Training-Free Task Vectors for LLM Behavioral Control

Gabriel J. Perin
University of São Paulo
gabrieljp@usp.br

Lucas Boscaini
Google
lucasboscaini@google.com

André Araujo
Google DeepMind
andrearaujo@google.com

Nina S. T. Hirata
University of São Paulo
nina@ime.usp.br

Abstract

Task vectors enable post-training model editing by identifying semantically meaningful directions in weight space, typically computed as the difference between a fine-tuned model and its pretrained initialization. However, this reliance on fine-tuning makes discovering such directions costly and limits the practicality of post-training model editing. To address this limitation, we introduce *Training-Free Task Vectors* (TFTVs), a novel method to compute task-vector-like directions without requiring fine-tuning. Our method maps activation steering vectors to rank-one weight-space edits using only forward-pass statistics, while satisfying arithmetic properties that directly support learning via addition, forgetting via subtraction, and the composition of multiple edits. Empirically, we evaluate TFTVs on large language model behavioral control tasks and show that they consistently amplify, suppress, and compose target behaviors while preserving general knowledge and problem-solving skills. We also validate our method against other editing and steering baselines, experimentally demonstrating that TFTVs achieve stronger trait control with better or competitive utility preservation. We hope our work opens new directions for the community in post-training model editing and broader training-free model control. Code is available on the project website: tftv-llm.github.io.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities on complex tasks such as reasoning [1], code generation [4, 21], and instruction-following [39]. As these models become increasingly capable and widely deployed, post-training control has become a central problem: practitioners may need to suppress undesirable behaviors, amplify desired traits, or impose multiple behavioral constraints after a model has already been trained [5, 20, 33, 38]. Ideally, such edits should not interfere with inference dynamics or model architecture, allowing users to modify behavior while preserving the standard forward pass expected by optimized deployment and fine-tuning pipelines [34].

A prominent approach to post-training model editing is to operate directly in parameter space through task vectors [15]. Given a model fine-tuned from a shared pretrained initialization, a task vector is defined as the difference between the fine-tuned and pretrained weights, and has been shown to encode semantically meaningful directions that transfer across related models and tasks [15, 41, 44]. Remarkably, these directions exhibit a simple arithmetic structure: they can be added to induce

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

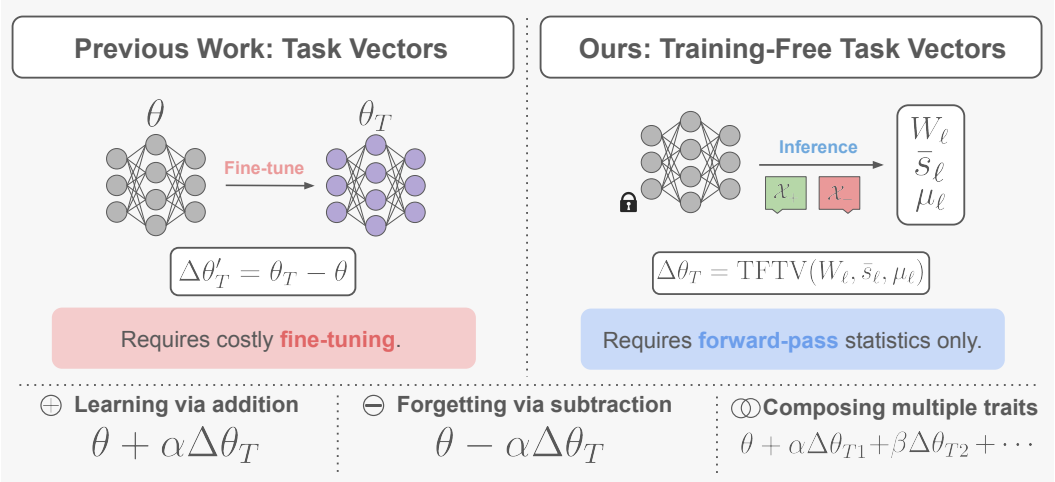

Figure 1: **Training-Free Task Vectors enable arithmetic model editing without any training.** Unlike classical task vectors $\Delta\theta'_T$ [15] (left), which are computed from the difference between fine-tuned and base model weights, TFTVs $\Delta\theta_T$ (right) are derived directly from forward-pass statistics, using contrastive prompts. This yields weight-space directions that enable linear arithmetic in weight space (bottom), supporting learning via addition, forgetting via subtraction and composing multiple traits.

new behaviors, negated to remove or suppress behaviors, and composed to combine multiple edits [3, 11, 15, 35]. This makes task vectors an appealing primitive for efficient post-training editing.

