SLOT-IR: Learning Slot Representations for Infrared Spectral Unmixing

Jingru Gan Yannah J.U. Melle Yanqiao Zhu Daniel Schwalbe-Koda Wei Wang
University of California, Los Angeles
{jrgan,yzhu,weiwang}@cs.ucla.edu, {yannah,dskoda}@ucla.edu

Abstract

Infrared (IR) spectral unmixing recovers individual component spectra from a measured mixture, enabling chemical identification. Existing methods either assume linear superposition of components, which does not hold for liquid-phase mixtures where molecular interactions alter spectral shapes, or use deep networks that encode the whole mixture into one latent vector without separate per-component representations. We introduce SLOT-IR, a slot-based architecture that decomposes mixture spectra into separate per-component embeddings and decodes each independently into a predicted spectrum. A two-stage training procedure first pretrains the encoder on single-molecule spectra, then trains the full unmixing model on mixtures. On a benchmark of liquid-phase infrared spectra computed with molecular dynamics simulations, SLOT-IR outperforms all baselines on both binary and ternary mixture identification. Controlled pretraining experiments suggest that the benefit comes from learning coherent spectra of individual molecules. Pretraining on gas-phase spectra alone performs comparably to pairing gas- and liquid-phase spectra, which points to a scalable source of pretraining data. A post-hoc sparse autoencoder further shows that the per-component embeddings carry information statistically associated with known functional groups.

1 Introduction

Spectroscopic characterization methods such as infrared (IR), nuclear magnetic resonance (NMR), or Raman spectroscopy are widely used for chemical identification (Pavia et al., 2014). Despite their ubiquity, a central challenge in spectroscopy is spectral unmixing: many real-world samples are liquid-phase mixtures, and their measured spectra are not simply superpositions of the constituent components. Thus, recovering individual component spectra from a mixture measurement is essential for chemical identification and quantification. However, the problem is challenging even when the identities of the components are known, and ill-defined when they are unknown.

In gas-phase measurements, mixture spectra are approximately linear combinations of individual component spectra. However, in liquid-phase samples, molecules interact through hydrogen bonding, dipole coupling, and solvation effects. These interactions shift peak positions, broaden absorption bands, and introduce features absent from isolated-molecule spectra. Methods that assume linear mixing cannot capture these changes. Recent deep learning methods such as DeepMIR (Tan et al., 2025) predict whether a given reference spectrum is present in a mixture. To our knowledge, no existing deep learning method reconstructs the individual component spectra of a liquid-phase IR mixture from the mixture spectrum alone.

Classical spectral unmixing methods can be divided into two groups. Library-based methods (Bioucas-Dias and Figueiredo, 2010; Lawson and Hanson, 1974) solve for the weights that best reconstruct the observed mixture as a weighted sum of pure spectra from a reference library. These methods are interpretable, as the weights directly indicate which compounds are present and in what proportions.

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

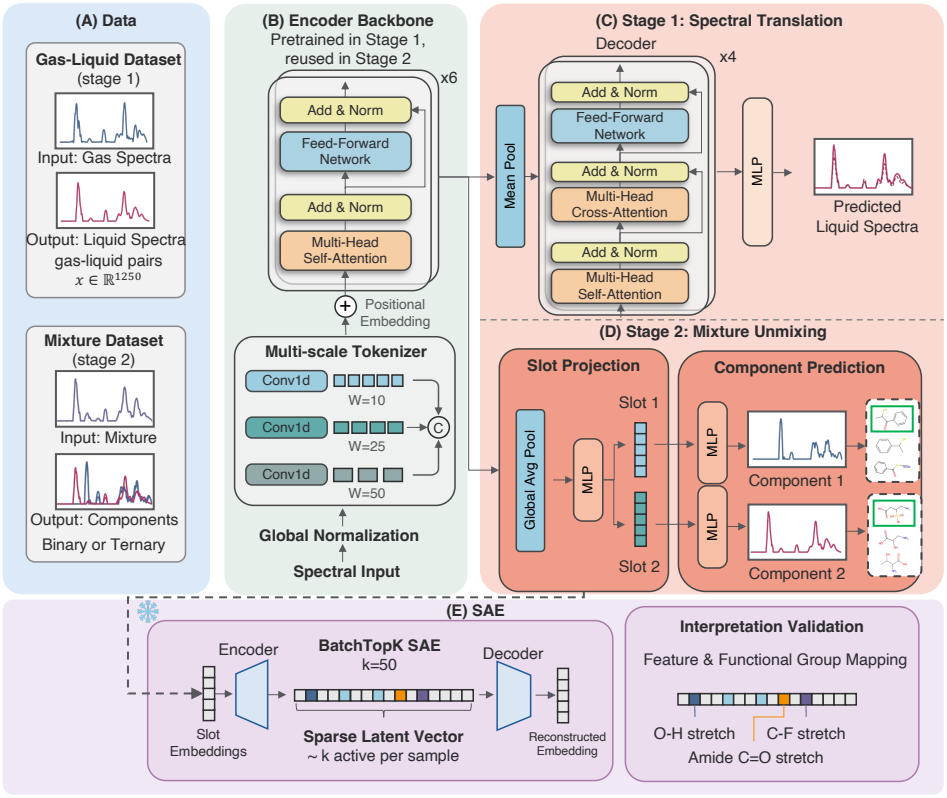


Figure 1: **SLOT-IR architecture.** SLOT-IR pretrains on gas-liquid pairs for spectral translation and then trains on the mixture dataset for unmixing (A). The model tokenizes a mixture spectrum at multiple scales, and encodes the tokens with a Transformer encoder (B). In Stage 1, a Transformer decoder predicts the liquid-phase spectrum (C). In Stage 2, the pooled encoder output is projected into K component-specific slot embeddings, each decoded into a predicted component spectrum (D). A BatchTopK sparse autoencoder (SAE) (E) decomposes frozen slot embeddings into sparse latent features.

However, they cannot model nonlinear spectral interactions under the linear mixing assumption. More importantly, they can only identify compounds already in the library, rather than reconstruct the component spectra in the mixture. Other classical unmixing methods (Nascimento and Bioucas-Dias, 2005; Tauler et al., 1993) avoid the need for a reference library by factorizing a set of mixture spectra into component spectra shared across them. The decompositions are obtained through iterative matrix factorization, which is sensitive to initialization. These methods also assume linear mixing and require a separate matching step to identify the extracted components.

Deep learning unmixing methods (Gao et al., 2022; Ghosh et al., 2022; Palsson et al., 2021) were developed for hyperspectral imaging and pair a nonlinear encoder with a linear mixing decoder. Applied to IR mixtures, they compress the mixture spectrum into a single latent vector from which all component spectra are decoded. Without per-component representations, what the model learns about each component cannot be examined separately.

In this paper, we introduce SLOT-IR, a slot-based architecture that combines nonlinear modeling with per-component representations. The model includes a Transformer encoder that processes the mixture spectrum, and a slot projection that decomposes its output into K separate embeddings for each component (Figure 1). Because each component has its own slot embedding, the slots can also be inspected post hoc. We study the setting where the number of components K is known: given a mixture spectrum and K, the model reconstructs K component spectra without access to the candidate molecules, and a reference library is used only afterwards to identify each reconstructed spectrum. We focus on component identification and do not estimate component proportions.

Because mixture spectra are costly to simulate, mixture training data cover only 213 molecules in our binary set, whereas single-molecule spectra are available for 8,387 molecules. We therefore adopt a two-stage training procedure. The encoder is first pretrained to translate gas-phase spectra into liquid-phase spectra of the same molecules, a task chosen to expose the encoder to the peak shifts and band broadening induced by liquid-phase interactions. The full model is then trained end-to-end on mixtures to learn the decomposition. Controlled experiments on the pretraining objective (Section 4.3) clarify what the pretraining stage provides. Removing pretraining makes optimization unreliable, and pretraining on shuffled gas-liquid pairs fails completely, suggesting that the encoder needs to learn coherent spectra of individual molecules. We further show that pretraining remains effective even without liquid-phase data, as a gas-phase autoencoder performs comparably to gas-to-liquid translation. Since gas-phase spectra require neither condensed-phase simulation nor liquid-phase measurement, this may offer a practical path to scaling pretraining to broader chemical spaces.

