3.3 RePS training objectives

RePS builds on BiPO [Cao et al., 2024] and SimPO [Meng et al., 2024], and has a reference-free bi-directional preference optimization objective. Unlike BiPO, we argue that the policy LM should not be constrained to stay close to the reference model given that the steering behaviors are usually considered as *irregular* and, thus, *not preferred* by the reference model. For example, responses to programming questions that mention the Golden Gate Bridge are very low probability, and so steering objectives are often at odds with the model's tendencies.

RePS is bi-directional, and first constructs the likelihood differences for positive steering as:

$$\Delta_{\Phi}^{+} = \overbrace{\frac{\beta^{+}}{|\mathbf{y}^{\mathbf{c}}|}\log\Big(p_{\Phi}\big(\mathbf{y}^{\mathbf{c}} \mid \mathbf{x}, \mathbf{h}^{l} \leftarrow \Phi_{\text{Steer}}\big)\Big)}^{\text{Likelihood of \textbf{steered} (winning) response}} - \underbrace{\frac{1}{|\mathbf{y}|}\log\Big(p_{\Phi}\big(\mathbf{y} \mid \mathbf{x}, \mathbf{h}^{l} \leftarrow \Phi_{\text{Steer}}\big)\Big)}_{\text{Likelihood of original (losing) response}} \tag{5}$$

where $\beta^{+} = \max(\log(p(\mathbf{y} \mid \mathbf{x})) - \log(p(\mathbf{y}^{\mathbf{c}} \mid \mathbf{x})), 1)$ serves as a scaling term to weight the likelihood of the steered response higher if the reference model considers the steered response to be *unlikely*. We adopt the length normalizations from SimPO [Meng et al., 2024].

RePS also constructs an asymmetric objective for negative steering as:

$$\Delta_{\Phi}^{-} = \overbrace{\frac{\beta^{-}}{|\mathbf{y}|}\log\Big(p_{\Phi}\big(\mathbf{y} \mid \mathbf{x}, \mathbf{h}^{l} \leftarrow \Phi_{\text{Null}}\big)\Big)}^{\text{Likelihood of original (winning) response}} - \underbrace{\frac{1}{|\mathbf{y}^{\mathbf{c}}|}\log\Big(p_{\Phi}\big(\mathbf{y}^{\mathbf{c}} \mid \mathbf{x}, \mathbf{h}^{l} \leftarrow \Phi_{\text{Null}}\big)\Big)}_{\text{Likelihood of \textbf{steered} (losing) response}} \tag{6}$$

where $\beta^{-} = \max(\log(p(\mathbf{y}^{\mathbf{c}} \mid \mathbf{x})) - \log(p(\mathbf{y} \mid \mathbf{x})), 1)$, and Φ_{Steer} and Φ_{Null} are two asymmetric intervention parameterizations. Learned parameters are shared across these two interventions. To illustrate, we can further contextualize these two interventions by instantiating them with SV interventions. Φ_{Steer} becomes $\Phi_{\text{SV}}(\mathbf{h}^{l}; f)$ where f is a randomly sampled positive steering factor from a predefined set as described in section 5.1 and appendix D.[2] Taking inspiration from Widdows [2003], we parameterize Φ_{Null} by *nulling out* any projection along the steering direction from from $\mathbf{h}^{l}$ as:

$$\Phi_{\text{Null}}(\mathbf{h}^{l}) = \mathbf{h}^{l} - \frac{\text{ReLU}(\mathbf{h}^{l} \cdot \mathbf{w}_1)}{\|\mathbf{w}_1\|^2}\mathbf{w}_1 \tag{7}$$

Finally, we sum up the preference losses for both directions as:

$$\min_{\Phi}\Big\{-\mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{y}^{\mathbf{c}})\sim\mathcal{D}_{\text{Train}}}\Big[\log\sigma\Big(\Delta_{\Phi}^{+}\Big) + \log\sigma\Big(\Delta_{\Phi}^{-}\Big)\Big]\Big\} \tag{8}$$

Intuitively, RePS learns to increase the likelihood of the steered response when the intervention is applied with a sampled positive steering factor, and learns to null out any information in the steering direction when the intervention is applied negatively. Note that RePS does not need additional training data other than preference pairs.