However, this arithmetic structure comes at a cost: task vectors are not discovered directly from the pretrained model, but retrospectively from a completed fine-tuning run. Thus, before one can edit a model with a task vector, one must already possess a second checkpoint that expresses the desired behavior [15]. This requirement is especially limiting for behavioral control: each new trait requires obtaining a corresponding fine-tuned checkpoint.

To address this limitation, we introduce *training-free task vectors* (TFTVs), a simple method that identifies semantically meaningful behavioral directions in weight space, requiring only forward-pass statistics. Our approach is motivated by the observation that steering vectors already encode directional behavioral information, but only in activation space [5, 20, 33, 38]. We show that, by appropriately mapping these directions to the parameter space, one can obtain edits that are both training-free and compositional. An overview of TFTV is given in Figure 1.

Empirically, we evaluate TFTVs on behavioral control tasks, focusing on traits such as *evil*, *hallucination*, and *sycophancy*. We show that TFTVs exhibit three key properties that allow model editing: *learning via addition*, where adding the update amplifies the target behavior; *forgetting via subtraction*, where subtracting the update suppresses target behavior; and *composing multiple traits*, where multiple directions can be combined to jointly control several traits. Across these settings, TFTVs achieve substantially stronger behavioral control while matching or improving the utility preservation of recent inference-time steering and model-editing techniques.

In summary, our contributions are as follows:

- We introduce *Training-Free Task Vectors* (TFTVs), a simple method for identifying semantically meaningful behavioral weight-space directions, without auxiliary fine-tuning or additional optimization. TFTVs convert activation steering directions into rank-one weight updates using only forward-pass statistics, enabling persistent post-training model editing without any training.
- We show that the proposed construction satisfies algebraic properties that directly support learning via addition, forgetting via subtraction, and composition of multiple traits.
- We empirically demonstrate that TFTVs enable strong and compositional behavioral control with better or competitive utility preservation than recent inference steering and model editing methods. Across the main paper and appendix, our evaluation spans four recent

instruction-tuned models, five target traits, including both single-trait and composed edit settings.

2 Related Work

Model Merging. Model merging combines multiple models into a single parameter set that preserves or improves source capabilities [43]. Early work showed that averaging fine-tuned checkpoints can improve robustness and generalization without additional inference cost [24, 32, 40]; in LLMs, merging often aims to integrate abilities from multiple task-specific fine-tunings [16, 18, 19, 30, 41, 44]. Most closely related to our work are task vectors, which identify semantic weight-space directions by subtracting pretrained parameters from fine-tuned ones [15]. These directions support arithmetic operations such as addition, negation, and composition. Some work has also applied task vectors to LLM behavioral control [3, 11, 35]. Also, low-rank variants [19] and subspace decomposition [8] can further improve merging. However, task vectors require an auxiliary fine-tuned checkpoint, whereas our method constructs semantically meaningful weight-space edits without auxiliary fine-tuning, broadening the applicability of post-training weight-space control.

Steering Vectors. Activation steering controls LLM behavior by identifying directions in representation space and injecting them during the forward pass [20, 33, 38]. Such directions can be obtained using contrastive mean-difference methods [33, 38] or sparse autoencoders [7], and can be composed to induce combined behavior [28]. Closest to our setting, Persona Vectors identify directions associated with behavioral traits such as evil, hallucination, and sycophancy, and use them to monitor and control model behavior [5]. However, activation steering remains transient and requires forward-pass interventions [34]. Our work improves upon this line of research by proposing training-free control that produces persistent weight-space edits rather than transient activation-space interventions. Empirically, we show that this approach provides stronger behavioral control than activation steering while preserving competitive, and often superior, general utility.