As a post-hoc check, we apply a BatchTopK Sparse Autoencoder (SAE) (Bricken et al., 2023; Cunningham et al., 2023) to the frozen slot embeddings and test whether its features are associated with known functional groups (FGs), molecular substructures that produce characteristic IR absorptions (Silverstein et al., 2014).

We evaluate on a benchmark of IR spectra computed with molecular dynamics simulations, spanning binary and ternary liquid-phase mixtures (Melle et al., 2026). SLOT-IR achieves 81.9% and 94.3% top-1 per-component identification accuracy on binary and ternary mixtures respectively, outperforming all baselines. Ablations in Section 4.3 support the two-stage training design and show that gas-phase pretraining alone performs comparably. We also study generalization in Section 4.4, from molecules held out of mixture training to drug molecules never seen in any training stage. The SAE analysis finds features statistically associated with functional groups without any chemical supervision, with clear alignment with IR bands for some groups and gaps for others.

2 Related Work

Spectral unmixing. Classical spectral unmixing methods decompose observed mixture spectra into pure component signatures. Multivariate curve resolution by alternating least squares (MCR-ALS) (Tauler et al., 1993) iterates under non-negativity constraints but assumes strict linearity and is sensitive to initialization. Vertex component analysis (VCA) (Nascimento and Bioucas-Dias, 2005) identifies endmembers as simplex vertices in reduced-dimensional space. When a reference library is available, sparse regression methods such as sparse unmixing by variable splitting and augmented Lagrangian (SUnSAL) (Bioucas-Dias and Figueiredo, 2010) formulate unmixing as ℓ_1-regularized non-negative regression. Deep learning approaches, including CNN autoencoders (Palsson et al., 2021) and Transformer-based methods (Ghosh et al., 2022), use nonlinear encoders with a linear mixing decoder and decode all components from one shared latent vector. These deep learning methods were developed for hyperspectral remote sensing rather than molecular IR spectra, and we adapt them as baselines (Section 4.1). Georgiev et al. (2024) develop physics-constrained autoencoders for Raman hyperspectral unmixing, which require a collection of spatially resolved spectra rather than a single mixture measurement. Recent work by Melle et al. (2026) demonstrates algorithmic identification of compounds in liquid-phase IR mixtures, achieving up to 90% accuracy. Our approach differs by learning to reconstruct component spectra rather than directly predicting identity, so the reference library is needed only for the final identification step.

Deep learning for molecular-spectroscopy mixtures. Given a mixture spectrum and a candidate reference spectrum, DeepMIR (Tan et al., 2025) predicts whether the candidate is present in the mid-IR mixture. DeepRaman (Fan et al., 2023) applies a pseudo-Siamese network to the analogous Raman problem. Both classify each library candidate as present or absent. SLOT-IR instead takes only the mixture spectrum and reconstructs the K component spectra themselves. The library is used only afterwards to identify each reconstructed spectrum. The per-component slot embeddings can also be analyzed post hoc. To our knowledge, no prior deep learning method performs mixture-only component reconstruction for nonlinear liquid-phase IR mixtures.

Slot-based decomposition. Slot Attention (Locatello et al., 2020) introduced iterative competitive attention for unsupervised object discovery, where slots initialized from a learned distribution compete through attention to bind to distinct objects. Extensions include video understanding (Kipf et al., 2022), probabilistic variants with identifiability guarantees (Kori et al., 2024), and diffusion-based

generation (Wu et al., 2023). DETR (Carion et al., 2020) demonstrated Hungarian matching for permutation-invariant set prediction. Slot-based models have mainly been applied to visual data. We adapt the paradigm to 1D spectral decomposition with a simplified feed-forward slot projection, where slots correspond to mixture components and are supervised via Hungarian matching with ground-truth component spectra.

Sparse autoencoders for interpretability. SAEs decompose dense neural activations into sparse features, ideally each tied to one concept (Bricken et al., 2023; Cunningham et al., 2023). Variants include TopK SAEs with fixed per-sample sparsity (Gao et al., 2024), BatchTopK SAEs (Bussmann et al., 2024) with batch-adaptive sparsity budgets, and Gated SAEs (Rajamanoharan et al., 2024) with learned feature selection. Temporal SAEs (Bhalla et al., 2025) extend dictionary learning to sequential data by maintaining feature coherence across positions. Applications to large language models have revealed features corresponding to code syntax, factual knowledge, and linguistic patterns (Templeton et al., 2024), with automated explanation frameworks (Han et al., 2026) beginning to replace manual inspection. A persistent limitation is that validation relies on subjective human judgment: researchers inspect maximally activating examples and propose interpretations, but ground truth is unavailable (Templeton et al., 2024). We take a step toward this by applying BatchTopK SAEs to a domain with external chemical labels.

Interpretability validation. Concept Bottleneck Models (Koh et al., 2020) route predictions through pre-defined concept labels but require explicit concept supervision during training. Network dissection (Bau et al., 2017) aligns neurons with visual concepts and was studied on image classification networks. We evaluate post-hoc SAE features against external chemical annotations, without affecting model performance.

3 Method

3.1 Problem Setup

Let $\mathbf{x}_{\text{mix}} \in \mathbb{R}^L$ denote an observed mixture spectrum sampled at $L = 1{,}250$ evenly spaced wavenumber points covering 400 to 4000 cm^{-1}. The spectral unmixing task recovers pure component spectra $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$, $\mathbf{c}_i \in \mathbb{R}^L$, from the observed mixture. Since the components of a mixture are order-invariant, we handle permutation invariance via Hungarian matching (Carion et al., 2020), which finds the optimal one-to-one assignment between predicted and ground-truth components. Identification is performed by comparing each predicted spectrum against a reference library via cosine similarity, producing a ranked list of candidate molecules.

Each molecule is annotated with $F = 48$ functional group (FG) labels $\mathbf{f} \in \mathbb{N}^F$, the number of matches of each group's SMARTS substructure pattern in its molecular structure notation (SMILES). These annotations serve exclusively as external labels for the post-hoc SAE analysis and are never used during model training.

3.2 SLOT-IR

As shown in Figure 1, SLOT-IR takes a mixture spectrum and produces K component-specific slot embeddings, each decoded into a predicted component spectrum. The architecture consists of a multi-scale tokenizer, a Transformer encoder, a slot projection head, and a component prediction head. Training proceeds in two stages: spectral translation followed by mixture unmixing.

IR spectra contain features at multiple scales including narrow peaks and broad bands. To capture the multi-scale features, the tokenizer applies three parallel 1D convolution branches with kernel widths $w \in \{10, 25, 50\}$, each mapping the raw spectrum to d=512 channels. Each branch output is average-pooled with stride equal to its kernel width, giving 125, 50, and 25 tokens respectively. The three token sets are concatenated along the sequence dimension into $\mathbf{T} \in \mathbb{R}^{N_T \times d}$ with $N_T = 200$. One learned positional embedding per position is added after concatenation, so it encodes both the branch identity and the spectral position within the branch. A Transformer encoder processes this sequence into contextualized representations $\mathbf{M} \in \mathbb{R}^{N_T \times d}$.

Stage 1: Spectral Translation. Stage 1 trains the tokenizer and encoder on a gas-to-liquid spectral translation task. For each pair of gas and liquid spectra, the tokenizer and encoder process the gas-phase spectrum, and a Transformer decoder with cross-attention to the encoder output predicts

the corresponding liquid-phase spectrum. The training loss combines mean-squared error (MSE) and cosine similarity between predicted and ground-truth liquid spectra.

