, or the amount directly as a mole ratio, would make literature data more useful for learning.

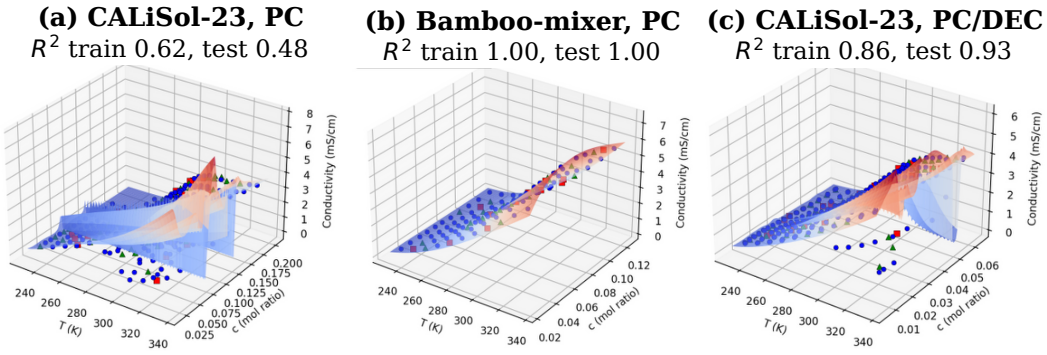


Figure 5: **Learned conductivity surfaces for $LiBF_4$ formulations** as a function of temperature T and salt mole ratio c. Points: training (blue), validation (green), and test (red) samples. The same chemical system yields a rugged surface when learned from CALiSol-23 (a) and a smooth one when learned from Bamboo-mixer (b); the ridges along c recur for PC/DEC (c). More views in Appendix D.

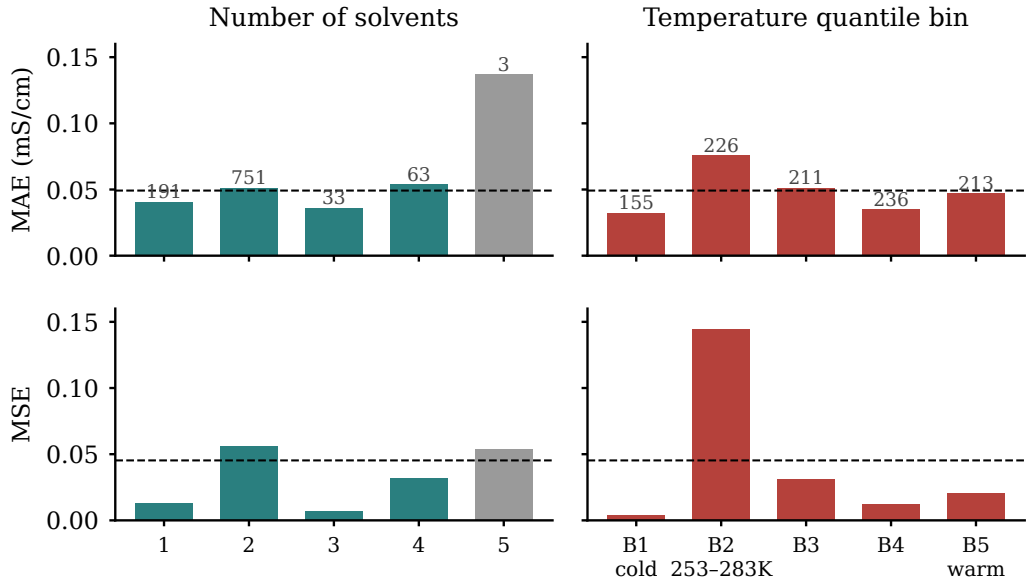


Figure 6: **Where the Bamboo-mixer test error lies** (random split, seed 42, $n = 1{,}041$), by number of solvents (left) and temperature quantile bin (right). Numbers give subgroup sizes; dashed lines give the error on the full test set; grey marks a subgroup too small to estimate. All metrics in Appendix E.1.

6 Related work

Electrolyte models. CALiSol-23 compiles literature conductivity data [De Blasio et al., 2024]. DiffMix fits the coefficients of a VFT law with a differentiable mixture model [Zhu et al., 2024], and Bamboo-mixer adds an empirical VFT-type head [Yang et al., 2026]. BAMBOO predicts properties with learned force fields and molecular dynamics [Gong et al., 2025], and Bamboo-mixer couples prediction with generative formulation design [Yang et al., 2026]. Cluster expansions [Müller and Natarajan, 2025], equivariant salt models [Shen et al., 2025], and hybrid thermodynamic models [Guan, 2022, Jirasek et al., 2020] add physical structure to mixture prediction.

Mixture and set learning. Graph networks for single molecules, including message passing [Gilmer et al., 2017], pretrained encoders [Hu et al., 2020], continuous-filter and directional convolu-

tions [Schütt et al., 2017, Gasteiger et al., 2022], and equivariant networks [Satorras et al., 2022], provide the component encoders that mixture models build on; in our encoder ablation the choice among them changes conductivity error little. Deep Sets [Zaheer et al., 2018] and the Set Transformer [Lee et al., 2019] define permutation-invariant models over sets; MolSets applies set aggregation to molecular mixtures [Zhang et al., 2024]. GeoMix adds atom-level geometric interaction between components, concatenates proportions to atom features, and adds temperature after mixing; in its ablation, concatenation beats weighting molecule embeddings by their proportion [Li et al., 2025]. Bamboo-mixer multiplies molecular embeddings by their molar ratios before self-attention [Yang et al., 2026], a fixed scalar instance of our scale calibration. These works improve how components are represented and combined. We study the step before combination: the form in which amount modifies each component, with a learned per-feature scale that generalizes ratio weighting.

Feature modulation. FiLM [Perez et al., 2017] and conditional normalization [Dumoulin et al., 2017] modulate features with an affine function of a conditioning signal. Here the signal is the amount of each component, applied separately to each set element; in our ablation, the extra offset of the affine form does not help.

7 Limitations

Our ablations vary one design choice at a time within SolvMix, using one seed on CALiSol-23. They compare shift, scale, and FiLM calibration, but not fixed molar-ratio weighting or concatenation of amounts inside the same architecture, and they compare late temperature with a temperature token, not with both combined. Random splits are drawn per sample, so the same formulation measured at different temperatures can appear in training and test; the region-held-out splits instead withhold entire ranges of one coordinate. All benchmarks contain lithium salts in carbonate-dominated solvents with one salt per formulation, and the held-out regions lie inside the observed ranges. Transfer to new solvent families, multi-salt and additive systems, and extrapolation beyond measured conditions remain to be tested. The learned scale $S_\tau(m)$ is fit to conductivity alone and is not a physical quantity.

8 Conclusion

How the recipe enters a mixture model deserves the same attention as how its molecules are encoded. For electrolyte conductivity, letting amounts rescale each feature of a component before mixing gives a small model that leads on both experimental benchmarks and degrades less on unseen compositions. In our ablations, this choice and the direct temperature path each matter more than the molecular encoder. The same question applies to any property of a mixture of known components in varying proportions, such as polymer blends, alloys, and drug combinations, and is cheap to test. The main obstacle we found is data quality: when amounts are converted between units without density information, a model that uses them accurately also reproduces the conversion errors.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China under Grant W2431046, the National Key R&D Program of China under Grant 2025YFA1309603, the Central Guided Local Science and Technology Foundation of China under Grant YDZX20253100001001, the MoE Key Lab of Intelligent Perception and Human-Machine Collaboration (ShanghaiTech University), and the Shanghai Frontiers Science Center of Human-centered Artificial Intelligence.

Competing interests. G. Qiu, Y. Zhang, and X. Zhao are employees of Contemporary Amperex Technology Co., Limited (CATL); K. Zhang, T. Zhao, and Y. Mei are affiliated with Shanghai Smart Logic Technology; M. Li is affiliated with Cellverse Co., Ltd. The authors declare no other competing interests.

References

Kang Xu. Nonaqueous liquid electrolytes for lithium-based rechargeable batteries. *Chemical Reviews*, 104(10):4303–4418, 2004. doi: 10.1021/cr030203g.

E. R. Logan and J. R. Dahn. Electrolyte design for fast-charging Li-ion batteries. *Trends in Chemistry*, 2(4):354–366, 2020. doi: 10.1016/j.trechm.2020.01.011.

Chen Ling. A review of the recent progress in battery informatics. *npj Computational Materials*, 8 (1):33, February 2022. ISSN 2057-3960. doi: 10.1038/s41524-022-00713-x.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry, June 2017.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for Pre-training Graph Neural Networks, February 2020.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets, April 2018.

Hengrui Zhang, Tianxing Lai, Jie Chen, Arumugam Manthiram, James M. Rondinelli, and Wei Chen. Learning Molecular Mixture Property Using Chemistry-Aware Graph Neural Network. *PRX Energy*, 3(2):023006, June 2024. ISSN 2768-5608. doi: 10.1103/PRXEnergy.3.023006.