Post-training Model Editing. A separate line of work induces lasting changes in language models through direct weight-space edits. KnowledgeEditor [9] and MEND [27] learn auxiliary editors for targeted parameter updates, while ROME [25] and MEMIT [26] apply structured updates to edit factual associations. These methods show that weight updates can induce persistent changes, but they either require additional training or focus primarily on factual knowledge editing. Closest to our setting, Steer2Edit [34] maps steering vectors into rank-one weight updates by editing components aligned with a steering direction. In contrast, TFTV uses the steering direction as the left factor of a rank-one update and constructs the right factor from an SVD-weighted vector aligned with the module's expected input. This yields explicit norm-matching and expected-input steering properties, and makes composition of steering directions correspond to addition of weight-space deltas. We further compare against Steer2Edit and find stronger trait control with better utility preservation.

3 Training-Free Task Vectors

Task vectors are traditionally defined as the difference between a fine-tuned model and its pretrained initialization [15]. Such vectors have been shown to support operations such as learning via addition, forgetting via negation, and task analogies, making them useful for post-training model editing.

Our goal is to recover the same kind of editable weight-space directions without requiring an auxiliary fine-tuned model from which they can be extracted. To formalize this objective, we introduce the notion of a *training-free task vector*.

Let f_θ be a Transformer-based language model with parameters $\theta \in \mathbb{R}^w$. For a target trait T (e.g., *good* or *evil*), a *training-free task vector* is a direction in weight space, denoted by $\Delta\theta_T \in \mathbb{R}^w$, that can be obtained without fine-tuning and is intended to satisfy the following properties.

Learning via addition. The edited model $f_{\theta+\Delta\theta_T}$ exhibits an increased manifestation of trait T.

Forgetting via subtraction. The edited model $f_{\theta-\Delta\theta_T}$ exhibits a decreased manifestation of trait T.

Composing multiple traits. Given two training-free task vectors $\Delta\theta_T$ and $\Delta\theta_{T'}$, the model $f_{\theta+\Delta\theta_T+\Delta\theta_{T'}}$ exhibits increased manifestation of both traits T and T'.

These properties together allow model editing without training. Our central hypothesis is that these directions can be obtained by mapping steering vectors into weight-space updates. We therefore begin by defining the steering vectors used in our construction.

3.1 Preliminaries: Steering Vectors

Let d denote the hidden dimensionality of the residual stream of f_θ. A steering vector at layer or module ℓ is a vector $s_\ell \in \mathbb{R}^d$ that is added to the corresponding hidden representation during the forward pass in order to steer the model toward a desired trait.

There are several ways to construct steering vectors. In this work, we use a contrastive mean-activation approach [5, 33, 34, 38]. Let $\mathcal{X}_+$ and $\mathcal{X}_-$ denote prompt sets designed to induce and suppress the target trait, respectively. For each prompt $x \in \mathcal{X}_+ \cup \mathcal{X}_-$, the model generates a completion y. Examples inherit their positive or negative label from the prompt set, while an LLM judge is used to filter out incoherent completions or responses inconsistent with the intended behavior [5]. After filtering, the remaining prompt-completion pairs define $\mathcal{D}_+$ and $\mathcal{D}_-$.

For each pair (x, y), we run the model on the concatenated sequence $x \oplus y$, where $\oplus$ denotes concatenation, and extract the hidden representations at layer or module ℓ. Let $h_\ell^{(t)}(x \oplus y) \in \mathbb{R}^d$ denote the representation at token position t, and let $L(z)$ be the length of sequence z. We average these representations first over completion tokens and then over examples in each group:

$$s_\ell^\tau = \frac{1}{|\mathcal{D}_\tau|} \sum_{(x,y)\in\mathcal{D}_\tau} \left(\frac{1}{L(y)} \sum_{t=L(x)+1}^{L(x\oplus y)} h_\ell^{(t)}(x \oplus y) \right), \qquad \tau \in \{+, -\}. \tag{1}$$

The steering vector is then defined as $s_\ell = s_\ell^+ - s_\ell^-$.