Gas and liquid IR spectra of the same molecule share most vibrational modes but differ in band shape and position because of intermolecular interactions such as hydrogen bonding. We refer to a liquid-phase spectrum as that of a pure liquid, in which the molecule interacts only with copies of itself and no solvent is present, so the gas-to-liquid mapping is defined for each molecule.

Our original motivation was that learning this mapping would expose the encoder to interaction-induced spectral changes before it sees mixtures. Section 4.3 suggests that the benefit comes instead from learning coherent spectra of individual molecules: pretraining on gas-phase spectra alone performs comparably.

Stage 2: Mixture Unmixing. Stage 2 initializes the tokenizer and encoder from Stage 1, adds the slot projection and prediction head, and trains all parameters end-to-end on mixture data. Given a mixture spectrum, the tokenizer and encoder produce $\mathbf{M}$. The slot projection globally average-pools the encoder output and projects it into K slot embeddings:

$$\bar{\mathbf{m}} = \frac{1}{N_T}\sum_{t=1}^{N_T}\mathbf{M}_t, \quad \mathbf{S} = \text{reshape}(\text{MLP}(\bar{\mathbf{m}}),\ [K, d]) \tag{1}$$

where the multi-layer perceptron (MLP) maps $\mathbb{R}^d$ to $\mathbb{R}^{Kd}$, and reshape partitions the output into K vectors of dimension d. We use a single feed-forward projection instead of the iterative cross-attention of Slot Attention (Locatello et al., 2020). The prediction head decodes each slot embedding $\mathbf{s}_i \in \mathbb{R}^d$ into a predicted component spectrum via another MLP:

$$\hat{\mathbf{c}}_i = \text{ReLU}(\text{MLP}_{\text{pred}}(\mathbf{s}_i)) \in \mathbb{R}^L_{\geq 0} \tag{2}$$

The final ReLU enforces non-negativity since IR absorbance values cannot be negative.

Predicted and ground-truth components are matched via Hungarian assignment that minimizes pairwise L2 distance. Let σ^* denote the optimal permutation. The training loss is:

$$\mathcal{L} = \frac{1}{KL}\sum_i \|\hat{\mathbf{c}}_i - \mathbf{c}_{\sigma^*(i)}\|_2^2 \tag{3}$$

4 Experiments

4.1 Setup

Dataset. All spectra in the benchmark (Melle et al., 2026) are computed with all-atom molecular dynamics (MD) simulations, sampled at L=1,250 wavenumbers (400–4000 cm^{-1}) and globally intensity-normalized to $[0, 1]$. Liquid-phase spectra of single molecules are simulated as pure liquids. Mixture spectra are simulated directly with all components in the same box, so they include intermolecular interactions and are not linear combinations of pure-component spectra (Section A). Stage 1 uses paired gas- and liquid-phase spectra for 8,387 molecules. The reference library for identification contains 8,550 liquid-phase spectra, covering every molecule in the binary, ternary and drug mixtures below. Binary mixtures (21,104 instances) contain 213 unique molecules and ternary mixtures (7,985 instances) contain 40 unique molecules. The binary and ternary data are split at the instance level (80/10/10), so every test mixture is a combination not seen in training, but its molecules also appear in training mixtures. Generalization to new molecules is evaluated separately (Section 4.4). "Drug" mixtures (6,553 instances) of 115 drug-like compounds serve as an out-of-distribution (OOD) set (Section 4.4). All drug molecules are excluded from both training stages.

Metrics. To make evaluation invariant to component order, we apply Hungarian matching to align the predicted component spectra with ground-truth components, minimizing pairwise L2 distance. Each predicted spectrum is then queried against the reference library by cosine similarity, producing a ranked list of candidate molecules. We report component cosine similarity (Cos), the mean cosine between each matched predicted and ground-truth pair. We also report R@K (per-component), the fraction of predicted components whose correct molecule appears in the top-K candidates. For mixture-level accuracy, we report Mix R@1 (Mix), the fraction of mixtures where every component is correctly identified at rank 1.

Baseline Methods. We compare against seven methods in three groups. None of them was designed for this task. We adapt each one and keep its original algorithm wherever possible (Section E). (1) Library-based: non-negative least squares (NNLS) (Lawson and Hanson, 1974) and SUnSAL (Bioucas-Dias and Figueiredo, 2010), which adds ℓ_1 regularization solved with the alternating direction method of multipliers, explain each mixture as a non-negative weighted sum of library spectra and return the K library spectra with the largest weights. They use the same 8,550-spectrum library as our identification step. (2) Matrix factorization: MCR-ALS (Tauler et al., 1993) and VCA with fully constrained least squares (VCA+FCLS) (Heinz and Chang, 2001; Nascimento and Bioucas-Dias, 2005) were designed for many spectra that share the same few pure components, such as a reaction monitored over time or a hyperspectral image. Each of our mixtures instead contains its own K of the 213 (binary) or 40 (ternary) molecules, so we factorize all training mixtures into one spectrum per molecule and keep the K spectra with the largest weights for each test mixture. These baselines do not use the library to unmix. They use it only to identify the predicted spectra. (3) Deep learning: CNNAEU (Palsson et al., 2021), CyCU-Net (Gao et al., 2022), and DeepTrans-HSU (Ghosh et al., 2022) are autoencoders for hyperspectral images, originally trained without supervision, whose endmember spectra are decoder weights shared by all pixels. To our knowledge no deep learning method exists for mixture-only unmixing of liquid-phase molecular IR spectra. Because our components differ from mixture to mixture, we keep each method's encoder, replace the shared decoder with one that outputs K spectra per mixture, and train with the same supervised component loss as SLOT-IR.

Comparison fairness. All methods use the same data splits, evaluation, and identification library. Baselines that learn from data use only the training split, and the test split is never used to choose any setting (Section E). SUnSAL's ℓ_1 weight is chosen by component cosine on the validation split. The three deep learning baselines share the loss and optimizer of SLOT-IR and one common setting (learning rate 10^{-4}), chosen on the validation split because it trained stably for all three methods. Section E describes this choice and the search behind it. SLOT-IR settings are also chosen by validation scores (Section G). Stochastic methods, including SLOT-IR and the deep learning baselines, are evaluated over five initialization seeds (mean $\pm$ std). 5-fold cross-validation is in Section B and SLOT-IR hyperparameters in Table 8.

Library and data access. NNLS and SUnSAL use the reference library at test time as part of the algorithm, whereas SLOT-IR uses it only for post-hoc identification. Because the library includes the drug molecules, this gives NNLS and SUnSAL an advantage on molecules absent from training. SLOT-IR also uses paired gas-liquid spectra in Stage 1, which the baselines do not use. The comparison is therefore on a shared benchmark and does not control for data access. Section 4.3 shows that pretraining on gas-phase spectra alone performs comparably.

4.2 Unmixing Results

Binary mixtures. As reported in Table 1, SLOT-IR leads on every metric. On the binary test set it achieves 81.9% per-component R@1, 1.4 points above the strongest baseline SUnSAL, which unmixes by searching the reference library directly. SLOT-IR also reaches 96.6% R@5, 11.4 points higher than SUnSAL, meaning the correct molecule appears in the top 5 candidates for nearly all components. CNNAEU and CyCU-Net identify 61.0% and 60.5% of components. DeepTrans-HSU reaches a high cosine similarity of 0.925 but identifies only 29.4% of components. Without any library, MCR-ALS and VCA+FCLS learn one spectrum per molecule from the training mixtures and identify 41.7% and 39.6% of components, about half the rate of SLOT-IR.

Ternary mixtures. On the ternary test set (n=799), SLOT-IR achieves 94.3% per-component R@1, 15.3 points above the strongest baselines, NNLS and SUnSAL (both 79.0%). Although unmixing three-component mixtures is inherently more complex than binary, the absolute performance is higher. This is likely a dataset effect. The ternary data contain only 40 unique molecules, compared with 213 for binary. Each ternary molecule therefore appears in about three times as many training mixtures (479 on average, compared with 159 for binary).