RePS with low-rank settings. While positive steering as Φ_{SV} or negative steering as Φ_{Null} assumes linear encoding, RePS can easily be adapted to low-rank settings, such as LoRA or ReFT. As described in eq. (10) and eq. (11), we provide randomly sampled steering factors during training. For LoRA or ReFT interventions, we replace Φ_{Null} by sampling negative steering factors.

4 Intervention-based methods for steering

Rank-1 steering vectors (SV; Turner et al. [2023a]). SV resembles the simplest form of interventions that stores the steering concept in a single rank-1 vector with little inference-time computation overhead [Rimsky et al., 2024, Li et al., 2024a, Marks and Tegmark, 2024]. We can formulate the intervention for any SV as:

$$\Phi_{\text{SV}}(\mathbf{h}^{l}, \alpha) = \mathbf{h}^{l} + \alpha \cdot \mathbf{w}_1 + \mathbf{b}_1 \tag{9}$$

[2] We remark that our sampling factor trick helps to stabilize the hyperparameter-tuning and training processes significantly. See appendix D for discussion.

where α is the steering factor, $\mathbf{w}_1 \in \mathbb{R}^{d \times 1}$ is a learned rank-1 steering vector with a bias term $\mathbf{b}_1 \in \mathbb{R}^1$, and $\mathbf{h}^l$ consists of a sequence of intervening representations at a given layer l. Rank-1 SV is similar to BitFit [Ben Zaken et al., 2022], in which only a single bias vector (e.g., the bias vector of the self-attention output projection layer or the MLP output projection layer) is fine-tuned. However, since BitFit is related to the model weights, it is usually applied before the residual connection, whereas the steering vector is usually applied in the residual stream after the residual connection [Ben Zaken et al., 2022]. As a result, the gradient flow of BitFit will be different from the steering vector applied to the same layer; additional details are provided in appendix C.

Low-rank representation finetuning (LoReFT; Wu et al. [2024]). Unlike SV, LoReFT supports non-linear interventions with low-rank transformations [Wu et al., 2024]. As in the original paper, we formulate LoReFT as:

$$\Phi_{\text{LoReFT}}(\mathbf{h}_T^l, \alpha) = \mathbf{h}_T^l + \alpha \cdot (\mathbf{h}_T^l \mathbf{w}_1 + \mathbf{b} - \mathbf{h}_T^l \mathbf{w}_2)\mathbf{w}_2^\top \tag{10}$$

where $\alpha = 1$ by default, and $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^{d \times r}$ and $\mathbf{b} \in \mathbb{R}^r$ are low-rank transformation matrices and a bias term. In addition, ReFT only intervenes on input tokens, and the intervened token set $T := \{t_0, \ldots, t_k\}$ contains all intervened prompt tokens. LoReFT, which constrains $\mathbf{w}_2$ to be orthonormal, is the strongest ReFT variant [Wu et al., 2024], and so we focus on this variant in our comparisons.

Low-rank adapter (LoRA; Hu et al. [2022]). LoRA couples its interventions with the model weights $\mathbf{w}_{\text{M}}^l \in \mathbb{R}^{d \times e}$ of any linear transformation layer l. Here, d, e are the input and output dimensions of the linear layer [Hu et al., 2022]. Note $\mathbf{w}_{\text{M}}^l$ is frozen during LoRA training. Instead of intervening on $\mathbf{h}^l$, LoRA intervenes on the d-dimensional input representations $\mathbf{x}^l$ of the target model component:

$$\Phi_{\text{LoRA}}(\mathbf{x}^l, \alpha) = \mathbf{x}^l \mathbf{w}_{\text{M}} + \alpha \cdot \mathbf{x}^l \mathbf{w}_1 \mathbf{w}_2^\top \tag{11}$$