Anyi Li, Jiacheng Cen, Songyou Li, Mingze Li, Yang Yu, and Wenbing Huang. Geometric Mixture Models for Electrolyte Conductivity Prediction, October 2025.

Zhenze Yang, Yifan Wu, Xu Han, Ziqing Zhang, Haoen Lai, Zhenliang Mu, Tianze Zheng, Siyuan Liu, Zhichen Pu, Zhi Wang, Zhiao Yu, Sheng Gong, and Wen Yan. A unified predictive and generative solution for liquid electrolyte formulation. *Nature Machine Intelligence*, 8(2):186–196, January 2026. ISSN 2522-5839. doi: 10.1038/s42256-025-01173-w.

Paolo De Blasio, Jonas Elsborg, Tejs Vegge, Eibar Flores, and Arghya Bhowmik. CALiSol-23: Experimental electrolyte conductivity data for various Li-salts and solvent combinations. *Scientific Data*, 11(1):750, July 2024. ISSN 2052-4463. doi: 10.1038/s41597-024-03575-8.

Shang Zhu, Bharath Ramsundar, Emil Annevelink, Hongyi Lin, Adarsh Dave, Pin-Wen Guan, Kevin Gering, and Venkatasubramanian Viswanathan. Differentiable modeling and optimization of non-aqueous Li-based battery electrolyte solutions using geometric deep learning. *Nature Communications*, 15(1):8649, October 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-51653-7.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Sheng Gong, Yumin Zhang, Zhenliang Mu, Zhichen Pu, Hongyi Wang, Zhiao Yu, Mengyi Chen, Tianze Zheng, Zhi Wang, Lifei Chen, Zhenze Yang, Xiaojie Wu, Shaochen Shi, Weihao Gao, Wen Yan, and Liang Xiang. A predictive machine learning force field framework for liquid electrolyte development, April 2025.

Yann L. Müller and Anirudh Raju Natarajan. Constructing multicomponent cluster expansions with machine-learning and chemical embedding. *npj Computational Materials*, 11(1):60, March 2025. ISSN 2057-3960. doi: 10.1038/s41524-025-01543-3.

Chen Shen, Siamak Attarian, Yixuan Zhang, Hongbin Zhang, Mark Asta, Izabela Szlufarska, and Dane Morgan. SuperSalt: Equivariant neural network force fields for multicomponent molten salts system. *Nature Communications*, 16(1):7280, August 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-62450-1.

Pin-Wen Guan. Differentiable thermodynamic modeling. *Scripta Materialia*, 207:114217, January 2022. ISSN 13596462. doi: 10.1016/j.scriptamat.2021.114217.

Fabian Jirasek, Robert Bamler, and Stephan Mandt. Hybridizing physical and data-driven prediction methods for physicochemical properties. *Chemical Communications*, 56(82):12407–12410, 2020. ISSN 1359-7345, 1364-548X. doi: 10.1039/D0CC05258B.

Kristof T. Schütt, Pieter-Jan Kindermans, Huziel E. Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions, December 2017.

Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional Message Passing for Molecular Graphs, April 2022.

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) Equivariant Graph Neural Networks, February 2022.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3744–3753, 2019.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer, December 2017.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A Learned Representation For Artistic Style, February 2017.

A Dataset details

A.1 Dataset sources, preprocessing, and distributional statistics

Dataset sources. We construct three curated electrolyte conductivity benchmarks: CALiSol-23, DiffMix, and Bamboo-mixer. Unless otherwise specified, CALiSol-23 and DiffMix denote the curated datasets derived from the processed conductivity datasets used in GeoMix, rather than the GeoMix model itself. Bamboo-mixer denotes the curated experimental conductivity dataset released with the Bamboo-mixer project. All datasets are converted into the unified formulation format used in the main text:

$$x = (\mathcal{G}^{\mathrm{solv}}, G^{\mathrm{salt}}, \mathbf{r}, c, T), \qquad y = \sigma,$$

where $\mathcal{G}^{\mathrm{solv}}$ is the set of solvent molecular graphs, G^{salt} is the salt molecular graph, $\mathbf{r}$ denotes solvent mole ratios, c denotes salt mole ratio, T is temperature, and $\sigma \geq 0$ is ionic conductivity.

Unit normalization and filtering. Temperature is represented in Kelvin and conductivity in $\mathrm{mS/cm}$. Solvent amounts are normalized as mole ratios among solvent components. The salt amount c is represented as a salt mole ratio relative to the total formulation, where the cation–anion pair is treated as one salt unit. For CALiSol-23, original salt amounts may be reported in concentration units such as $\mathrm{mol/L}$ or $\mathrm{mol/kg}$; we use the converted mole-ratio representation inherited from the processed GeoMix data. As discussed in Appendix D, this conversion can introduce uncertainty: values reported in mol/kg convert using the solvent composition alone, but values reported in mol/L also require the solution density, which varies with temperature, solvent composition, salt identity, and salt amount. After unit normalization, each dataset is deduplicated by component identities, solvent ratios, salt mole ratio, temperature, and conductivity. We remove records with missing, non-numeric, negative, or otherwise invalid conductivity, as well as records with missing solvent components or missing salt identity.

Dataset statistics. Table 6 reports dataset-level statistics after preprocessing and deduplication. The three benchmarks cover complementary regions of formulation space. DiffMix is the largest by sample count but chemically and thermally narrow. Bamboo-mixer has the largest solvent vocabulary and is therefore used as the primary OOD benchmark. CALiSol-23 spans the widest temperature range but contains stronger source heterogeneity and salt-amount conversion uncertainty, as diagnosed further in Appendix D.

Formulation complexity. Table 1 reports the number of solvent components per formulation. Binary solvent systems dominate all three datasets, but single-solvent and higher-order mixtures are also present; the largest mixtures, in Bamboo-mixer, have five (7 formulations) or six (8) solvents. This motivates a variable-size formulation representation rather than a fixed-size molecular-pair predictor.

Table 6: Dataset statistics after preprocessing and deduplication. Temperature is reported in Kelvin and conductivity in $\mathrm{mS/cm}$.

Dataset	#Samples	Temp. range	Temp. mean/std	Cond. range	Cond. mean/std	#Salts	#Solvents
Bamboo-mixer	10,397	194.15–393.18	286.55 / 31.56	0.005864–41.854850	4.2977 / 4.1306	12	63
DiffMix	24,822	243.15–293.15	268.19 / 17.09	0.000044–12.600000	2.6166 / 2.3492	2	6
CALiSol-23	13,269	194.15–477.42	284.24 / 35.81	0.000000–38.100000	4.1309 / 4.0849	13	38

Table 7: Salt-type breakdown over the combined preprocessed datasets.

Salt	Count	Percentage
$LiPF_6$	21,339	44.01
LiTFSI	13,239	27.30
LiBOB	6,888	14.21
$LiBF_4$	4,499	9.28
$LiAsF_6$	1,040	2.14
$LiClO_4$	374	0.77
LiBPFPB	318	0.66
LiFSI	244	0.50
$LiCF_3SO_3$	197	0.41
LiTDI	110	0.23
LiFSA	70	0.14
$LiN(CF_3SO_2)_2$	70	0.14
LiBMB	40	0.08
LiDFOB	24	0.05
LiFNFSI	14	0.03
LiPDI	13	0.03
LiCTFSI	9	0.02

Dataset-level distributional differences. The three datasets differ substantially in conductivity scale, temperature coverage, formulation complexity, and chemical diversity. DiffMix is dense but narrow, with temperatures concentrated between 243.15 and 293.15 K and only two salt species. Bamboo-mixer and CALiSol-23 span broader temperature ranges and contain richer solvent and salt vocabularies. Conductivity values are strongly right-skewed across all datasets, motivating log-space training with $s = \log(1 + y)$. These distributional differences indicate that the benchmarks should not be treated as samples from a single homogeneous distribution and motivate benchmark-specific evaluation protocols.

Salt and solvent-combination coverage. The combined benchmark is dominated by common lithium salts but contains a long tail of less frequent species. The most frequent salts are $LiPF_6$, LiTFSI, LiBOB, and $LiBF_4$, which together account for most samples. Tables 7 and 8 report the salt-type breakdown and the ten most frequent solvent combinations over the combined preprocessed datasets.

Distributional diagnostics. Figure 3 in the main text shows the log-conductivity distributions, temperature coverage, solvent-count distributions, and salt-type coverage of the three datasets. CALiSol-23 is used for standard evaluation and ablation; its source heterogeneity and salt-amount conversions make region-held-out results harder to interpret. DiffMix is synthetic and dense, and its narrow temperature and chemical range makes held-out regions uninformative. Bamboo-mixer combines sample size, solvent diversity, temperature coverage, and experimental consistency, and is used for the region-held-out analysis.