4.3 Ablation Study

Removing Stage 1. We train SLOT-IR from random initialization without Stage 1, using the best of 36 hyperparameter configurations (Section D). Without Stage 1, R@1 on the binary test set falls from

Table 1: Unmixing results on binary and ternary mixtures. Methods are grouped into Matrix Factorization (MF), Library-based (Library), and Deep Learning (DL). Cos: mean cosine similarity between matched predicted and ground-truth component spectra. R@K: fraction of components whose correct molecule is in the top-K library matches. Mix: fraction of mixtures with all components correct at rank 1. Accuracy values are in %. Stochastic methods report mean ± std over 5 initialization seeds. For SLOT-IR, Stage 1 is trained once and Stage 2 is trained with each seed.

Method	Type	Binary (n=2,111)					Ternary (n=799)		
		Cos	R@1	R@5	R@10	Mix	R@1	R@5	Mix
MCR-ALS	MF	.685	41.7	52.1	55.7	17.6	77.7	85.7	48.2
VCA+FCLS	MF	.825	39.6	59.1	66.4	15.3	51.0	61.9	12.1
NNLS	Library	.942	80.2	85.1	87.6	64.0	79.0	83.8	52.8
SUnSAL	Library	.943	80.5	85.2	87.7	64.2	79.0	83.9	52.4
CNNAEU	DL	$.899_{\pm.002}$	$61.0_{\pm1.0}$	$78.6_{\pm0.9}$	$83.2_{\pm0.9}$	$37.1_{\pm1.4}$	$57.8_{\pm1.0}$	$70.4_{\pm0.6}$	$15.7_{\pm2.1}$
CyCU-Net	DL	$.898_{\pm.001}$	$60.5_{\pm0.6}$	$77.5_{\pm0.6}$	$82.2_{\pm0.5}$	$37.7_{\pm1.1}$	$57.5_{\pm0.9}$	$70.1_{\pm0.2}$	$15.3_{\pm1.2}$
DeepTrans	DL	$.925_{\pm.013}$	$29.4_{\pm8.4}$	$57.0_{\pm9.1}$	$72.4_{\pm8.8}$	$6.8_{\pm4.4}$	$43.2_{\pm4.0}$	$55.9_{\pm4.9}$	$4.1_{\pm1.7}$
SLOT-IR	DL	$\mathbf{.983}_{\pm.004}$	$\mathbf{81.9}_{\pm5.8}$	$\mathbf{96.6}_{\pm1.9}$	$\mathbf{98.5}_{\pm0.7}$	$\mathbf{67.1}_{\pm9.9}$	$\mathbf{94.3}_{\pm2.6}$	$\mathbf{98.3}_{\pm1.2}$	$\mathbf{83.9}_{\pm7.0}$

Table 2: **Ablation and data augmentation analysis** (mean ± std over 5 seeds). Rows with Stage 1 share one pretrained model, and only Stage 2 is retrained for each seed. Accuracy values are in %. †Evaluated on each seed's drug test split (656 mixtures). All other drug evaluations use all drug mixtures.

Variant	Binary Test				Drug			
	Cos	R@1	R@5	Mix	Cos	R@1	R@5	Mix
SLOT-IR (default)	$.983_{\pm.004}$	$81.9_{\pm5.8}$	$96.6_{\pm1.9}$	$67.1_{\pm9.9}$	$.752_{\pm.003}$	$1.1_{\pm0.1}$	$4.9_{\pm0.5}$	$0.0_{\pm0.0}$
w/o Stage 1 (tuned)	$.815_{\pm.203}$	$44.7_{\pm36.6}$	$56.9_{\pm45.8}$	$32.5_{\pm27.2}$	$.629_{\pm.105}$	$0.4_{\pm0.3}$	$2.3_{\pm1.5}$	$0.0_{\pm0.0}$
+ Drug train	$.987_{\pm.002}$	$86.2_{\pm2.3}$	$98.4_{\pm0.5}$	$74.2_{\pm4.1}$	$.963_{\pm.005}$	$83.2_{\pm3.9}$†	$93.7_{\pm1.6}$†	$71.2_{\pm5.5}$†
+ Synthetic drug	$.983_{\pm.003}$	$80.4_{\pm3.7}$	$96.3_{\pm1.5}$	$64.3_{\pm6.2}$	$.893_{\pm.007}$	$51.9_{\pm5.7}$	$68.2_{\pm4.3}$	$28.3_{\pm6.5}$

81.9% to 44.7 ± 36.6% and Mix R@1 from 67.1% to 32.5% (Table 2). Two of the five seeds collapse to near 0% R@1, predicting close to an average spectrum, and the other three average 74.4% R@1. Training from random initialization is fragile. Only 8 of the 36 configurations exceed 1% R@1, and the same two seeds still collapse when early stopping is relaxed (Section D). Stage 1 therefore raises average accuracy and makes training reliable across seeds.

Changing the Stage 1 objective. We next ask what Stage 1 needs to learn. We replace the gas-to-liquid translation objective with three alternatives while keeping Stage 2 identical (Table 3). The first two are a liquid-to-liquid autoencoder and a gas-to-gas autoencoder. The third is translation between shuffled gas-liquid pairs, which keeps the same spectra but breaks the correspondence between input and target molecules. Both autoencoders reach accuracy similar to translation. The liquid and gas autoencoders reach 88.1 ± 2.7% and 84.6 ± 7.5% R@1, and translation reaches 86.8 ± 4.4%. With 5 seeds, neither difference is statistically significant ($p = 0.63$ and 0.62, Welch's t-test). The gas autoencoder varies more across seeds, with two of its five seeds at 78.7% and 72.8%. Shuffled pairs fail on every seed (0.0% R@1, with predictions close to an average spectrum). The failure of shuffled pairs suggests that Stage 1 helps only when the input and target are spectra of the same molecule. The comparable autoencoders further indicate that learning coherent spectra of individual molecules in either phase gives a similar benefit.

Gas-phase pretraining. The gas autoencoder is the most practical of these objectives. Pretraining on gas-phase spectra alone, without any liquid-phase data, performs comparably to paired translation. Gas-phase spectra describe isolated molecules, so obtaining them requires neither a condensed-phase simulation nor a liquid-phase measurement (Section A). This result therefore points to a practical way to scale pretraining to much broader chemical spaces.

Table 3: **Stage 1 objectives.** Stage 2 training on binary mixtures is identical for all rows and uses the setting of Table 1. Each seed trains its own Stage 1 and Stage 2 (mean ± std over 5 seeds), so the translation row differs from Table 1, where Stage 1 is trained once. Accuracy values in %.

Stage 1 objective	**Binary test** (n=2,111)		
	Cos	**R@1**	**Mix**
Gas→Liquid translation (default)	$.987_{\pm.005}$	$86.8_{\pm4.4}$	$75.4_{\pm7.8}$
Liquid→Liquid autoencoder	$.989_{\pm.002}$	$88.1_{\pm2.7}$	$77.7_{\pm4.9}$
Gas→Gas autoencoder	$.985_{\pm.007}$	$84.6_{\pm7.5}$	$72.0_{\pm12.5}$
Shuffled gas-liquid pairs	$.585_{\pm.015}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$

4.4 Generalization and Data Augmentation

We test generalization at three levels, based on how much of each molecule the model saw in training: (1) new combinations of molecules seen in both stages, (2) held-out molecules that were seen in Stage 1 but not in mixture training, and (3) drug molecules that were seen in no training stage. To judge component cosine similarity, we use the input mixture as a reference. If the mixture spectrum itself is taken as the prediction for every component, its cosine with the true components is 0.769 on the binary test set and 0.798 on the drug mixtures. A model that separates the components should score above this reference. After the three levels, we test whether two data augmentation strategies improve generalization to drug molecules.