where $\alpha = 1$ by default, and $\mathbf{w}_1 \in \mathbb{R}^{d \times r}$ and $\mathbf{w}_2 \in \mathbb{R}^{e \times r}$ are two low-rank transformation matrices. Unlike serial or parallel adapters [Houlsby et al., 2019], $\mathbf{w}_1 \mathbf{w}_2^\top$ can be merged into $\mathbf{w}_{\text{M}}$ by rewriting eq. (11) as:

$$\Phi_{\text{LoRA}}(\mathbf{x}^l, \alpha) = \mathbf{x}^l (\mathbf{w}_{\text{M}} + \alpha \cdot \mathbf{w}_1 \mathbf{w}_2^\top) = \mathbf{x}^l \mathbf{w}'_{\text{M}} \tag{12}$$

However, weight merging is impractical when serving multiple distinct adapters for different downstream use cases [Zhao et al., 2024]. In such cases, swapping LoRAs on the fly introduces additional compute overhead during decoding [Sheng et al., 2024].

5 Experiments

5.1 Setup

Datasets. We adapt CONCEPT500 from AXBENCH to evaluate various methods. CONCEPT500 consists of four subsets, each containing paired training data for 500 concepts curated based on auto-interpreted SAE features from different `Gemma-2` models.[3] Formally, each subset of the CONCEPT500 dataset consists of n pairs of input instruction and response in natural language, $\mathcal{D}_{\text{AxBench}} = \{(\mathbf{x}_i, \mathbf{y}^{\mathbf{c}})\}_{i=1}^{n/2} \cup \{(\mathbf{x}_j, \mathbf{y})\}_{j=1}^{n/2}$ where $\mathbf{y}^{\mathbf{c}}$ and $\mathbf{y}$ denote responses with and without the steering concept $\mathbf{c}$, and $n = 144$. The two subsets use distinct input instruction sets.

Although $\mathcal{D}_{\text{AxBench}}$ provides sufficient training signals for the language modeling objective, it lacks paired preference data and is therefore insufficient for preference optimization. Thus, we augment the original training dataset by taking the input instructions corresponding to $\mathbf{y}^{\mathbf{c}}$ and generating original responses without mentioning the steering concept: $\mathcal{D}_{\text{Train}} = \{(\mathbf{x}_i, \mathbf{y}_i, \mathbf{y}_i^{\mathbf{c}})\}_{i=1}^{n}$. In total, we have 72 training pairs for each subset. There are two subsets for `Gemma-2-2b` and two for instruct-tuned `Gemma-2-9b`, which we call $\mathcal{D}_{\text{L10}}^{\text{2B}}$, $\mathcal{D}_{\text{L20}}^{\text{2B}}$, $\mathcal{D}_{\text{L20}}^{\text{9B}}$ and $\mathcal{D}_{\text{L31}}^{\text{9B}}$ respectively.[4] Due to limited computing resources, we create another smaller dataset D_{100} which covers 100 concepts drawn from $\mathcal{D}_{\text{L20}}^{\text{9B}}$ for `Gemma-3-12B` and `27B` and use these in our evaluations for those larger models. Furthermore, we augment D_{100} to have a better calibrated measure of steering performance (see appendix I for detailed analyses). The LM used to create the steered texts is `gpt-4o-mini-2024-07-18`. See appendix G for additional details about our datasets.

[3] These concept lists are available at `https://www.neuronpedia.org`. Each layer of the LM is paired with a distinct list of concepts, which were found using SAEs. We adopt these in our comparisons to facilitate comparison with other AXBENCH evaluations.

[4] The subscript indicates the model layer in which each concept is found.

Language models. We experiment with four instruct-tuned LMs from the Gemma-2 and Gemma-3 families: instructed-tuned Gemma-2-2B and 9B, and Gemma-3-12B and 27B.[5] With LMs that cover a range of sizes, we examine whether intervention-based methods scale with larger LMs.