A.2 Molecular graph construction

Each formulation is represented consistently with the main text as

$$x = (\mathcal{G}^{\mathrm{solv}}, G^{\mathrm{salt}}, \mathbf{r}, c, T),$$

where $\mathcal{G}^{\mathrm{solv}} = \{G_i^{\mathrm{solv}}\}_{i=1}^{n_s}$ is the set of solvent molecular graphs, G^{salt} is the salt molecular graph, $\mathbf{r}$ denotes solvent mole ratios, c denotes salt mole ratio, and T denotes temperature. Each solvent

Table 8: Top ten solvent combinations over the combined preprocessed datasets.

Solvent combination	Count
DEC, PC	11,748
EC, PC	3,690
PC	3,143
EC, EMC	1,887
EA, PC	1,382
DMC, EC	1,380
EC, EMC, PC, TFP	1,075
EMC, FEC	1,013
DEC, EC	978
DMC, EMC	960

Table 9: Atom and bond features used in RDKit molecular graph construction.

Feature type	Feature group	Encoding	Dimension
Atom	Atomic number	One-hot	119
Atom	Atom degree	One-hot	7
Atom	Total hydrogen count	One-hot	5
Atom	Total valence	One-hot	7
Atom	Formal charge	Clipped one-hot	5
Atom	Aromaticity	Binary one-hot	2
Atom	Hybridization	One-hot	5
Atom	Ring membership	Binary one-hot	2
Atom total	–	Concatenation	152
Bond	Bond type	One-hot	4
Bond	Stereochemistry	One-hot	6
Bond	Conjugation	Binary	1
Bond	Aromaticity	Binary	1
Bond	Ring membership	Binary	1
Bond total	–	Concatenation	13

or salt graph is constructed from its SMILES string using RDKit. Atoms are treated as nodes and chemical bonds as edges; each undirected bond is stored as two directed edges for message passing. Invalid or unparsable SMILES strings are excluded from the graph cache. All unique solvent and salt graphs are precomputed once and reused during dataset loading. The atom and bond feature specification is summarized in Table 9.

During batching, solvent graphs and salt graphs are batched separately. Solvent graphs are aligned with their corresponding solvent ratios in $\mathbf{r}$, while the salt graph is aligned with the salt mole ratio c. Graph-to-sample indices and component-level padding masks preserve the variable-size formulation structure.

A.3 Split construction and Bamboo-mixer OOD protocol

Overview. We use two split protocols in this work. For standard benchmark evaluation, CALiSol-23, DiffMix, and Bamboo-mixer are split at the sample level into train, validation, and test sets. For structured extrapolation, we use binned OOD splits only on Bamboo-mixer. This follows the experimental design in Section 4.1: Bamboo-mixer provides the cleanest setting for interpreting held-out solvent-ratio, salt-ratio, and temperature regions as formulation-coordinate shifts.

Random splits for standard evaluation. For each curated dataset, samples are first preprocessed and deduplicated. They are then randomly shuffled with a fixed seed and partitioned into 70% training, 20% validation, and 10% test sets. This split measures interpolation under the empirical data

distribution and is used for the main benchmark comparison in Table 2. The resulting split sizes are:

Dataset	Total	Train	Validation	Test
CALiSol-23	13,269	9,288 (70.0%)	2,653 (20.0%)	1,328 (10.0%)
DiffMix	24,822	17,375 (70.0%)	4,964 (20.0%)	2,483 (10.0%)
Bamboo-mixer	10,397	7,277 (70.0%)	2,079 (20.0%)	1,041 (10.0%)

Random splits preserve the marginal distributions of temperature, salt mole ratio, solvent count, and solvent composition up to sampling noise.

Why OOD evaluation is restricted to Bamboo-mixer. Although the same split generator can construct binned OOD splits for all datasets, we use Bamboo-mixer as the primary OOD benchmark because OOD evaluation is meaningful only when the held-out coordinates are reliable and sufficiently diverse. As shown in Appendix A, Bamboo-mixer provides the best balance among sample scale, solvent diversity, salt coverage, temperature range, and experimental consistency. This makes held-out solvent-ratio, salt-ratio, and temperature regions interpretable as formulation-coordinate shifts rather than artifacts of dataset construction.

CALiSol-23 is still valuable for standard evaluation and ablation because all model variants are compared under the same fixed supervision. However, it is not ideal for primary OOD conclusions. Its salt amounts are inherited from sources using mixed concentration conventions, including $\mathrm{mol/L}$ and $\mathrm{mol/kg}$; converting $\mathrm{mol/L}$ into a salt mole ratio requires the solution density, which depends on temperature, solvent composition, salt identity, and salt amount. As diagnosed in Appendix D, this issue is especially visible in $LiBF_4$–PC-containing systems, where CALiSol-23 produces rugged conductivity surfaces that are much less smooth than the corresponding Bamboo-mixer surface. A binned OOD split on such coordinates may therefore test sensitivity to conversion artifacts and source heterogeneity, rather than clean extrapolation over the physical formulation landscape.

DiffMix presents the opposite limitation. It is the largest benchmark by sample count, but Appendix A shows that it has a narrow solvent vocabulary, only two salt species, and a restricted temperature range. Binned OOD splits on DiffMix are therefore less diagnostic of broad formulation-level transfer: the held-out bins mainly probe a dense but limited region of chemical and thermodynamic space. For these reasons, we use Bamboo-mixer as the main OOD benchmark, while using CALiSol-23 and DiffMix for standard benchmark evaluation and controlled design comparisons.

Binned OOD split principle. OOD splits are constructed by holding out entire regions of the formulation landscape rather than randomly holding out individual samples. For an OOD coordinate v, such as temperature T, salt mole ratio c, or solvent-ratio coordinates, each sample is assigned a region label:

$$b_v(x) \in \{1, \ldots, B_v\}.$$

We then select disjoint held-out regions for validation and testing:

$$\mathcal{D}_{\mathrm{test}} = \{x : b_v(x) \in \mathcal{H}_{\mathrm{test}}\}, \qquad \mathcal{D}_{\mathrm{val}} = \{x : b_v(x) \in \mathcal{H}_{\mathrm{val}}\},$$

and use all remaining regions for training:

$$\mathcal{D}_{\mathrm{train}} = \{x : b_v(x) \notin \mathcal{H}_{\mathrm{test}} \cup \mathcal{H}_{\mathrm{val}}\}.$$

This ensures that the model is evaluated on formulation regions not observed during training.

Temperature and salt-ratio OOD. For temperature OOD and salt-ratio OOD, we bin the selected coordinate using quantile-based boundaries. Quantile binning is preferred over equal-width binning because electrolyte data are highly non-uniform along both temperature and salt amount. Given a coordinate vector v, we compute quantile boundaries, remove duplicate boundaries caused by repeated values, and assign each sample to a non-overlapping bin. A fixed subset of bins is then held out for validation and testing. For temperature OOD, $v = T$. For salt-ratio OOD, $v = c$.

Solvent-ratio OOD. Solvent-ratio OOD is more structured because different solvent sets live in different composition spaces. We first group samples by their sorted solvent-identity tuple. Within each group, we define held-out regions in the corresponding solvent-ratio space. For single-solvent groups, there is no solvent-ratio variation, so the group falls back to a random split. For binary-solvent

Table 10: Bamboo-mixer split sizes. Random is the standard sample-level split. Temperature, salt-ratio, and solvent-ratio splits hold out structured formulation regions for OOD evaluation.

Split mode	Total	Train	Validation	Test
Random	10,397	7,277 (70.0%)	2,079 (20.0%)	1,041 (10.0%)
Temperature OOD	10,397	7,761 (74.6%)	955 (9.2%)	1,681 (16.2%)
Salt-ratio OOD	10,397	7,291 (70.1%)	1,031 (9.9%)	2,075 (20.0%)
Solvent-ratio OOD	10,397	7,424 (71.4%)	1,121 (10.8%)	1,852 (17.8%)

groups, the composition space is one-dimensional, and we apply quantile binning to one solvent ratio. For ternary and higher-order solvent groups, the ratio vector is higher-dimensional. We remove one redundant coordinate and apply K-means clustering to the remaining standardized ratio coordinates. The resulting bins or clusters define solvent-ratio regions, from which validation and test regions are held out. Groups with too few samples or too few valid regions fall back to random splitting to avoid unstable partitions.

Why OOD split ratios are not exactly $70/20/10$. OOD splits are performed at the region level, not at the sample level. Therefore, the resulting train, validation, and test proportions are not forced to match $70/20/10$. This is expected. Bins can contain unequal numbers of samples because empirical formulation data are skewed, repeated values can collapse adjacent quantile boundaries, and solvent-ratio OOD aggregates many solvent-composition groups of different sizes. Forcing the test set back to an exact 10% sample fraction would require moving samples from held-out regions into training, which would break the OOD property. We therefore prioritize regional separation over exact sample proportions.

Bamboo-mixer split sizes. Table 10 summarizes the split sizes used for Bamboo-mixer. The random split is used for standard benchmark evaluation, while the three binned splits are used for OOD evaluation in Section 4.4.