Seen molecules. Level (1) is the main evaluation in Table 1, where every test mixture is a new combination of molecules seen in both training stages. SLOT-IR reaches a component cosine of 0.983, well above the 0.769 of the input mixture. It identifies 81.9% of components at rank 1.

Held-out molecules. We hold out 50 molecules from mixture training and keep them in Stage 1 (held-out molecules). For the 47 of them that appear in test mixtures, the predicted spectra of these molecules reach a cosine of 0.865 with their true spectra. Using the input mixture as the prediction instead gives 0.751 for the same molecules. The model therefore moves its predictions closer to the true spectra, but only 11.0% of these molecules are identified at rank 1 (Section C). The predictions are not yet close enough for identification.

Unseen drug molecules. Trained only on binary mixtures, SLOT-IR does not generalize well to drug molecules. Its predicted components have cosine 0.752 with the true components, which is lower than the 0.798 of the input mixture. R@1 drops to 1.1%. The drug mixtures were also built to be difficult, with confounding cases such as isomers and similar functional-group pairs (Melle et al., 2026).

To test whether limited data coverage explains this, we try two data augmentation strategies. The first adds drug mixtures to training. The second shows the model each drug molecule but never a drug mixture.

Training on drug mixtures. We split the drug mixtures 8:1:1 and add the 5,242 training mixtures to the binary training split, which converts the drug evaluation from OOD to in-distribution. On the 656 mixtures of the drug test split, R@1 reaches 83.2% and Mix R@1 71.2%, while binary test performance is preserved (Table 2). The model can therefore learn to identify drug molecules when they appear in training.

Training on synthetic drug mixtures. For each pair of the 115 drug molecules, we average their liquid-phase spectra with shifting noise. This gives 6,555 synthetic binary mixtures, which we add to the binary training split. The MD-simulated drug set is never used in training, so the whole set serves as the test set. R@1 rises from 1.1% to 51.9% and Mix R@1 to 28.3%. Component cosine rises to 0.893, above the 0.798 of the input mixture. Binary test R@1 stays within seed variation (80.4%, compared with 81.9%). Training on the simulated drug mixtures themselves still works better (83.2% R@1 on the drug test split). This gap is consistent with averaged spectra missing the intermolecular effects that mixture simulations include.

SLOT-IR generalizes to new combinations of molecules it has seen. For held-out molecules, its predictions are closer to the true spectra than the input mixture but are rarely identified. It does not

generalize well to drug molecules absent from all training. This weakness is consistent with limited molecular coverage in mixture training. Adding synthetic mixtures of the new molecules reduces it.

5 Post-hoc SAE Analysis

Because SLOT-IR keeps a separate embedding for each component, we can ask post hoc whether these embeddings carry chemical information. We apply a Sparse Autoencoder (SAE) to frozen slot embeddings and compare its features with functional group (FG) annotations. The SAE is trained after unmixing training, is not part of the unmixing model, and does not affect its performance.

Setup. SAEs decompose dense neural activations into a sparse set of latent features, ideally each corresponding to one interpretable concept (Bricken et al., 2023; Cunningham et al., 2023). We use a BatchTopK SAE (Bussmann et al., 2024), which encodes each slot embedding $\mathbf{s} \in \mathbb{R}^d$ into D=2,048 latent features and reconstructs it:

$$\mathbf{z} = \text{BatchTopK}(\mathbf{W}_{\text{enc}}\,\mathbf{s} + \mathbf{b}_{\text{enc}},\ k{\cdot}B) \in \mathbb{R}^D, \quad \hat{\mathbf{s}} = \bar{\mathbf{W}}_{\text{dec}}\,\mathbf{z} + \mathbf{b}_{\text{dec}}, \tag{4}$$

where $\bar{\mathbf{W}}_{\text{dec}}$ is the column-normalized decoder and BatchTopK keeps the largest $k{\cdot}B$ activations in a batch of size B, so that on average k=50 features are active per sample. The SAE is trained with mean squared reconstruction error plus a small auxiliary loss (weight 0.01) that revives inactive features. It is trained on the 33,766 training-split slot embeddings of the binary model (seed 0), with 20% of them held out to monitor reconstruction loss. All analysis uses the 4,222 test-split slot embeddings. We use the 48 FG labels of Section 3.1, which are never used during training. Slots have no fixed order, so each slot takes the FG labels of the ground-truth molecule matched to its decoded spectrum by the Hungarian L2 assignment used in training.

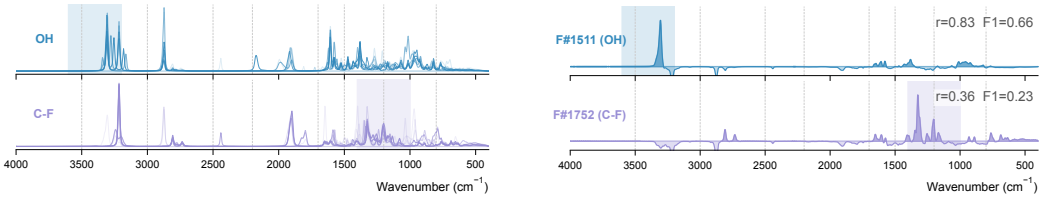
Selective features. For each SAE feature and FG, we compute the Spearman correlation between the feature's activations and the FG label across test-set slots. We call a feature *selective* for an FG if its largest absolute correlation exceeds 0.3 and is at least twice its second largest. This yields 118 selective features out of 2,048. To test whether this could arise by chance, we randomly permute the FG labels across samples and repeat the entire selection procedure: in 500 permutations, no feature is ever selective ($p < 0.002$). The association therefore depends on the true correspondence between slots and their functional groups.

Alignment with IR bands. Correlation with a label does not show that a feature responds to the group's absorption. We therefore check each selective feature against the textbook IR bands of its group (Pavia et al., 2014; Silverstein et al., 2014), which played no role in selecting it. We decode the feature into a spectrum and measure how much of its signal falls inside the group's band, compared with the share expected if the signal were spread evenly over the spectrum (Section F). All 29 OH features put 1.9 to 8.2 times more signal in the O–H stretching band than this even spread. Over all groups with a textbook band, 70 of the 106 selective features are above it. Alignment is clear for OH and weaker for some groups, such as SO_2. (Section F).

Figure 2 shows two examples: the feature most correlated with OH (r=0.83) and a weaker one, the only selective C–F feature (r=0.36). On the test set, the 20 slots on which each feature is most active all belong to molecules with its group. By chance, only about 3 of 20 random slots would contain OH, and fewer than 1 would contain C–F. We also subtract the average true spectrum of slots where the feature is inactive from that of slots where it is strongly active (the top 10% of its nonzero activations). The difference peaks inside the textbook band, near 3305 cm^{-1} for O–H and near 1325 cm^{-1} for C–F (Figure 2b). Together, these checks show that per-component embeddings carry information associated with functional groups, with clear alignment with IR bands for groups such as OH.

6 Discussion and Conclusion

Limitations. All spectra are MD simulations from a single benchmark, so performance on measured spectra is untested. The training data cover a limited set of molecules, and new molecules are rarely identified unless their mixtures are added to training (Section 4.4). We assume the number of components K is known, evaluate only equal-proportion mixtures, and do not estimate proportions. A molecule absent from the reference library cannot be identified. The SAE analysis is correlational, and its SMARTS-derived FG labels are approximations.



(a) Highest-activating mixture spectra.

(b) Activation-conditioned spectral difference.

Figure 2: **SAE features with spectral signatures** for OH and C–F. (a) Mixture spectra of the samples on which each feature is most active. (b) Mean ground-truth component spectrum of high-activation samples (top 10% of nonzero activations) minus that of inactive samples. Shaded bands mark textbook absorption ranges.