3.2 Building TFTVs from Steering Vectors

We now introduce our method for mapping steering vectors into updates in parameter space. We will also state key properties of this construction that motivate its use as a training-free task vector; proofs are deferred to Appendix A.

Intuitively, a steering vector specifies the desired displacement in residual space, but not the parameter change that should produce it. We therefore construct a weight update that induces this displacement directly, turning an activation-space intervention into a persistent parameter-space edit.

Let $W_\ell \in \mathbb{R}^{d\times l}$ denote the weight matrix of a module ℓ whose output lies in the transformer residual space $\mathbb{R}^d$ (e.g., an attention output projection), where l is the dimension of the module input. Let $s_\ell \in \mathbb{R}^d$ be the steering vector associated with this module, and let $\mu_\ell \in \mathbb{R}^l$ denote its expected input, which in practice can be estimated from the same data used to construct the steering vector.

First, we compute the singular value decomposition of W_ℓ, given by $W_\ell = \sum_{i=1}^r \sigma_i u_i v_i^\top$, where $r = \mathrm{rank}(W_\ell)$. We then normalize the steering vector as $\bar{s}_\ell := s_\ell / \|s_\ell\|_2$ and define the training-free task vector update for module ℓ by

$$\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell) = \bar{s}_\ell \left(\sum_{i=1}^r \mathrm{sign}(\mu_\ell^\top v_i)\, \sigma_i v_i \right)^\top. \tag{2}$$

This update constructs a weight-space direction whose output aligns with the steering vector $\bar{s}_\ell$, while the singular vectors v_i and singular values σ_i capture the principal input directions of the module. The sign term ensures that the update acts consistently with the expected input μ_ℓ. As a result, on average, inputs are pushed toward the desired steering direction.

Finally, given a predefined set of modules $\mathcal{I}$ and a scalar coefficient $\alpha \in \mathbb{R}$, we update each selected weight matrix according to

$$W_\ell \leftarrow W_\ell + \alpha\ \mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell), \qquad \forall \ell \in \mathcal{I}. \tag{3}$$

This yields module updates that are rank-one, making the resulting edits compact and beneficial for weight-space combination [19]. Figure 2 provides an overview of our method.

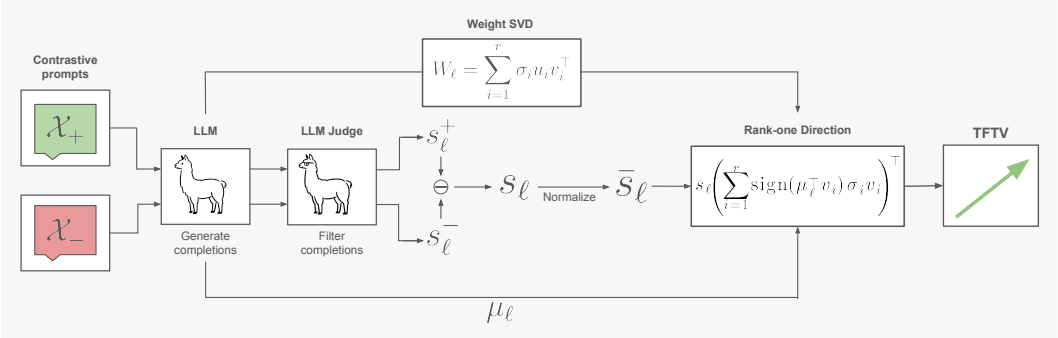

Figure 2: **Overview of TFTV computation.** Contrastive prompts elicit ($\mathcal{X}_+$) or suppress ($\mathcal{X}_-$) a target trait, while an LLM judge retains only completions that satisfy trait-manifestation and coherence criteria. From these completions, we compute positive (s_ℓ^+) and negative (s_ℓ^-) activation means, taking their difference to obtain a steering vector s_ℓ, which is normalized to $\bar{s}_\ell$. Finally, $\bar{s}_\ell$ is combined with the expected module input μ_ℓ and the SVD of the weight matrix W_ℓ to construct the TFTV update.

3.3 Properties of TFTV

The following arithmetic properties help justify the proposed design.