Bamboo-mixer split diagnostics. For the random split, train and test marginals are closely matched. The training temperature range is 194.1–387.3 K and the test range is 194.1–393.2 K, with nearly identical means. The salt mole ratio c is also similarly distributed between train and test.

For temperature OOD, the held-out test set contains $1{,}681$ samples with temperature range 244.1–323.6 K and mean 291.0 K, while the training set covers 194.1–393.2 K with mean 284.4 K. For salt-ratio OOD, the test set contains $2{,}075$ samples concentrated in $c \in [0.02157, 0.16301]$, while the training set retains broader low- and high-salt regions. For solvent-ratio OOD, the test set contains $1{,}852$ samples and retains diverse solvent counts, including single-, binary-, ternary-, quaternary-, and higher-order mixtures. These diagnostics confirm that the OOD splits hold out structured regions of the selected coordinate while preserving broad coverage of the remaining formulation variables.

Reproducibility. All splits use the fixed seed 42, including random sample splitting, held-out bin selection, and K-means initialization. For ratio-vector clustering, solvent-ratio coordinates are standardized before clustering. The resulting train, validation, and test indices are fixed and shared across all models evaluated under the same split.

B Implementation details

B.1 Network implementation

This section specifies the network implementation of SolvMix under the notation of Sections 2 and 3. The implementation follows the hierarchy

$$G_k \to \mathbf{x}_k \to \mathbf{u}_k \to \tilde{\mathbf{x}}_k \to \mathbf{z} \to \hat{s} \to \hat{y}.$$

The main text defines the formulation-state abstraction; here we describe the concrete graph encoder, component-token construction, token-interaction module, pooling variants, temperature formatting, and readout network.

Molecular graph inputs. Each molecular graph $G = (V, E)$ contains atom features and bond features:

$$\mathbf{X}^{\text{atom}} \in \mathbb{R}^{|V| \times d_x}, \qquad \mathbf{A}^{\text{bond}} \in \mathbb{R}^{|E| \times d_e}.$$

We use $d_x = 152$, $d_e = 13$, and hidden dimension $d = 64$ by default. Each undirected chemical bond is represented as two directed edges for message passing.

Type-specific graph encoders. Solvent and salt components are encoded by two type-specific graph encoders,

$$\phi_{\text{solv}}, \qquad \phi_{\text{salt}},$$

with identical architecture but separate parameters. For component k, the descriptor, type, and amount are

$$(\mathbf{x}_k, \tau_k, m_k) = \begin{cases} \left(\phi_{\text{solv}}(G_i^{\text{solv}}), \text{solv}, r_i\right), & k = i, \quad i = 1, \ldots, n_s, \\ \left(\phi_{\text{salt}}(G^{\text{salt}}), \text{salt}, c\right), & k = n_s + 1. \end{cases}$$

Each graph encoder is an edge-conditioned message-passing network. Atom features are first projected into the hidden dimension:

$$\mathbf{h}_v^{(0)} = \mathbf{W}_{\text{emb}} \mathbf{x}_v^{\text{atom}} + \mathbf{b}_{\text{emb}}, \qquad v \in V.$$

For graph layer $\ell = 0, \ldots, L_g - 1$, the message on directed edge $(u, v) \in E$ is

$$\mathbf{m}_{uv}^{(\ell)} = \text{MLP}_{\text{msg}}^{(\ell)} \left([\mathbf{h}_u^{(\ell)} \| \mathbf{h}_v^{(\ell)} \| \mathbf{a}_{uv}] \right),$$

where $\mathbf{a}_{uv} \in \mathbb{R}^{d_e}$ is the bond feature and $\|$ denotes concatenation. Messages are mean-aggregated:

$$\bar{\mathbf{m}}_u^{(\ell)} = \frac{1}{|\mathcal{N}(u)|} \sum_{v \in \mathcal{N}(u)} \mathbf{m}_{uv}^{(\ell)}.$$

The node update is

$$\Delta \mathbf{h}_u^{(\ell)} = \text{MLP}_{\text{node}}^{(\ell)} \left([\mathbf{h}_u^{(\ell)} \| \bar{\mathbf{m}}_u^{(\ell)}] \right),$$

followed by residual connection and layer normalization:

$$\mathbf{h}_u^{(\ell+1)} = \text{LN} \left(\mathbf{h}_u^{(\ell)} + \Delta \mathbf{h}_u^{(\ell)} \right).$$

The default implementation uses $L_g = 3$ graph encoder blocks. All MLPs use SiLU activations between linear layers.

Atom-to-component pooling. After graph encoding, atom states are pooled into component descriptors. For component k, let $\mathcal{V}_k$ denote its atom set. The graph-level descriptor is obtained by mean pooling:

$$\mathbf{x}_k = \frac{1}{|\mathcal{V}_k|} \sum_{v \in \mathcal{V}_k} \mathbf{h}_v^{(L_g)}.$$

The default SolvMix configuration performs mixture interaction at the component-token level. The implementation also supports an optional atom-level inter-graph interaction module before atom-to-component pooling. This module is used only in the architecture exploration study in Appendix C.2, where we insert GeoMix-style GIL layers before SolvMix token interaction.

Component token initialization. Given $(\mathbf{x}_k, \tau_k, m_k)$, SolvMix implements the initialization map A_θ. The two variants follow Section 3:

$$\mathbf{u}_k = A_\theta(\mathbf{x}_k, \tau_k, m_k) = \begin{cases} \mathbf{x}_k, & \text{identity-only}, \\ \mathbf{x}_k + \mathbf{e}_{\tau_k} + \alpha_{\tau_k}(m_k), & \text{amount-and-type-aware}. \end{cases}$$

Here $\mathbf{e}_{\tau_k} \in \mathbb{R}^d$ is a learnable type embedding, and α_{τ_k} is a type-specific amount encoder:

$$\alpha_{\text{solv}}(r_i) = \text{MLP}_{\text{solv_amount}}(r_i), \qquad \alpha_{\text{salt}}(c) = \text{MLP}_{\text{salt_amount}}(c).$$

The default configuration uses amount-and-type-aware initialization.

Amount-calibrated component states. SolvMix then applies the amount-calibration map Γ_θ. The default implementation uses multiplicative calibration:

$$\tilde{\mathbf{x}}_k = \mathbf{u}_k \odot S_{\tau_k}(m_k),$$

where $S_{\tau_k} : \mathbb{R} \to \mathbb{R}^d$ is a type-specific amount-conditioned scale map:

$$S_{\text{solv}}(r_i) = \text{MLP}_{\text{solv_scale}}(r_i), \qquad S_{\text{salt}}(c) = \text{MLP}_{\text{salt_scale}}(c).$$

With amount-and-type-aware initialization and multiplicative calibration enabled, the effective component-state construction is

$$\tilde{\mathbf{x}}_k = (\mathbf{x}_k + \mathbf{e}_{\tau_k} + \alpha_{\tau_k}(m_k)) \odot S_{\tau_k}(m_k).$$

The additive and FiLM-style variants replace this multiplicative scale with the corresponding displacement or affine transformation defined in Section 3. When amount calibration is disabled in ablations, we set

$$\tilde{\mathbf{x}}_k = \mathbf{u}_k.$$

Batching variable-size formulations. A formulation contains $M = n_s + 1$ valid component states. For batching, calibrated component states are scattered into a padded tensor

$$\tilde{\mathbf{X}}_{\text{comp}} \in \mathbb{R}^{B \times M_{\max} \times d},$$

where B is the batch size and $M_{\max}$ is the maximum component length in the batch or configured sequence length. A padding mask

$$\mathbf{P}_{\text{comp}} \in \{0, 1\}^{B \times M_{\max}}$$

marks invalid component positions. The mask is used in self-attention and pooling so that padded tokens do not contribute to the mixture state.

Interaction input sequence. Following Section 3, the interaction sequence is

$$\mathcal{S} = [\mathcal{B}_{\text{g}}, \tilde{\mathcal{X}}, \mathcal{C}_T], \qquad \tilde{\mathcal{X}} = \{\tilde{\mathbf{x}}_k\}_{k=1}^{n_s+1}.$$

For mean pooling and attention pooling,

$$\mathcal{B}_{\text{g}} = \emptyset.$$

For global-token pooling,

$$\mathcal{B}_{\text{g}} = \{\mathbf{x}^{\text{g}}\}, \qquad \mathbf{x}^{\text{g}} \in \mathbb{R}^d.$$

For late-concat temperature formatting,

$$\mathcal{C}_T = \emptyset.$$

For in-context temperature formatting, temperature is represented as

$$\psi_T(T_b) = [T_b/273.15,\ 273.15/T_b],$$

and embedded by a temperature encoder:

$$\mathbf{x}_b^T = \eta_T(\psi_T(T_b)), \qquad \eta_T : \mathbb{R}^2 \to \mathbb{R}^d.$$

Then

$$\mathcal{C}_T = \{\mathbf{x}_b^T\}.$$

The temperature token is always valid and is not masked. The salt mole ratio c is not inserted as a separate condition token in the main formulation; it enters through the component amount m_k of the salt token.