Broader impact. SLOT-IR is intended as a tool for automated chemical analysis. It is trained and evaluated on simulated spectra, so its predictions on real samples should be confirmed with standard analytical methods before they inform decisions, for example in quality control or forensic analysis.

Conclusion. We introduced SLOT-IR, a two-stage slot-based architecture for IR spectral unmixing that first pretrains the encoder on single-molecule spectra and then decomposes mixture spectra into per-component predictions. SLOT-IR outperforms all baselines on binary and ternary mixtures and is, to our knowledge, the first deep learning method that reconstructs each component spectrum from a single liquid-phase IR mixture spectrum. Our pretraining experiments suggest that the key ingredient is learning coherent spectra of individual molecules, and that pretraining on gas-phase spectra alone performs comparably. Since gas-phase spectra of isolated molecules need no condensed-phase simulation, this offers a path to scaling spectral unmixing to broader chemical coverage. A post-hoc SAE analysis further shows that the per-component embeddings carry information statistically associated with functional groups.

Acknowledgments and Disclosure of Funding

This work was partially supported by NSF (2106859, 2119643, 2312501, and 2609582) and NIH (U54DK097771). We also gratefully acknowledge support from Amazon and NEC. Y.J.U.M and D.S.K. acknowledge support from Scialog grant #SA-AUT-2024-024a from Research Corporation for Science Advancement and Frederick Gardner Cottrell Foundation. Y.J.U.M. acknowledges supplementary funding from the Amazon AI Ph.D. Fellowship at UCLA Samueli.

References

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4

Usha Bhalla, Alex Oesterling, Claudio Mayrink Verdun, Himabindu Lakkaraju, and Flavio P Calmon. Temporal sparse autoencoders: Leveraging the sequential nature of language for interpretability, 2025. 4

José M Bioucas-Dias and Mário A T Figueiredo. Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing. In *2nd Workshop on Hyperspectral Image and Signal Processing (WHISPERS)*, 2010. 1, 3, 6, 15

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, Anthropic, 2023. 3, 4, 9

Bart Bussmann, Patrick Leask, and Neel Nanda. BatchTopK sparse autoencoders, 2024. 4, 9

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020. 4

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. 3, 4, 9

Xiaqiong Fan, Yue Wang, Chuanxiu Yu, Yuanxia Lv, Hailiang Zhang, Qiong Yang, Ming Wen, Hongmei Lu, and Zhimin Zhang. A universal and accurate method for easily identifying components in raman spectroscopy based on deep learning. *Analytical Chemistry*, 2023. 3

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders, 2024. 4

Lianru Gao, Zhu Han, Danfeng Hong, Bing Zhang, and Jocelyn Chanussot. CyCU-Net: Cycle-consistency unmixing network by learning cascaded autoencoders. *IEEE Transactions on Geoscience and Remote Sensing*, 2022. 2, 6, 15

Dimitar Georgiev, Álvaro Fernández-Galiana, Simon Vilms Pedersen, Georgios Papadopoulos, Ruoxiao Xie, Molly M Stevens, and Mauricio Barahona. Hyperspectral unmixing for raman spectroscopy via physics-constrained autoencoders. *Proceedings of the National Academy of Sciences*, 2024. 3

Preetam Ghosh, Swalpa Kumar Roy, Bikram Koirala, Behnood Rasti, and Paul Scheunders. Hyperspectral unmixing using transformer network. *IEEE Transactions on Geoscience and Remote Sensing*, 2022. 2, 3, 6, 15

Jiaojiao Han, Wujiang Xu, Mingyu Jin, and Mengnan Du. SAGE: An agentic explainer framework for interpreting SAE features in language models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Industry Track*, 2026. 4

Daniel C Heinz and Chein-I Chang. Fully constrained least squares linear spectral mixture analysis method for material quantification in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 2001. 6, 15

Thomas Kipf, Gamaleldin F Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. In *International Conference on Learning Representations (ICLR)*, 2022. 3

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning (ICML)*, 2020. 4

Avinash Kori, Francesco Locatello, Ainkaran Santhirasekaram, Francesca Toni, Ben Glocker, and Fabio De Sousa Ribeiro. Identifiable object-centric representation learning via probabilistic slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 3

Charles L Lawson and Richard J Hanson. *Solving Least Squares Problems*. Prentice-Hall, 1974. 1, 6, 15

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3, 5

Yannah J. U. Melle, Thanh Nguyen, Jeffrey Lopez, and Daniel Schwalbe-Koda. Automatic identification of compounds in molecular mixtures from liquid-phase infrared spectra. *Chem. Sci.*, 2026. 3, 5, 8, 13, 21

José M P Nascimento and José M Bioucas-Dias. Vertex component analysis: A fast algorithm to unmix hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 2005. 2, 3, 6, 15

Burkni Palsson, Magnus O Ulfarsson, and Johannes R Sveinsson. Convolutional autoencoder for spectral–spatial hyperspectral unmixing. *IEEE Transactions on Geoscience and Remote Sensing*, 2021. 2, 3, 6, 15

Donald L Pavia, Gary M Lampman, George S Kriz, and James A Vyvyan. *Introduction to Spectroscopy*. Cengage Learning, 2014. 1, 9, 16

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024. 4

Robert M Silverstein, Francis X Webster, David J Kiemle, and David L Bryce. *Spectrometric Identification of Organic Compounds*. John Wiley & Sons, 2014. 3, 9, 16

Lin Tan, Yue Wang, Hailiang Zhang, Jinyu Sun, Qiong Yang, Xiao Yang, Zhimin Zhang, and Hongmei Lu. DeepMIR: A hybrid convolutional neural network-transformer framework for accurate identification of target components from mid-infrared spectra of mixtures. *Analytical Chemistry*, 2025. 1, 3

Romá Tauler, Bruce R Kowalski, and Sydney Alexis Fleming. Multivariate curve resolution applied to spectral data from multiple runs of an industrial process. *Analytical Chemistry*, 1993. 2, 3, 6, 15

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C Daniel Freeman, Theodore R Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. Transformer Circuits Thread, Anthropic, 2024. 4

Ziyi Wu, Jingyu Hu, Wuyue Lu, Igor Gilitschenski, and Animesh Garg. Slotdiffusion: Object-centric generative modeling with diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 4

A Dataset Construction Details

Data generation. All gas-phase, liquid-phase, and mixture spectra are computed with all-atom molecular dynamics (MD) simulations (Melle et al., 2026). Liquid-phase simulations are nanosecond-long trajectories with up to 100 molecules per box. Pure-liquid simulations contain a single molecular species, and mixture simulations contain two or three species in the same box. The simulations explicitly treat intermolecular interactions such as hydrogen bonding, restricted rotational and translational motion, and anharmonicity. As a result, mixture spectra show peak shifts, peak broadening, overlapping peaks, and obscured fingerprint regions, and are not linear combinations of pure-component spectra. Gas-phase spectra are also computed with MD, which captures anharmonic and rotational effects that are not available from harmonic approximations. Each liquid-phase trajectory costs about 15 GPU-minutes, and the dataset corresponds to about 44k simulations. The full simulation protocol is described in Melle et al. (2026).

Splits and preprocessing. Each mixture dataset contains unique combinations of molecules. The binary and ternary mixture datasets are partitioned into 80/10/10 train/val/test splits with random seed 42. Spectra are 1,250-dimensional, sampled at evenly spaced points from 400 to 4000 cm^{-1} (about 2.9 cm^{-1} apart), and clamped to $[0, 1]$ after global intensity normalization. The spectral reference library contains 8,550 liquid-phase spectra used for post-hoc cosine matching during evaluation, including the 115 drug molecules. The paired gas- and liquid-phase spectra for Stage 1 include 8,387 pairs. Of the 115 drug molecules, 35 also have paired gas-liquid spectra in the source data. We remove these from Stage 1 so that no drug molecule is seen in any training stage.