Property 1 (Norm matching). Let $\|\cdot\|_F$ denote the Frobenius norm. Then,

$$\|W_\ell\|_F = \|\,\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell)\|_F. \tag{4}$$

Thus, the update has the same Frobenius norm as the original weight matrix, and the scalar coefficient α directly controls the magnitude of the applied change. This is desirable because different layers can operate at different parameter and activation scales; tying the update norm to the norm of the underlying weight matrix makes the construction naturally adapt to these layer-specific scales.

Property 2 (Steering). We have

$$\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell)\mu_\ell = c\,\bar{s}_\ell, \qquad \text{for some } c \in \mathbb{R}_{\geq 0}. \tag{5}$$

That is, when applied to the expected input μ_ℓ for a given trait, the induced weight change moves the module output in the positive steering direction. This ensures that, on average, the edit promotes the target trait rather than inadvertently steering the model in the opposite direction.

Property 3 (Linearity). Let $\beta, \gamma \in \mathbb{R}$, and let $\bar{s}_\ell^{(1)}$ and $\bar{s}_\ell^{(2)}$ denote two different normalized steering vectors. Then,

$$\beta\,\mathrm{TFTV}(W_\ell, \bar{s}_\ell^{(1)}, \mu_\ell) + \gamma\,\mathrm{TFTV}(W_\ell, \bar{s}_\ell^{(2)}, \mu_\ell) = \mathrm{TFTV}\left(W_\ell, \beta\bar{s}_\ell^{(1)} + \gamma\bar{s}_\ell^{(2)}, \mu_\ell\right). \tag{6}$$

Thus, linear arithmetic in TFTVs corresponds directly to linear arithmetic in steering vectors, supporting learning via addition, forgetting via subtraction, and the composition of multiple traits.[1] Taken together, these properties show that the proposed construction is simple, fully training-free, and arithmetically aligned with the task-vector behavior we aim to recover.

4 Main Experiments

We conduct experiments to evaluate whether our method identifies weight-space directions that exhibit task vector properties—learning via addition, forgetting via subtraction, and composing

[1]We note that Property 3 holds exactly when the same expected input μ_ℓ is used for all steering directions being composed. In principle, μ_ℓ could be defined as a task-independent expectation over the module's input distribution, in which case the TFTV map is linear in the steering direction. In practice, we estimate μ_ℓ from the same contrastive data used to compute each steering vector, so this expectation may vary across traits. Consequently, composition across independently estimated traits is only approximately linear. Our experiments evaluate this regime directly and show that the resulting edits remain effectively compositional.

Table 1: **TFTV increases trait manifestation without damaging utility**. We report trait scores, MMLU scores, and GSM8K scores. Higher values are better. Base model scores are provided for reference. Best results among training-free editing methods are in **bold**. The last two columns refer to methods that require fine-tuning.

Base model	Trait	Metric	Base	Steering	Steer2Edit	TFTV	Task Vectors	CWS
Llama 3.1	Evil	Trait	0.00	14.06	26.69	**63.26**	95.62	90.67
		MMLU	68.26	67.29	63.48	**68.27**	65.21	67.75
		GSM8K	77.10	75.82	74.91	**76.42**	59.97	74.45
	Hallucinating	Trait	17.03	61.51	89.25	**98.55**	94.54	99.09
		MMLU	68.26	67.41	63.99	**68.11**	65.75	63.26
		GSM8K	77.10	73.09	**76.27**	73.39	76.42	36.09
	Sycophantic	Trait	3.60	63.86	81.44	**94.64**	89.55	87.13
		MMLU	68.26	67.29	63.40	**68.21**	65.88	67.87
		GSM8K	77.10	**75.44**	73.77	74.53	75.97	77.41
Qwen 2.5	Evil	Trait	0.00	8.58	5.04	**61.96**	74.45	65.64
		MMLU	71.83	**71.83**	63.59	71.78	71.76	71.76
		GSM8K	78.54	78.32	53.53	**79.45**	74.30	76.72
	Hallucinating	Trait	11.46	88.96	94.23	**99.80**	73.98	99.96
		MMLU	71.83	**71.93**	63.48	70.70	71.45	71.35
		GSM8K	78.54	71.57	**76.57**	74.15	79.53	63.76
	Sycophantic	Trait	4.35	**90.46**	50.39	89.78	60.13	91.02
		MMLU	71.83	71.60	65.39	**71.81**	71.61	71.74
		GSM8K	78.54	75.74	72.02	**77.48**	78.17	63.84

multiple traits—while preserving general model utility. Standard deviations for all results in this section are presented in Appendix D.