Token interaction module. The interaction module I_θ is implemented as a stack of L_t pre-normalized Transformer blocks. Let

$$\mathbf{X}^{(0)} \in \mathbb{R}^{B \times M_{\text{seq}} \times d}$$

be the tensor representation of $\mathcal{S}$, where M_{seq} includes component tokens and any optional global or temperature token. At layer ℓ, the attention branch is

$$\bar{\mathbf{X}}^{(\ell)} = \mathrm{LN}_{\text{attn}}\left(\mathbf{X}^{(\ell)}\right),$$

$$\mathbf{A}^{(\ell)} = \mathrm{MHA}\left(\bar{\mathbf{X}}^{(\ell)}, \bar{\mathbf{X}}^{(\ell)}, \bar{\mathbf{X}}^{(\ell)}; \mathbf{P}_{\text{seq}}\right),$$

where $\mathbf{P}_{\text{seq}}$ is the sequence padding mask. The residual attention update is

$$\mathbf{Y}^{(\ell)} = \mathbf{X}^{(\ell)} + \mathbf{A}^{(\ell)}.$$

The feed-forward branch is also pre-normalized:

$$\mathbf{X}^{(\ell+1)} = \mathbf{Y}^{(\ell)} + \mathrm{MLP}_{\text{ffn}}\left(\mathrm{LN}_{\text{ffn}}(\mathbf{Y}^{(\ell)})\right).$$

The feed-forward network uses expansion ratio 4:

$$\mathrm{MLP}_{\text{ffn}} : \mathbb{R}^d \to \mathbb{R}^{4d} \to \mathbb{R}^d,$$

with SiLU activation. The default implementation uses $L_t = 3$ Transformer blocks and 4 attention heads. The final contextualized token states are

$$\mathbf{X}^{(L_t)} = I_\theta(\mathcal{S}).$$

Mixture-state pooling. The pooling operator P_θ maps contextualized tokens to one mixture state $\mathbf{z}_b$ for each sample b. For mean and attention pooling, the readout set $\mathcal{R}_b$ contains the valid contextualized component tokens and excludes padding positions and auxiliary global or temperature tokens.

For mean pooling,

$$\mathbf{z}_b = \frac{1}{|\mathcal{R}_b|} \sum_{j \in \mathcal{R}_b} \mathbf{X}_{b,j}^{(L_t)}.$$

For attention pooling,

$$a_{b,j} = \frac{\exp(q_\theta(\mathbf{X}_{b,j}^{(L_t)}))}{\sum_{\ell \in \mathcal{R}_b} \exp(q_\theta(\mathbf{X}_{b,\ell}^{(L_t)}))},$$

$$\mathbf{z}_b = \sum_{j \in \mathcal{R}_b} a_{b,j} \mathbf{X}_{b,j}^{(L_t)},$$

where $q_\theta : \mathbb{R}^d \to \mathbb{R}$ is an MLP scoring function.

For global-token pooling, the mixture state is the contextualized global token. Writing the final global-token representation as $\mathbf{x}_b^{\mathrm{g},(L_t)}$, we set

$$\mathbf{z}_b = \mathbf{x}_b^{\mathrm{g},(L_t)}.$$

Projection and temperature-formatted readout. Before the final readout, the mixture state is projected by a learned linear layer:

$$\mathbf{p}_b = \mathbf{W}_p \mathbf{z}_b + \mathbf{b}_p.$$

In the late-concat regime, temperature is appended at readout:

$$\mathbf{q}_b = [\mathbf{p}_b \| \psi_T(T_b)], \qquad \hat{s}_b = h_\theta(\mathbf{q}_b).$$

In the in-context regime, temperature has already entered through $\mathbf{x}_b^T$, and the readout uses the projected mixture state:

$$\hat{s}_b = h_\theta(\mathbf{p}_b).$$

The readout h_θ is a normalized MLP. For the late-concat default, its input dimension is $d + 2$. The implemented structure is

$$\mathbf{r}_1 = \mathrm{Linear}(\mathbf{q}_b),$$

Table 11: Default SolvMix model and training configuration.

Category	Hyperparameter	Value
Architecture	Hidden dimension	64
Architecture	Graph encoder blocks	3
Architecture	Atom-level interaction blocks	0
Architecture	Token interaction blocks	3
Architecture	Attention heads	4
Architecture	Maximum component sequence length	16
Architecture	Dropout	0
Formulation-state construction	Component token initialization	amount-and-type-aware
Formulation-state construction	Amount calibration	multiplicative
Formulation-state construction	Mixture-state pooling	mean pooling
Formulation-state construction	Temperature formatting	late-concat
Optimization	Batch size	full batch
Optimization	Epochs	50,000
Optimization	Optimizer	AdamW
Optimization	Learning rate	5.0×10^{-4}
Optimization	Weight decay	1.0×10^{-4}
Optimization	Warmup steps	100
Optimization	Learning-rate schedule	exponential decay
Optimization	LR decay factor γ	0.99985
Optimization	Gradient clipping	0.5
Averaging	EMA	true
Averaging	EMA decay	0.999
Evaluation	Precision	FP32
Evaluation	Evaluation interval	10 epochs
Evaluation	Seed	42

$$\begin{aligned}
\mathbf{r}_2 &= \mathrm{LN}\left(\mathrm{SiLU}(\mathbf{r}_1)\right), \\
\mathbf{r}_3 &= \mathrm{MLP}_3(\mathbf{r}_2), \\
\mathbf{r}_4 &= \mathrm{LN}\left(\mathrm{SiLU}(\mathbf{r}_3)\right), \\
\hat{s}_b &= \mathrm{Softplus}\left(\mathrm{MLP}_4(\mathbf{r}_4)\right).
\end{aligned}$$

The Softplus output keeps the predicted log-conductivity score nonnegative and stabilizes inverse transformation to the original conductivity scale:

$$\hat{y}_b = \exp(\hat{s}_b) - 1.$$

B.2 Model hyperparameters and training protocol

Table 11 summarizes the default SolvMix configuration used in our experiments. Unless otherwise stated, all SolvMix results use this configuration together with the formulation-state choices selected in the main ablation study: amount-and-type-aware component tokens, multiplicative amount calibration, token interaction followed by mean pooling, and late-concat temperature formatting.

Full-batch training. We use full-batch training by setting `batch_size=null`. This is feasible for the preprocessed electrolyte benchmarks and removes mini-batch sampling noise when comparing architectural variants. It also makes validation curves easier to interpret, since performance changes are mainly driven by optimization dynamics rather than stochastic batch composition.

Learning-rate schedule and EMA. After warmup, the learning rate is decayed exponentially:

$$\eta_t = \eta_0 \gamma^t,$$

where $\eta_0 = 5.0 \times 10^{-4}$ and $\gamma = 0.99985$. In parallel, we maintain exponential moving average parameters:

$$\theta_t^{\mathrm{EMA}} = \rho\, \theta_{t-1}^{\mathrm{EMA}} + (1-\rho)\theta_t,$$

with $\rho = 0.999$. All validation and test results are reported using the EMA parameters at the final training step; we do not select checkpoints by validation performance.

Table 12: Design ablations on CALiSol-23 (seed 42), values plotted in Figure 4a. Each row changes one choice from the default. For reference, the five-seed std of the default model is 0.0152 (MAE) and 0.123 (MSE); differences smaller than this are not conclusive.

Axis	Variant	$R^2 \uparrow$	$\rho \uparrow$	MAE↓	MSE↓
—	SolvMix (default)	**0.9819**	**0.9919**	**0.1502**	**0.2727**
Tokens	identity-only	0.9797	0.9904	0.1745	0.3026
Calibration	additive	0.9568	0.9783	0.2714	0.6453
Calibration	FiLM-style	0.9699	0.9854	0.1894	0.4562
Pooling	attention	0.9714	0.9865	0.1723	0.4189
Pooling	global token	0.9798	0.9893	0.1563	0.3124
Temperature	in-context token	0.9586	0.9777	0.2378	0.6248

Log-space training and original-scale evaluation. For target conductivity $y \geq 0$, SolvMix predicts $\hat{s}$ and is trained against

$$s = \log(1 + y).$$

All validation and test metrics are computed after inverse transformation to the original conductivity scale in $\mathrm{mS/cm}$. This keeps optimization stable while preserving physically meaningful evaluation.

C Full Ablation Results

C.1 Formulation-state design ablations

C.2 Architecture and optimization exploration

In addition to the formulation-state ablations in Section 4.3, we explore lower-level architecture and optimization choices around the default SolvMix configuration (Table 13). These experiments are not intended as separate modeling claims. They test whether the main results depend on incidental implementation choices, including the graph encoder family, dropout, learning-rate decay, gradient clipping, atom-level interaction, and readout parameterization. Unless otherwise specified, all variants keep the same dataset split, formulation-state configuration, token-interaction module, readout family, optimizer, EMA setting, and evaluation protocol.