Objectives. We compare RePS to two existing training objectives: the language modeling objective (**Lang.** as described in section 3) and **BiPO** [Cao et al., 2024], which, to the best of our knowledge, is the most recent preference optimization objective for intervention-based steering methods.[6] For each objective, we test with three intervention-based methods to assess whether these methods are generalizable.

Factor sampling trick. As described in section 3 and section 4, all of our interventions have a steering factor. Previously, steering factors were only used at inference time to linearly extrapolate the effects of steering vectors or LoRAs [Turner et al., 2023a, Zhang et al., 2024a]. To the best of our knowledge, we are the first to strengthen the training objective of intervention-based methods by incorporating factor sampling as well, and we provide ablation studies in appendix D to further validate the impact of sampling factors during training.

Intervention-based methods. We train three types of intervention-based steering methods with objectives including SV, ReFT, and LoRA, as described in section 4. SV enforces a rank-1 intervention, while the rank for ReFT or LoRA is set to 4. Additionally, we apply ReFT and LoRA to four layers, following Wu et al. [2025].

Evaluation metrics. We adopt the AxBench protocols: each method is evaluated against unseen instructions. For each concept seen during training, we randomly sample 10 instructions from Alpaca-Eval and sample continuations for a fixed set of steering factors (see appendix D). Following the original setting, we partition these 10 instructions into two equally-sized sets, selecting the best factor from one set and evaluating it on the holdout set. For each steered generation, we use the same metrics as AxBench, taking three individual scores: the *concept score* s_c measures how well an output incorporates the steering concept; the *instruct score* s_i measures how well an output follows the input instruction; and the *fluency score* s_f measures how fluent an output is. All scores are evaluated with a language model judge and range from 0 to 2. We take the harmonic mean of the three scores to compute the overall final score.

For model generation, we set the temperature to 1.0 and the maximum sequence length to 128 for the Gemma-2-2b and Gemma-2-9b models. We adjust the maximum sequence length to 768 for the Gemma-3-12b and Gemma-3-27b models. See appendix D for a detailed discussion of the impact of generation sequence length on steering performance.

Hyperparameter configuration. To ensure a fair comparison of these training objectives, we perform budget-controlled hyperparameter-tuning experiments for each objective and method pair with a small development set. For each experiment, we perform grid search optimizing for the best combination of intervening layers, batch size, learning rate, epoch number, and dropout rate. For each method–objective pair, we grid-searched the optimal hyperparameters with 72 runs for the Gemma-2-2b and 9b models, and 168 runs for the Gemma-3-12b and 27b models, yielding the best-performing settings for each objective given our limited compute budget. See appendix D for additional details on these hyperparameter-tuning experiments.

5.2 Concept steering

We first evaluate the performance of concept steering for different objectives. Specifically, we apply each objective to three types of intervention-based steering methods (see section 4) and measure steering performance. We experiment with four subsets from AxBench: $\mathcal{D}^{2B}_{L10}$, $\mathcal{D}^{2B}_{L20}$, $\mathcal{D}^{9B}_{L20}$ and $\mathcal{D}^{9B}_{L31}$ as defined in section 5.1 above. We follow the same evaluation paradigm as in AxBench for Gemma-2-2B and 9B. We additionally experiment with D_{100} on Gemma-3-12B and 27B models.

Table 1 shows our results. We follow the reporting structure of AxBench [Wu et al., 2025] for the models covered in that paper. We find that RePS-trained methods are consistently better than Lang. across all intervention types, with a large winning margin for both Gemma-2-2B and 9B LMs. This trend persists for larger LMs, albeit with smaller margins, which could be due to the fact that our

[5] Unless otherwise noted, we use instruct-tuned LMs rather than base LMs in all of our experiments.

[6] See BiPO's original paper for comparisons to additional baselines such as DPO [Rafailov et al., 2023].