Dataset	N Total	N Test	Molecules	Components
Binary	21,104	2,111	213	K=2
Ternary	7,985	799	40	K=3
Drug	6,553	6,553	115	K=2

Table 4: Mixture dataset statistics. The drug mixtures are excluded from training entirely, except in the + Drug train experiment of Table 2, which splits them 8:1:1. All 213 binary molecules appear in training, and 210 of them appear in the test split.

B Cross-Validation

We evaluate SLOT-IR with 5-fold cross-validation on both binary and ternary datasets to check whether performance depends on the data split. Each fold is used as the test set exactly once, and the remaining data is split 90/10 for training and validation. Instances are assigned to folds with `random.Random(42).shuffle`, and all folds use the same Stage 1 checkpoint.

Setting	Cos	R@1 (%)	R@5 (%)	R@10 (%)	Mix R@1 (%)
Binary (n=21,104, 5-fold)	0.984±0.002	84.3±3.3	98.0±1.0	99.4±0.3	70.9±5.9
Ternary (n=7,985, 5-fold)	0.979±0.005	94.5±1.6	98.7±0.6	99.1±0.5	84.6±4.1

Table 5: **5-fold cross-validation.** Results are consistent with the fixed-split evaluation in Table 1.

Binary R@1 averages $84.3 \pm 3.3\%$ across folds and ternary R@1 averages $94.5 \pm 1.6\%$, close to the fixed-split results (81.9% and 94.3%).

C Molecule Holdout

To test whether the model decomposes molecules that it was never trained to unmix, we hold out 50 molecules from Stage 2 mixture training while keeping them in Stage 1 (the held-out molecules of Section 4.4), and 47 of them appear in test mixtures. We evaluate per-molecule decomposition quality over 5 seeds (Table 7). The data split and the held-out molecules are the same for every seed. The predicted spectra of held-out molecules reach cosine 0.865, compared with 0.751 when the input

Data	Fold	n_{test}	Cos	R@1 (%)	R@5 (%)	Mix R@1 (%)
Binary	0	4,221	0.982	79.7	96.6	62.8
	1	4,221	0.981	80.9	97.0	64.8
	2	4,221	0.984	86.1	98.3	74.1
	3	4,221	0.986	87.4	99.0	76.3
	4	4,220	0.987	87.4	99.0	76.5
Ternary	0	1,597	0.985	96.2	99.4	89.3
	1	1,597	0.981	94.7	99.0	85.2
	2	1,597	0.983	96.1	99.1	88.6
	3	1,597	0.973	92.9	98.2	81.4
	4	1,597	0.974	92.5	97.9	78.8

Table 6: **Per-fold details.** Binary R@1 ranges from 79.7% to 87.4%, and ternary R@1 from 92.5% to 96.2%.

mixture is used as the prediction. These molecules are still rarely identified (11.0% R@1), far below molecules seen in both stages (83.9%). The drug molecules, absent from all training, reach cosine 0.742 (averaged per molecule) and near-zero R@1.

Category	Definition	N	Cos	R@1 (%)
Seen	In Stage 1 and Stage 2 training	163	0.979±0.014	83.9
Held out	In Stage 1 only	47	0.865±0.165	11.0
Drug	In no training stage	115	0.742±0.113	1.1

Table 7: **Molecule holdout.** N is the number of molecules in each category that appear in test mixtures. Cos is averaged per molecule (mean ± std across molecules), and R@1 is the per-molecule identification rate averaged over molecules and 5 seeds.

D Training Without Stage 1

For the ablation without Stage 1 (Section 4.3), we grid-search the randomly initialized model over learning rate $\{10^{-4}, 5{\times}10^{-4}, 10^{-3}\}$, dropout $\{0.1, 0.3, 0.5\}$, epochs $\{100, 200\}$, and token windows $\{[10, 25, 50], [5, 10, 25]\}$, a total of 36 configurations with one seed each, ranked by validation component cosine (ranking by validation loss selects the same configuration). Only 8 configurations exceed 1% R@1, all with learning rate 10^{-4}, and every configuration with learning rate $5{\times}10^{-4}$ or 10^{-3} fails to train. The best configuration (learning rate 10^{-4}, dropout 0.1, 100 epochs, windows $[10, 25, 50]$) is then trained with 5 seeds for Table 2, reaching R@1 of 71.0, 0.4, 81.6, 0.0, and 70.6% per seed. Because the two failed seeds stop early while still predicting close to an average spectrum, we repeat the 5 seeds with a longer early-stopping patience (20 instead of 5 validation checks). The same two seeds still fail (0.0% R@1), and the other three reach 79.7, 92.8, and 82.3% ($51.0 \pm 41.8\%$ on average). The failures therefore reflect unstable optimization from random initialization rather than early stopping.

E Baseline Adaptation

All baselines were designed for settings different from ours. We first checked each against its original publication (or, where the full text was not accessible, the authors' code), then adapted it as described below.

Shared protocol. All methods use the splits and evaluation of Section 4.1. Methods that learn from data use only the training split. SUnSAL's ℓ_1 weight is chosen by component cosine on the validation

split, and the deep learning setting is described below. The test split is used once, for the reported numbers. NNLS and SUnSAL use the 8,550-spectrum identification library.

NNLS. Non-negative least squares (Lawson and Hanson, 1974) is solved for each mixture against the full library (SciPy implementation), and the K spectra with the largest weights are returned. It has no hyperparameters.

SUnSAL. SUnSAL (Bioucas-Dias and Figueiredo, 2010) solves $\min_x \frac{1}{2}\|Ax - y\|^2 + \lambda\|x\|_1$ subject to $x \geq 0$ with the alternating direction method of multipliers, where A is the library and y the mixture. As in the original, no sum-to-one constraint is used (with it, the ℓ_1 term is constant), and the penalty parameter starts at 0.01 and is doubled or halved every 10 iterations to balance the primal and dual residuals. The original stopping tolerance (10^{-4}) stopped too early at the intensity scale of our spectra: with λ near 0, the fit residual was 4.5 times that of NNLS (20 validation mixtures). We therefore use a tolerance of 10^{-6} with at most 5,000 iterations, at which $\lambda \to 0$ reproduces the NNLS residual. λ is chosen from $\{10^{-5}, \ldots, 10^{-1}\}$ on validation: 10^{-2} for binary and 10^{-3} for ternary. Validation cosine varies little over this range (0.938 to 0.943 binary, 0.880 to 0.896 ternary).

MCR-ALS and VCA+FCLS. Both assume that all spectra in a data set mix the same few pure components. We therefore factorize the matrix of all training mixtures into M spectra, where M is the number of distinct molecules in the training mixtures (213 binary, 40 ternary). Only this count, not the molecule identities, is used. For each test mixture, non-negative weights over the M spectra are estimated and the K spectra with the largest weights are returned. No library is used for unmixing. VCA (Nascimento and Bioucas-Dias, 2005) follows the published algorithm, including its choice of projection by estimated signal-to-noise ratio, and weights are then estimated by FCLS (Heinz and Chang, 2001). VCA selects measured spectra as endmembers and assumes that some measured spectra are nearly pure. Our training data contain only mixtures, so this assumption does not hold, which is a limitation of VCA in this setting rather than of the adaptation. MCR-ALS (Tauler et al., 1993) alternates non-negative least squares for concentrations and spectra, with non-negativity on both, starting from the VCA spectra and rescaling each spectrum to unit maximum after every iteration. It stops when the relative change in lack of fit falls below 0.1%, which took 92 (binary) and 26 (ternary) iterations.