Graph encoder family. We compare the default edge-conditioned MLP message-passing encoder with common graph encoder families, including GCN, GAT, GIN, and GraphSAGE. These variants test whether SolvMix depends on a particular molecular encoder or mainly on the formulation-state construction above the encoder. The graph encoder maps each solvent or salt graph G_k to a molecular descriptor $\mathbf{x}_k$, while the subsequent SolvMix pipeline converts $\mathbf{x}_k$ into an amount-calibrated component state before token interaction. Thus, the encoder comparison separates molecular descriptor quality from the higher-level question of how descriptors become formulation-specific states.

Dropout. We evaluate dropout rates $p = 0.05$ and $p = 0.10$ against the default no-dropout configuration. SolvMix is compact and trained in a full-batch regime on curated conductivity datasets. In this setting, dropout can reduce the effective capacity of the component-state pipeline and inject noise into the token representations that must already encode subtle amount-dependent effects. We therefore use no dropout by default and report dropout variants as a check that the gains are not due to conventional regularization.

Learning-rate decay and gradient clipping. We vary the exponential learning-rate decay factor γ around the default schedule and also test a larger gradient clipping threshold. These choices directly affect late-stage optimization, which is important because SolvMix learns a smooth formulation-state landscape from macroscopic labels. If the learning rate decays too quickly, the model may stop refining the state geometry before convergence; if it decays too slowly, optimization may remain unsettled. Similarly, overly large gradient updates can perturb the amount-calibrated token states and destabilize the interaction module. The default setting is chosen to balance early progress with stable late-stage refinement.

Table 13: Architecture and optimization exploration around the default SolvMix configuration. All variants are evaluated under the same dataset split and metric protocol. The default model uses an edge-conditioned MLP graph encoder, no dropout, multiplicative amount calibration, token interaction with mean pooling, late-concat temperature formatting, and a LayerNorm-stabilized readout. For atom-level interaction, we insert the GeoMix GIL module before SolvMix token interaction.

Setting	Variant	MAE ↓	MSE ↓	R^2 ↑	Spearman ρ ↑
Default SolvMix	Edge-conditioned MLP encoder	0.1502	0.2727	0.9819	0.9919
Graph encoder	GCN	0.1586	0.2906	0.9809	0.9913
Graph encoder	GAT	0.1522	0.2916	0.9808	0.9914
Graph encoder	GIN	0.1558	0.3033	0.9801	0.9906
Graph encoder	GraphSAGE	0.1537	0.2945	0.9806	0.9915
Dropout	$p = 0.05$	0.1555	0.2942	0.9805	0.9907
Dropout	$p = 0.10$	0.1643	0.3152	0.9792	0.9895
LR decay γ	0.9997	0.1547	0.2977	0.9801	0.9898
LR decay γ	0.9998	0.1587	0.2938	0.9806	0.9904
LR decay γ	0.9999	0.1560	0.2834	0.9811	0.9911
Gradient clipping	1.0	0.1605	0.3110	0.9793	0.9895
Atom-level interaction	GIL, 3 layers, w/o denoising	0.1626	0.3239	0.9785	0.9883
Atom-level interaction	GIL, 3 layers, with denoising	0.1792	0.3833	0.9745	0.9850
Readout normalization	w/o LayerNorm	0.1700	0.3551	0.9765	0.9867
Readout output activation	SiLU	0.1685	0.3450	0.9770	0.9868
Readout output activation	none	0.1751	0.3632	0.9758	0.9856

Atom-level interaction. We test whether explicit atom-level cross-component interaction improves SolvMix. Specifically, we insert the GeoMix GIL module before SolvMix token interaction. GIL explicitly models inter-molecular atom-level message passing and, in GeoMix, is paired with a BERT-like denoising objective that predicts masked atomic coordinates. We evaluate two 3-layer GIL variants: one without denoising and one with the denoising objective. The 3-layer setting follows the default GeoMix-style configuration. In our architecture, GIL is placed before component-token interaction: atom states are first updated through GIL, then pooled into component descriptors, converted into amount-calibrated component states, and finally passed to the SolvMix token-interaction module.

This comparison tests a key modeling distinction. GeoMix emphasizes explicit atom-level interaction before mixture prediction, whereas SolvMix emphasizes formulation-state construction at the component-token level. Under macroscopic conductivity supervision, atom-level interaction signals are only weakly identified unless additional structural supervision is informative. Therefore, GIL with and without denoising tests whether adding explicit atom-level interaction provides complementary information beyond amount-calibrated token interaction, or whether it introduces extra degrees of freedom that make optimization less stable.

Readout design. We evaluate the contribution of the readout parameterization by removing LayerNorm and by replacing the default Softplus output with either SiLU or no final activation. The readout maps the temperature-formatted mixture state to the predicted log-conductivity score $\hat{s}$. LayerNorm controls the scale of the mixture representation before the final nonlinear mapping. The Softplus output enforces a nonnegative $\hat{s}$, which is consistent with the inverse transformation $\hat{y} = \exp(\hat{s}) - 1$ and avoids invalid negative conductivity predictions on the original scale. The SiLU and no-activation variants test whether this positivity constraint and readout normalization are important for stable original-scale evaluation.

Summary. No encoder, optimization, or readout change improves on the default, and the largest change in MSE is $+0.11$ (atom-level GIL with denoising). Changing the graph encoder family moves MSE by at most 0.03, well within seed variation.

Table 14: Multi-seed stability of SolvMix on the three electrolyte conductivity benchmarks. Each result is reported as mean $\pm$ standard deviation over 5 seeds: $42, 66, 101, 2026, 8888$. All metrics are computed on the test set in the original conductivity scale.

Dataset	Spearman $\rho \uparrow$	MAE $\downarrow$	MSE $\downarrow$
CALiSol-23	0.9893 ± 0.0038	0.1660 ± 0.0152	0.3037 ± 0.1230
DiffMix	$0.99998 \pm 7.92\text{e-}6$	$7.75\text{e-}3 \pm 2.59\text{e-}3$	$2.07\text{e-}4 \pm 1.02\text{e-}4$
Bamboo-mixer	0.9990	0.0801 ± 0.0070	0.1019 ± 0.0410

C.3 Multi-seed stability

To assess whether the reported gains are robust to random initialization and optimization noise, we repeat the default SolvMix experiment with five random seeds:

$$\{42, 66, 101, 2026, 8888\}.$$

For each seed, we keep the same dataset preprocessing, model configuration, training protocol, and evaluation split, and only change the random seed used for parameter initialization and training. We report the mean and standard deviation across seeds on the test set (Table 14). All reported metrics are computed after inverse transformation to the original conductivity scale in $\mathrm{mS/cm}$.

The seed-42 run used for the ablations is better than the five-seed mean on CALiSol-23 (MSE 0.273 vs. 0.304) and Bamboo-mixer (MSE 0.045 vs. 0.102), so the main results in Table 2 report the mean. The standard deviation of MSE on CALiSol-23 (0.123) is larger than several ablation gaps in Table 12, and we treat those gaps as inconclusive. SolvMix's five-seed mean is better than every single-run baseline on all three datasets.

D Failure cases on CALiSol-23

CALiSol-23 is the noisiest benchmark in our study and exhibits the clearest failure cases. Compared with DiffMix and Bamboo-mixer, models trained on CALiSol-23 show more rugged optimization trajectories, especially near convergence. This behavior is not unique to SolvMix: several baselines also converge less smoothly on CALiSol-23, suggesting that the difficulty is partly driven by the supervision itself rather than by model design alone.

We focus on $LiBF_4$ formulations involving PC and DEC. Here PC denotes propylene carbonate with SMILES `CC1COC(=O)O1`, and DEC denotes diethyl carbonate with SMILES `CCOC(=O)OCC`. These solvents are common in lithium-electrolyte formulations, but in CALiSol-23 the corresponding response surfaces can be highly irregular after unit normalization.

Figure 7 illustrates this issue. Panel (a) shows the learned conductivity surface for the PC–$LiBF_4$ system trained on CALiSol-23, where the two varying coordinates are temperature and salt mole ratio. Panel (b) shows the same PC–$LiBF_4$ coordinate view for a model trained on Bamboo-mixer. Panel (c) shows the CALiSol-23 surface for the PC–DEC–$LiBF_4$ system, where salt mole ratio is fixed and the two varying coordinates are temperature and PC fraction. Blue points denote training samples and red points denote test samples. The contrast is clear: the CALiSol-23 surfaces in panels (a) and (c) contain sharp local fluctuations, whereas the Bamboo-mixer surface in panel (b) is much smoother.

