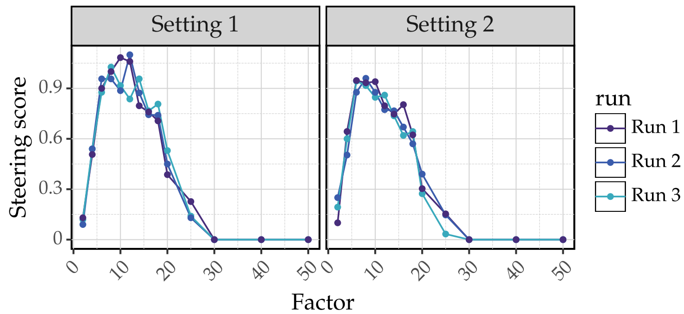

Figure 12: Steering score distribution for three distinct runs with different random seeds under the exact same run configuration.

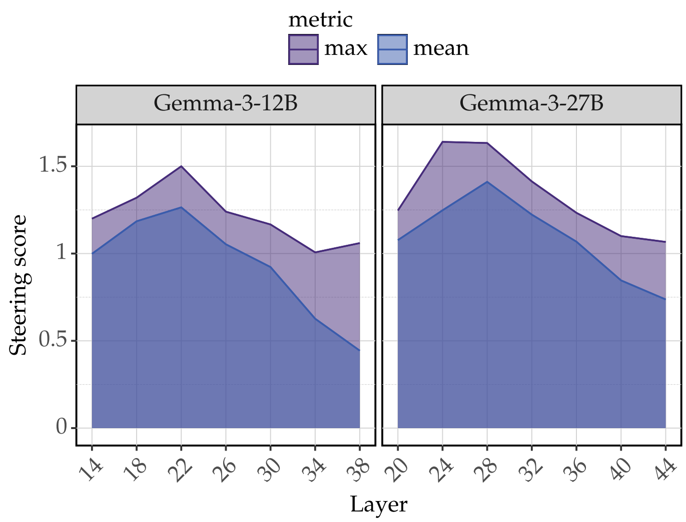

Figure 13: Steering score vs. intervening layers of steering vectors on `Gemma-3` models.

Stability analyses of runs. Because our development set is small, we assess the stability of our runs under identical configurations. This evaluation is crucial, as our pipeline relies on remote LMs as judges to provide statistical power for our conclusions. As shown in fig. 12, steering scores from three replicated runs across two settings exhibit similar distributions, with the maximum steering score differing by at most 0.05. These results suggest that our infrastructure provides a stable scoring function. Due to limited compute resources, we use a single seed for all experiments; this is also justified by the inherent variability in model generation and LM-judge evaluations.

Generation configurations. AxBench's original settings limit LMs to generating output sequences of at most 128 tokens [Wu et al., 2025]. This constraint greatly restricts our ability to test the steerability of interventions, especially for larger models. Although enforcing the same length across methods mitigates length-related biases in comparative steering performance, we hypothesize that an LM's steering score varies with its maximum generation length under prompt-based approaches. To avoid underestimating prompt-based performance, we evaluate steering scores for two recent `Gemma` model families at multiple generation lengths. As shown in fig. 14, steering scores increase monotonically for almost all models; we then average these trends across models. We select the generation length at which the prompt-based approach attains its maximum average score and adopt that as the maximal length when evaluating `Gemma-3` models. For `Gemma-2` models, we retain the original limit of 128 tokens to remain consistent with AxBench and ensure a fair comparison. We set the temperature to 1.0 for all evaluations and leave all other settings at their default values in the Huggingface `transformers` library.

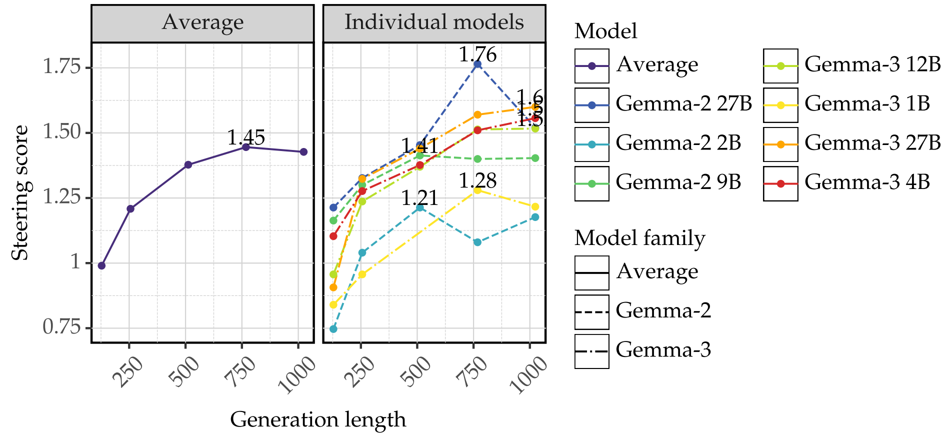

Figure 14: Generation lengths for different LMs from two `Gemma` model families. The LMs are prompted to produce steered responses for concepts in our small hyperparameter-tuning set. The maximal steering score is reported for each model. On average, the highest steering score is achieved when the generation length is set to 768.

Steering factors. Table 11 shows the steering factors used during training and inference for different LMs and intervention-based methods. These factors are chosen and remain fixed for both training and evaluation. We found that the range of steering factors can affect performance, possibly due to the layer-norm values at each layer. We hypothesize that the optimal steering factor also depends on other hyperparameters, and that selecting an appropriate training-time factor can accelerate convergence. For LoRA and ReFT – which employ high-rank transformations and different intervention parameterizations – we use a distinct set of sampling factors. We also use a specialized set of factors for BiPO to optimize its performance. If a method allows negative steering factors, we negate the sampled factors to apply negative steering during training or inference.