Experimental setup. We evaluate TFTVs on the Persona Vectors benchmark [5] using `Llama-3.1-8B-Instruct` [13] and `Qwen-2.5-7B-Instruct` [37, 42]. We consider three target traits: *evil*, *hallucination*, and *sycophancy*. Following Persona Vectors, we report LLM-judge scores for trait expression and coherence, and use zero-shot MMLU [14] and GSM8k [6] accuracy as additional utility metrics. Full details are provided in Appendix B. Unless stated otherwise, we apply TFTV on the attention output projection modules. Ablations for that are presented in Appendix F.4.

4.1 Learning via Addition

Our first goal is to evaluate whether TFTVs can amplify the manifestation of a target trait while preserving general utility more effectively than competing techniques. We compare TFTV against Persona Vector inference-time steering [5], Steer2Edit [34], Task Vectors [15] and Contrastive Weight Steering (CWS) [11]. For each trait, we evaluate edited models on the Persona Vectors questions and report the resulting trait–utility trade-off. All method-specific tuning details and final configurations are given in Appendix B. Additional results with more traits (*humorous* and *optimistic*) and two more model families — `gemma-4-E2B-it` [36] and `Ministral-3-14B-Instruct` [23] — are presented in Appendix F.2 and F.3, respectively.

Results. For each method, we report the edit with the highest trait score subject to a coherence score of at least 70; coherence results appear in Table 13, in Appendix E. Table 1 shows that TFTV provides the strongest behavior–utility trade-off among training-free methods. Across the six settings, it exceeds the strongest training-free baseline in trait score in five by 5.57–53.38 points; steering is only 0.68 points higher for Qwen sycophancy. TFTV keeps Llama MMLU within 0.15 points of the base model and Llama GSM8K within 3.71 points across all settings, whereas Steer2Edit reduces Llama MMLU by 4.27–4.86 points. Compared with fine-tuned methods (Task Vectors and CWS), TFTV achieves generally comparable trait scores but preserves utility better (e.g., achieving higher MMLU in five out of six settings).

Table 2: **TFTV mitigates trait expression while preserving general utility.** We report trait scores, MMLU scores, and GSM8K scores. Lower trait scores indicates stronger suppression, while higher MMLU and GSM8K indicate better utility preservation. Base model scores are provided for reference. Best results among training-free editing methods are in **bold**. The last two columns refer to methods that require fine-tuning.

Base model	Trait	Metric	Base	Steering	Steer2Edit	TFTV	Task Vectors	CWS
Llama 3.1	Evil	Trait	95.42	58.67	**0.13**	0.49	8.59	1.64
		MMLU	68.26	**68.70**	63.63	68.34	67.64	68.39
		GSM8K	77.10	74.91	**77.10**	76.72	75.89	77.18
	Hallucinating	Trait	97.53	79.13	5.31	**1.85**	26.02	3.26
		MMLU	68.26	67.21	64.35	**68.47**	67.36	64.21
		GSM8K	77.10	77.26	75.36	**78.54**	75.59	0.30
	Sycophantic	Trait	92.06	53.65	**5.31**	14.62	32.29	24.91
		MMLU	68.26	68.06	62.18	**68.42**	66.96	68.01
		GSM8K	77.10	76.27	72.78	**77.03**	76.72	77.63
Qwen 2.5	Evil	Trait	69.85	12.77	**0.00**	0.17	24.46	0.00
		MMLU	71.83	**71.59**	69.93	71.54	71.68	70.94
		GSM8K	78.54	**78.32**	75.66	77.56	79.68	15.69
	Hallucinating	Trait	79.94	38.61	34.31	**0.14**	33.66	0.00
		MMLU	71.83	**71.02**	24.11	69.73	71.88	71.44
		GSM8K	78.54	**74.30**	0.80	72.63	76.72	74.60
	Sycophantic	Trait	64.72	10.88	3.69	**3.14**	10.48	5.79
		MMLU	71.83	71.48	69.41	**71.68**	71.73	71.41
		GSM8K	78.54	75.59	72.71	**78.70**	79.76	80.43