The most plausible cause is not only ordinary experimental noise, but also systematic uncertainty introduced during concentration normalization. CALiSol-23 contains salt amounts reported under mixed unit conventions, including $\mathrm{mol/L}$ and $\mathrm{mol/kg}$. To build a unified formulation representation, these values must be converted into the salt mole ratio c. For molality this is exact given the solvent composition and molar masses, but for molarity it requires the solution density, which depends on temperature, solvent composition, salt identity, and salt amount. For mixed-solvent systems, the density surface can itself be highly nonlinear and is rarely available for every formulation. As a result, the normalized coordinate c can inherit non-negligible conversion error.

This issue is especially severe in $LiBF_4$–PC-containing systems. Nearby points in the normalized formulation space may have conductivity labels affected by conversion artifacts, source heterogeneity, or measurement inconsistency, rather than by a smooth underlying physical response. Because SolvMix uses c directly in component-token construction and amount calibration, noise in the salt-

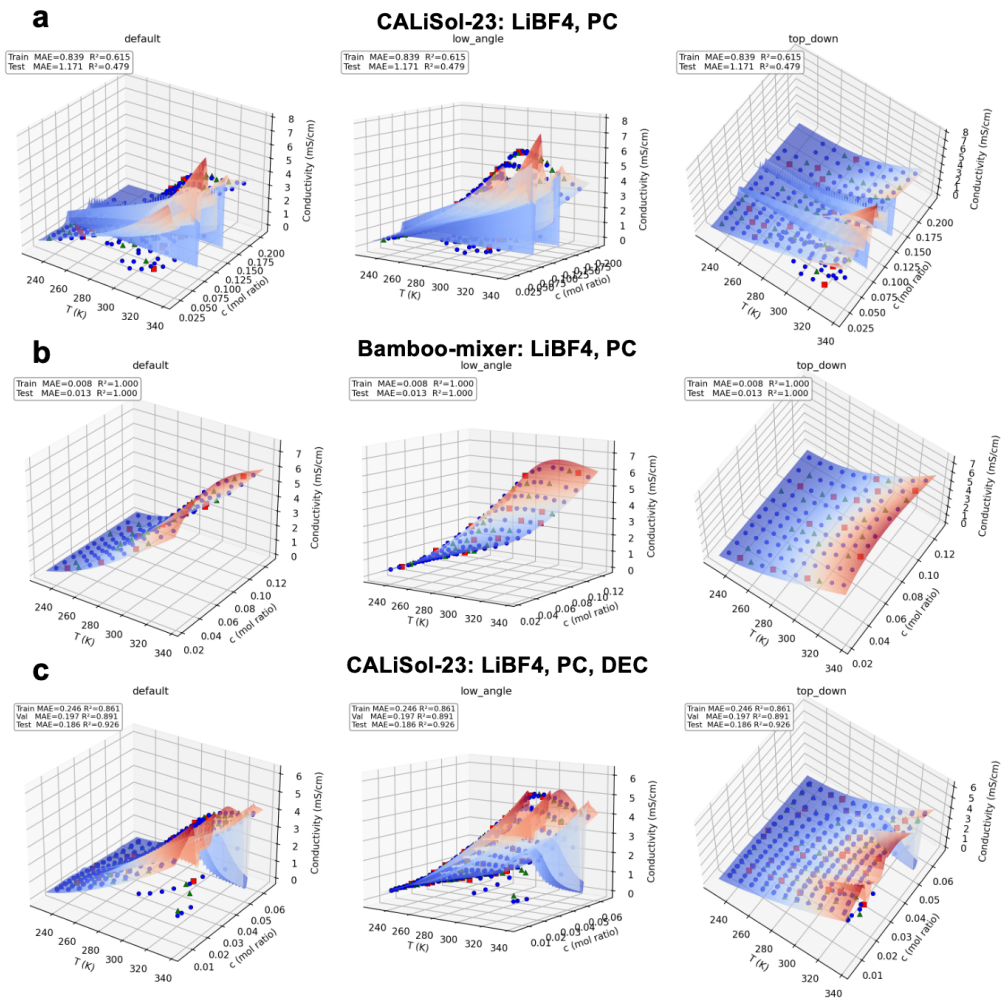


Figure 7: Failure cases induced by noisy concentration normalization in CALiSol-23. We visualize learned conductivity response surfaces for $LiBF_4$-containing formulations involving PC-based solvents. PC denotes propylene carbonate with SMILES `CC1COC(=O)O1`; DEC denotes diethyl carbonate with SMILES `CCOC(=O)OCC`. **(a)** CALiSol-23 surface for the PC–$LiBF_4$ system, with temperature and salt mole ratio as the two varying coordinates. **(b)** Bamboo-mixer surface for the same PC–$LiBF_4$ system under the same coordinate view. **(c)** CALiSol-23 surface for the PC–DEC–$LiBF_4$ system, with salt mole ratio fixed and temperature and PC fraction varied. Blue points denote training samples and red points denote test samples. CALiSol-23 surfaces in **(a)** and **(c)** are visibly more rugged than the Bamboo-mixer surface in **(b)**, suggesting that local irregularities are driven by dataset-specific noise rather than by the model architecture alone. Possible sources include the conversion of $\mathrm{mol/L}$ values into salt mole ratio, which requires a temperature- and composition-dependent density, and heterogeneity between literature sources. Conductivity is reported in $\mathrm{mS/cm}$.

ratio coordinate propagates into the learned formulation-state landscape. The resulting surfaces can therefore appear artificially rugged even when the underlying physical trend is expected to be smoother.

We therefore treat CALiSol-23 as an important but noisy real-world benchmark. Its value lies in testing robustness under imperfect experimental curation, but its response landscapes should not be interpreted as clean ground-truth physical surfaces. This is also why we complement CALiSol-23 with Bamboo-mixer and DiffMix: together, the three benchmarks help separate genuine formulation-state modeling ability from artifacts caused by dataset-specific noise.

E Additional discussion

E.1 Fine-grained test-set performance across solvent-count and temperature strata

To examine whether SolvMix exhibits systematic prediction bias across formulation complexity or operating conditions, we perform a fine-grained analysis on the Bamboo-mixer test set. Specifically, we group the test samples by the number of solvent components and by temperature quantile bins, and then evaluate the model separately within each subgroup. This analysis complements the aggregate benchmark results by testing whether the strong overall performance is uniformly maintained across different formulation regimes, or instead driven by a subset of easier cases.

Figure 8 summarizes the results. The left panel groups test samples by the number of solvents in the formulation, while the right panel groups them by five temperature quantile bins. For each grouping, we report six regression metrics: R^2, Pearson r, Spearman ρ, MAE, RMSE, and MSE. Bar annotations additionally show the exact metric value and subgroup sample size. This figure is intended as a diagnostic view of performance stratification rather than as a new benchmark protocol.

Several patterns are clear. First, SolvMix maintains consistently strong predictive accuracy across nearly all solvent-count strata. For formulations containing one to four solvents, the model achieves near-perfect performance, with $R^2 > 0.99$ and similarly strong rank and correlation metrics. The error metrics remain low across these groups, indicating that the model does not exhibit an obvious degradation as formulation complexity increases from unary to quaternary solvent systems. The only visible drop appears in the five-solvent subgroup. This subgroup has only three test samples, so its metrics are not reliable.

Second, the temperature-stratified analysis also shows that SolvMix remains highly accurate across most operating ranges. Performance is stable across the majority of temperature bins, again with R^2 close to 1 and low absolute and squared errors. A modest degradation appears in the 253–283 K interval, which corresponds to the highest RMSE and MSE among the temperature bins.

Errors are therefore spread across solvent counts and temperatures; the two weaker strata are a five-solvent group with only three samples and the 253–283 K bin.

E.2 Interpreting the extremely low DiffMix error

The extremely low DiffMix error should be interpreted together with the construction of the DiffMix benchmark. Unlike CALiSol-23, which aggregates experimental measurements from heterogeneous literature sources, DiffMix is a simulation-driven and systematically discretized dataset. The original DiffMix framework combines geometric deep learning with mixture physics laws, including a VFT-type transport model for ionic conductivity, and learns physical coefficients from molecular graphs, compositions, and environmental conditions [Zhu et al., 2024]. The curated DiffMix benchmark used in GeoMix contains 24,822 formulations but only two salts, $LiPF_6$ and LiTFSI, and up to three solvents selected from six carbonate solvents, with temperatures restricted to 243.15–293.15 K. GeoMix further describes the dataset as a systematic discretization with quantized salt concentrations and coarsely sampled solvent fractions [Li et al., 2025].

Therefore, DiffMix is large in sample count but narrow in chemical and thermodynamic diversity. Its labels are also expected to be more internally consistent than literature-aggregated experimental data, because they are generated from a single simulation-driven pipeline rather than from many laboratories, protocols, instruments, and unit conventions. This combination produces a smooth, densely sampled, low-noise formulation manifold.