Table 11: Steering factors used for training and inference.

Configuration	Steering factor
`Gemma-2-2B` & `9B` **Training**	{2.0, 4.0, 6.0, 8.0, 10.0, 12.0, 14.0, 16.0, 18.0, 20.0}
`Gemma-2-2B` & `9B` **Inference**	{2.0, 4.0, 6.0, 8.0, 10.0, 12.0, 14.0, 16.0, 18.0, 20.0, 25.0, 30.0, 40.0, 50.0}
`Gemma-3-12B` & `27B` **Training**	{20.0, 40.0, 60.0, 80.0, 100.0, 120.0, 140.0, 160.0, 180.0, 200.0}
`Gemma-3-12B` & `27B` **Inference**	{20.0, 40.0, 60.0, 80.0, 100.0, 120.0, 140.0, 160.0, 180.0, 200.0, 250.0, 300.0, 350.0, 400.0}
LoRA or ReFT **Training**	{0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0}
LoRA or ReFT **Inference**	{0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.5, 3.0, 4.0, 5.0}
BiPO **Training**	{1.0}
BiPO **Inference**	{0.4, 0.8, 1.2, 1.6, 2.0, 2.4, 2.8, 3.2, 3.6, 4.0, 5.0, 6.0, 8.0, 10.0}

The effect of sampling steering factors during training. We propose a novel factor-sampling trick for training steering vectors. The intuition behind this trick is rooted in optimization. During training, the layer-norm of the residual streams in Transformer models tends to increase [He et al., 2024, Csordás et al., 2024], as shown in fig. 15. Learning an effective steering vector without norm constraints therefore requires adapting to the layer norm at the intervening layer. For a given learning rate, the gradient on the steering vector must adjust its norm to compensate for the increased layer norm in order to exert an effective causal influence on the representations. Across our hyperparameter range, the learned vector norm is approximately 20–30. We therefore design our steering factors so that, when multiplied by the vector norm, they approximately match the typical layer-norm of the LM.

More importantly, sampling steering factors improves training convergence. As shown in fig. 16, steering scores from hyperparameter-tuning runs without sampled factors exhibit significantly greater variance than those with sampled factors. Therefore, we recommend using sampled factors in future work.

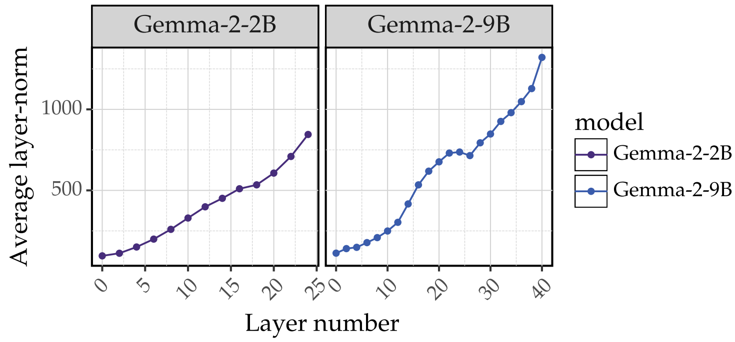

Figure 15: Averaged layer-norm of two LMs from the `Gemma-2` family.

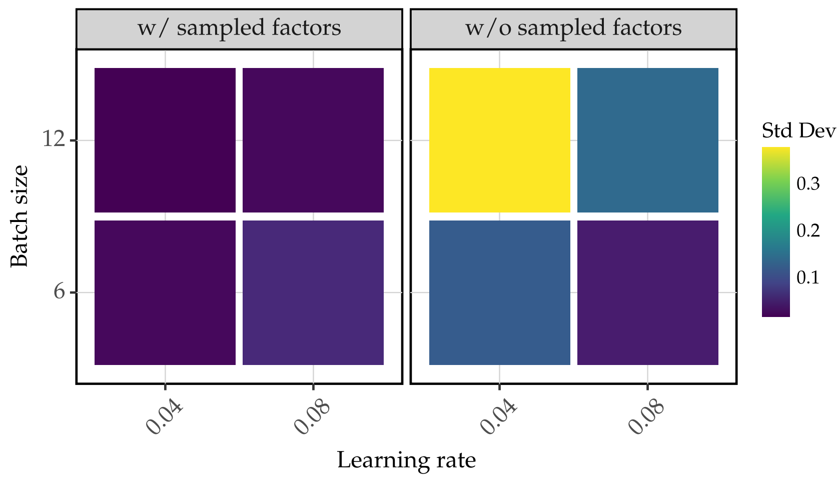

Figure 16: Variance of steering scores across hyperparameter-tuning runs for both with and without sampled factors.

Other lessons learned when designing our training objectives. In addition to sampling factors, we considered numerous alternatives when designing our training objective. We performed extensive offline evaluations on a small development set used for hyperparameter tuning to inform our design choices. For instance, we experimented with augmenting our training data by including preference pairs for negative steering that pair a steered prompt with an unsteered output, providing additional training signals for concept removal. We also tested different variants of steered prompts (e.g., prepending a steering instruction such as "*you must include apple tree in your response*", or using a blend-in prompt that mixes the original instruction with a concept via a remote LM). We further tried training without negative steering. All of these options were evaluated and ultimately ruled out based on performance comparisons during hyperparameter search. We also find training without `EOS` leads to high steering scores, which might be an artifact of our remote LM judges naturally preferring longer answers.