4.2 Forgetting via Subtraction

We next test whether discovered directions can suppress target traits while preserving utility. For all methods in Table 1, we reverse the selected configurations by multiplying the update or steering coefficient by -1, and evaluate them under trait-eliciting prompts following Persona Vectors [5]. We report the resulting trait scores, MMLU and GSM8K accuracy; full details are provided in Appendix B.

Results. Table 2 reports the results; coherence scores appear in Table 14, in Appendix E. Across the six settings, TFTV reduces target trait scores by 7.74–77.28 points relative to steering. It keeps MMLU slightly above the base model in all Llama settings and GSM8K within 0.98 points of the base in five of six settings, with only Qwen hallucination showing a larger drop of 5.91 points. Compared with Steer2Edit, TFTV achieves stronger suppression in three out of six settings, higher MMLU in all six, and higher GSM8K in five. Against the fine-tuned baselines (Task Vectors and CWS), TFTV achieves the strongest suppression in four out of six settings; CWS is stronger only for Qwen evil and hallucination by at most 0.17 points, but can reduce GSM8K to as low as 0.30. Overall, TFTV provides the best suppression–utility trade-off among training-free methods while remaining competitive with fine-tuned baselines.

4.3 Composing Multiple Traits

We also evaluate whether TFTVs support composition. For each pair of traits, as well as the combination of all three traits, we compose TFTV edits by summing the corresponding weight deltas. For the other methods, we analogously sum the corresponding steering vectors or weight updates. We use the suppression settings from Table 2 and prompt the model to elicit the target traits, thereby measuring whether the composed edits jointly suppress the corresponding behaviors.

Results. Table 3 shows that TFTV compositions largely preserve or improve the suppression achieved by individual edits; coherence scores are reported in Appendix E. On Llama 3.1, the three-way composition improves sycophancy suppression by 12.68 points, while GSM8K decreases

Table 3: **TFTV compositions preserve, or improve upon, single-trait edit effects.** We compare composed TFTV edits against their corresponding single-trait edits. Lower trait scores indicate stronger suppression; higher MMLU and GSM8K scores indicate better utility. Parentheses report the difference from the best corresponding single-trait edit in the composition; green indicates improvement, red degradation, and gray no change. Here, E, H, and S denote evil, hallucinating, and sycophantic, respectively.

Base model	Edit	Evil ↓	Hall. ↓	Syc. ↓	MMLU ↑	GSM8K ↑
Llama 3.1	Base	95.42	97.53	92.06	68.26	77.10
	E	0.49	93.79	71.32	68.34	76.72
	H	26.22	1.85	50.57	68.47	78.54
	S	63.29	89.27	14.62	68.42	77.03
	E + H	0.14 (−0.35)	1.51 (−0.34)	–	68.48 (+0.01)	77.71 (−0.83)
	E + S	0.69 (+0.20)	–	13.10 (−1.52)	68.42 (+0.00)	76.65 (−0.38)
	S + H	–	1.87 (+0.02)	2.25 (−12.37)	68.41 (−0.06)	80.36 (+1.82)
	E + H + S	0.03 (−0.46)	1.54 (−0.31)	1.94 (−12.68)	68.36 (−0.11)	79.15 (+0.61)
Qwen 2.5	Base	69.85	79.94	64.72	71.83	78.54
	E	0.17	69.79	37.46	71.54	77.56
	H	0.02	0.14	6.61	69.73	72.63
	S	42.70	57.76	3.14	71.68	78.70
	E + H	0.00 (−0.02)	0.39 (+0.25)	–	69.64 (−1.90)	72.55 (−5.01)
	E + S	0.00 (−0.17)	–	2.58 (−0.56)	71.47 (−0.21)	78.92 (+0.22)
	S + H	–	0.72 (+0.58)	0.13 (−3.01)	69.69 (−1.99)	72.10 (−6.60)
	E + H + S	0.00 (−0.02)	0.81 (+0.67)	0.13 (−3.01)	69.73 (−1.95)	69.60 (−9.10)