Deep learning baselines. The three methods were designed for hyperspectral images, where many pixels share the same few endmembers. They are trained without supervision, and the endmember spectra are the weights of a linear decoder shared by all pixels. In our task each mixture has its own components, so a shared decoder cannot apply. We keep each method's encoder, replace the shared decoder with one that outputs K component spectra per mixture, and train with the Hungarian-matched MSE component loss of SLOT-IR plus a weighted reconstruction loss between the mixture and the abundance-weighted sum of the predicted components. For CNNAEU (Palsson et al., 2021), the encoder uses 1D instead of 2D convolutions. For CyCU-Net (Gao et al., 2022), we use 1D convolutions and add a cycle term that maps the predicted components back to a mixture, penalizing the difference from the input mixture and the component loss on the components decoded from that cycled mixture. For DeepTrans-HSU (Ghosh et al., 2022), the Transformer operates on tokens of 5 adjacent spectral points instead of spatial patches. All three share the optimizer recipe of SLOT-IR: AdamW (weight decay 10^{-5}), gradient norm clipped at 0.5, cosine learning-rate schedule that falls over one epoch and rises over the next, batch size 64, up to 100 epochs, and early stopping with patience 20 on the validation loss, restoring the best checkpoint. All three use one shared setting, learning rate 10^{-4} and reconstruction-loss weight 1, trained with five seeds. We chose it from a validation-split grid over learning rate (3×10^{-5} to 3×10^{-3}, extended to 10^{-2} for CNNAEU ternary and 10^{-5} for DeepTrans-HSU ternary) and reconstruction-loss weight (0 or 1), run with one seed: it trained stably for all three methods on both data sets, whereas some settings with higher single-seed validation cosine collapsed to near-zero output spectra on other seeds (CyCU-Net binary at 10^{-3}, CNNAEU ternary at 3×10^{-3}). The shared setting is therefore not the best for each method individually. With it, validation cosine averaged over seeds was still rising slowly at 100 epochs, by at most 0.007 between the 90th and the 100th epoch.

F Additional SAE Analysis

Spectral localization. For each selective feature, we decode its direction into a spectrum through the prediction head and compute the fraction of spectral energy inside the textbook IR bands of its

Hyperparameter	Stage 1 (translation)	Stage 2 (unmixing)
Model dimension (d) / attention heads	512 / 8	
Encoder layers / Stage 1 decoder layers	6 / 4	
Token window sizes (spectral points)	10, 25, 50	
Optimizer	AdamW, weight decay 10^{-5}, gradient norm clipped at 0.5	
Learning rate	5×10^{-5}	10^{-4}
Learning-rate schedule	cosine schedule that falls over one epoch and rises over the next, repeating	
Batch size	64	
Dropout	0.1	0.5
Maximum epochs	200	50
Early stopping	none	patience of 5 validation checks (every 500 steps)
Loss	MSE + cosine similarity	Hungarian-matched MSE
Interpretation: BatchTopK SAE		
Latent dimension / active features	2048 / k=50 per sample on average (batch-level)	
Optimizer / epochs / batch size	Adam, learning rate 10^{-3} / 50 / 256	

Table 8: **Hyperparameters.** Ternary Stage 2 uses the same settings as binary.

FG (Pavia et al., 2014; Silverstein et al., 2014), compared with the fraction of the wavenumber range those bands cover (the value expected by chance). This test uses spectral positions, which play no role in selecting the features, so it is independent of the selection. Across the 106 selective features whose FG has a textbook band, the mean in-band energy fraction is 0.203, compared with 0.065 by chance, and 70 of 106 features are above chance (one-sided Wilcoxon signed-rank test, $p = 5 \times 10^{-8}$). All 29 OH features are 1.9 to 8.2 times above chance. Most COO features (11 of 15) are above chance, while SO_2 (1 of 11) and CHO (0 of 3) features are not. A few band-table entries are less certain (N=N, the ring-oxygen group Ar-O, SO_3H, P=O). Excluding their features, 44 of 67 remain above chance ($p = 5 \times 10^{-7}$).

Choice of examples for Figure 2. For every FG, we score its three most correlated features by the share of their activation-conditioned spectral difference that falls inside the FG's textbook bands. The OH and C–F features score highest among all candidates, so we show the feature most correlated with OH (feature 1511) and the only selective C–F feature (feature 1752).

G Hyperparameters and Reproduction

The hyperparameters are provided in Table 8. Experiments run on single NVIDIA RTX A6000 GPUs. One Stage 1 run takes about 2 GPU-hours (200 epochs of about 33 seconds each), and one binary Stage 2 run takes up to about 2 GPU-hours (at most 50 epochs of about 2.5 minutes each). All experiments in the paper, including the hyperparameter searches and baselines, took a few hundred GPU-hours. Code is available at `https://github.com/JingruG/SLOT-IR`.

Hyperparameter selection. Settings are chosen by validation scores, never test scores. For Stage 1, learning rate $\{5\times10^{-5}, 10^{-4}\}$ and dropout $\{0.1, 0.5\}$ are ranked by validation translation cosine. Dropout 0.1 and 0.5 tie (0.9559 and 0.9560 at learning rate 5×10^{-5}), and we keep 0.1, which has the lower validation loss. For Stage 2, learning rate $\{5\times10^{-5}, 10^{-4}\}$ and dropout $\{0.3, 0.5\}$ (ternary also 5×10^{-4}) are ranked by validation component cosine with one seed. Learning rate 10^{-4} with dropout 0.5 wins for binary and ternary, and it is also better than 5×10^{-5} on validation averaged over five seeds, with validation cosine 0.9837 compared with 0.9826 for binary and 0.9843 compared with 0.9750 for ternary.

Multi-seed evaluation. All stochastic results report mean $\pm$ std (population std) across five seeds {0,1,2,3,4} with a fixed data split (seed 42). Unless stated otherwise, Stage 1 is trained once (seed 42) and Stage 2 starts its slot projection and prediction head from fresh random weights for each seed. Table 3 instead trains a separate Stage 1 for each seed.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state three claims: state-of-the-art unmixing performance of a two-stage slot-based architecture (Section 4.2), that pretraining on single-molecule spectra, including gas-phase spectra alone, drives the gain (Section 4.3), and that an SAE finds features statistically associated with functional groups (Section 5). The scope of the SAE claim is stated in Sections 5 and 6.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 6 discusses limitations including simulated data, limited molecular coverage, fixed K and equal proportions, and the correlational nature of the SAE analysis.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: This work does not contain formal theorems or proofs. Mathematical formulations for the model and loss functions are given in Section 3, and the metrics in Section 4.1.

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: Section G includes all hyperparameters and the code repository link. Section 4.1 and Section E specify baseline configurations. Section A details dataset construction. Section 3 details the model architecture.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Code is available at https://github.com/JingruG/SLOT-IR (Section G).

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: Table 8 lists all hyperparameters. Dataset details and splits are specified in Section 4.1 and Section A.

   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: We report mean $\pm$ std over 5 initialization seeds, and Welch's t-tests for the Stage 1 controls. 5-fold cross-validation is also reported.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Section G states the hardware (single NVIDIA RTX A6000 GPUs), the time per run (about 2 GPU-hours for Stage 1 and up to about 2 GPU-hours for one binary Stage 2 run), and the total of a few hundred GPU-hours for all experiments, including the hyperparameter search.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research has no ethical concerns.

   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: Section 6 states the intended use (automated chemical analysis) and that, because the model is trained on simulated spectra, its predictions on real samples should be confirmed with standard analytical methods. We see no direct path to harmful use.

    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 work is domain-specific and has no meaningful risk for misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The spectra are from Melle et al. (2026), cited in Section 4.1 and used with the permission of its authors, two of whom (Melle and Schwalbe-Koda) are co-authors of this paper. All baselines are our implementations of the cited methods (Section E), and all reference methods are cited. The code uses PyTorch, SciPy, and RDKit, all under BSD licenses.

    Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: The code is released at https://github.com/JingruG/SLOT-IR under the MIT license, with a README describing installation, data preparation, and the command that produces each table and figure.

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: This work does not involve crowdsourcing or human subjects.

Guidelines:

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

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

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

Answer: [N/A]

Justification: No human subjects were involved in this research.

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: LLMs are not used as a component of the core methodology.

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.