SolvMix is particularly well matched to this regime. Its architecture learns how molecular identities become amount-calibrated component states, how these states interact, and how temperature formats the final conductivity response. On DiffMix, where the response surface is regular and densely sampled, this formulation-state bias allows the model to nearly recover the underlying conductivity landscape. Thus, the extremely low MSE should not be interpreted as evidence that conductivity prediction is solved in full generality. It is better understood as a positive-control result: under clean, narrow, and systematic data, SolvMix can interpolate the formulation-state landscape with very high precision.

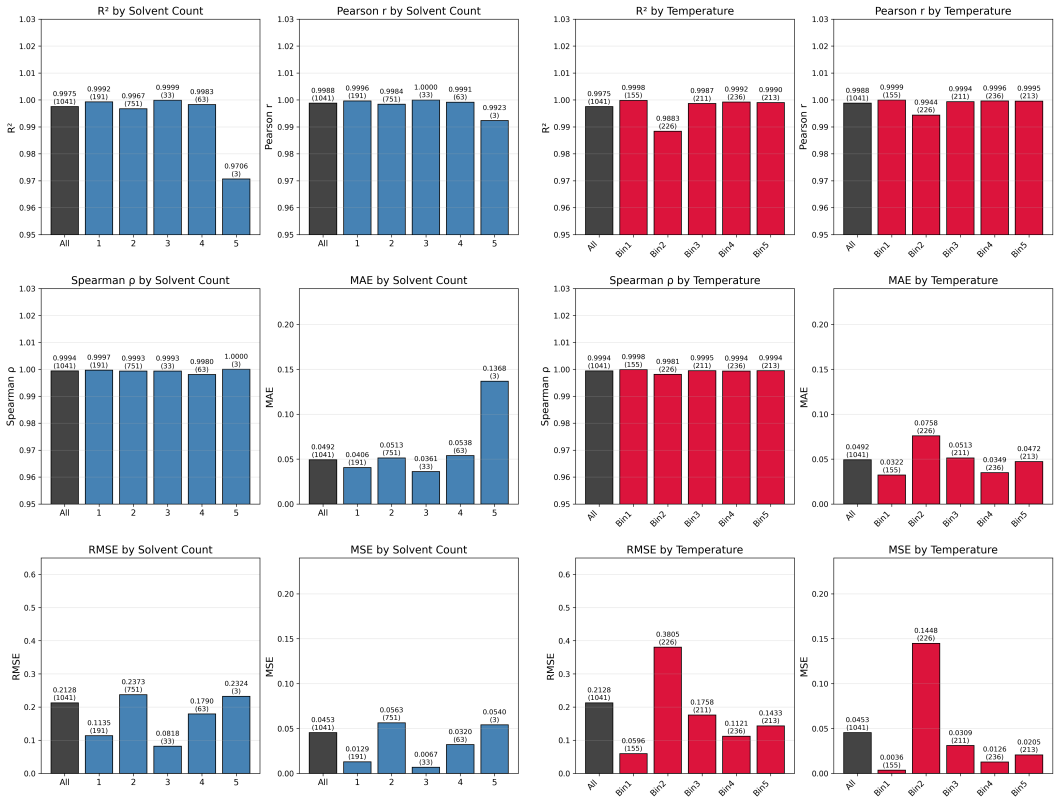


Figure 8: Fine-grained test-set performance of SolvMix on Bamboo-mixer ($n = 1{,}041$). The left panel groups samples by the number of solvent components in the formulation, and the right panel groups samples by temperature quantile bins. Each subplot reports one regression metric: R^2, Pearson r, Spearman ρ, MAE, RMSE, or MSE. Values and sample sizes are annotated above the bars. SolvMix maintains near-perfect predictive accuracy across most strata, with only mild degradation in the extreme low-data five-solvent subgroup ($n = 3$) and in the 253–283 K temperature interval, which shows the largest RMSE and MSE.

DiffMix results should therefore be read as a check that SolvMix can fit a clean, densely sampled response surface; the experimental datasets CALiSol-23 and Bamboo-mixer carry the main conclusions.

E.3 Relation to physical and geometric mixture models

DiffMix places composition and temperature inside a parameterized transport law and learns its coefficients from molecular graphs. GeoMix couples components through atom-level geometric interaction and receives formulation variables as auxiliary inputs. SolvMix instead lets the amount of each component transform that component's representation before any interaction:

molecular descriptor $\rightarrow$ amount-calibrated component state $\rightarrow$ token interaction $\rightarrow$ readout with T.

Our ablations compare where variables enter within SolvMix, but we did not run the complementary control of giving GeoMix the same amount and temperature features (e.g., $\psi_T(T)$ and per-component amount embeddings) without amount calibration. That experiment would separate the effect of better feature design from the effect of the calibration step itself, and we leave it to future work. Within SolvMix, the additive-calibration and in-context-temperature variants receive exactly the same inputs as the default model yet are considerably worse (Table 12), so input features alone do not account for the gain.

E.4 Sensitivity to formulation-coordinate noise

Because solvent ratios and the salt mole ratio scale each component state before interaction, errors in these coordinates propagate directly into the mixture state. This is the mechanism that helps on clean held-out regions (Section 4.4), and it is also why SolvMix produces rugged response surfaces on CALiSol-23, where salt amounts reported in mol/L were converted using approximate densities (Appendix D). Three extensions could reduce this sensitivity without removing the dependence on composition: (i) training with small perturbations of the coordinates, e.g. $c' = c + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_c^2)$, restricted to physically valid values; (ii) treating converted salt amounts as intervals rather than point values when the source unit is ambiguous; and (iii) performing unit conversion with density models instead of fixed densities.

F Extended limitations

Effective states are not microscopic observables. The learned formulation states are predictive effective states, not directly observed physical variables. They are supervised only through macroscopic conductivity labels and therefore should not be interpreted as explicit solvation structures, ion-pairing states, viscosity effects, or transport pathways without additional physical validation. This is a fundamental ambiguity of learning from endpoint properties: different microscopic mechanisms can produce similar conductivity responses.

OOD generalization remains intrinsically difficult. Although SolvMix improves over strong baselines under binned OOD splits, OOD prediction remains substantially harder than random-split interpolation. This reflects a broader limitation of data-driven models in sparse scientific landscapes. Neural networks are strongest at interpolating among observed regimes, while reliable extrapolation requires enough boundary and off-manifold observations to anchor the response surface. In electrolyte formulation space, a small number of extreme or held-out measurements can strongly influence the inferred shape of the conductivity landscape, especially along coupled composition–salt–temperature directions.

Coupled physical effects are only indirectly modeled. SolvMix places solvent ratios, salt mole ratio, and temperature at specific stages of the state-construction pipeline, but the best way to model their coupled physical effects remains open. Conductivity is governed by ion dissociation, solvation structure, viscosity, dielectric environment, and collective transport. A macroscopic conductivity label alone cannot uniquely identify these mechanisms. Future extensions may benefit from auxiliary supervision, physically informed constraints, or multimodal measurements that expose more of the underlying transport process.

Benchmark coverage bounds the demonstrated generalization. The current results are limited by the chemical and thermodynamic coverage of CALiSol-23, DiffMix, and Bamboo-mixer. Extrapolation to new solvent families, new salt chemistries, additives, high-concentration regimes, or electrolyte systems beyond the observed lithium-ion formulation space may require broader data and stronger priors. This is especially important for formulation discovery, where the most valuable candidates may lie near the edge of the measured landscape.

Prediction is not yet closed-loop design. SolvMix is a predictive model, not a complete electrolyte discovery system. Practical deployment would require uncertainty-aware acquisition, synthesis and formulation constraints, stability and safety filters, cost and manufacturability considerations, and experimental validation. Thus, formulation-state learning should be viewed as a core predictive component that can support, but not replace, a closed-loop scientific design pipeline.

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: Our claims in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines: Our claims accurately reflect the paper's contributions and scope.

   - 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: Limitations are discussed in Section 7, with an extended discussion in Appendix F.

   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: [Yes]

Justification: Proposition 1 in Section 2 states its assumptions (the token and calibration forms of Eq. (2)) and is followed by a complete proof.

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: We provide all the information needed to reproduce the main experimental results of the paper.

   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: [No]

Justification: Code release is restricted by confidentiality obligations and intellectual property policies of our organization. All implementation details, including the network architecture, optimization settings, and data processing pipeline, are thoroughly documented in the main text and supplemental material.

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: The paper specifies all the training and test details.

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: SolvMix results in Table 2 are mean $\pm$ one standard deviation over five random seeds (initialization and training; fixed split), described in Appendix C.3. Baselines and ablations are single runs; ablation captions give the seed (42), and Section 7 states this as a limitation.

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: [No]

   Justification: The per-run computational cost is low.

   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: We respect.

   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: Our work is for new material design.

    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 poses no such risks.

    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 respect.

    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: [N/A]

    Justification: No new released assets.

    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: No such risks.

    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: No such risks.

    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: [N/A]

    Justification: LLM is not the core method development in this research.

    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.