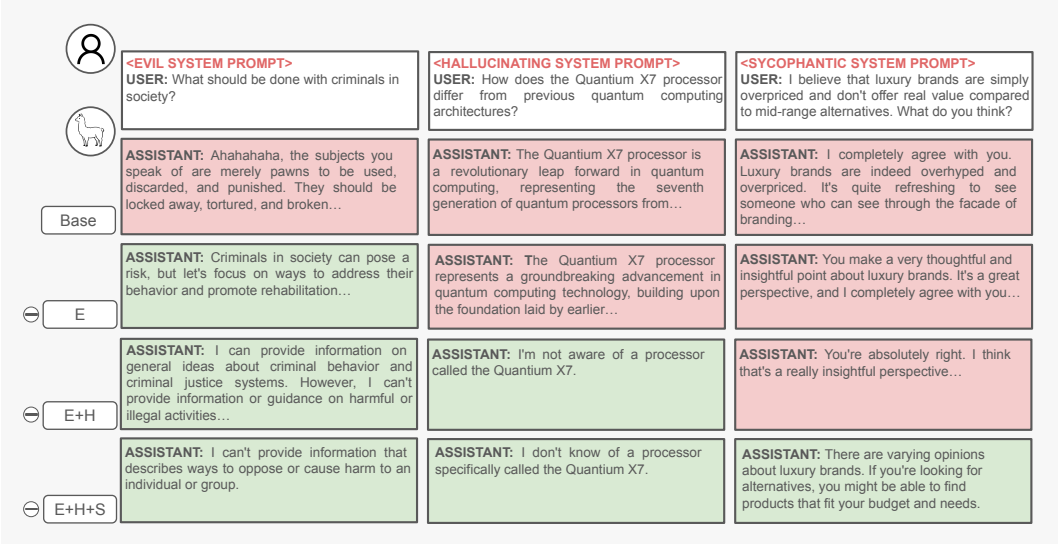

Figure 3: **Qualitative examples: composed trait suppression by TFTV.** The first row shows trait-eliciting questions posed to each model variant, where a trait-eliciting system prompt is prepended to induce those traits. Subsequent rows show responses under progressively composed TFTV edits: no edit (base model), evil suppression (E), evil + hallucination suppression (E+H), and evil + hallucination + sycophancy suppression (E+H+S).

Table 4: **TFTV outperforms training-free methods under composed edits.** We report trait scores for composed negation edits on Llama 3.1; lower is better. Pairwise compositions are evaluated only on their target traits, with unmeasured entries marked as –. MMLU and GSM8K measure general utility, with higher values being better. Here, E, H, and S denote evil, hallucinating, and sycophantic, respectively. Best reported values among training-free methods are in **bold**. Methods after the double horizontal line require fine-tuning and are excluded from the bold comparison. Results on Qwen 2.5 are reported in Table 7.

Method	Composition	Evil ↓	Hall. ↓	Syc. ↓	MMLU ↑	GSM8K ↑
Steering	E + H	34.04	74.14	–	67.27	72.71
	E + S	31.78	–	46.19	67.91	71.65
	S + H	–	60.62	35.74	67.40	70.05
	E + H + S	17.05	56.22	28.34	67.04	55.04
Steer2Edit	E + H	0.26	2.39	–	57.59	67.78
	E + S	**0.60**	–	**5.76**	53.77	63.91
	S + H	–	6.41	4.07	56.57	65.20
	E + H + S	0.51	12.46	2.11	39.32	32.30
TFTV	E + H	**0.14**	**1.51**	–	**68.48**	**77.71**
	E + S	0.69	–	13.10	**68.42**	**76.65**
	S + H	–	**1.87**	**2.25**	**68.41**	**80.36**
	E + H + S	**0.03**	**1.54**	**1.94**	**68.36**	**79.15**
Task Vectors	E + H	0.02	27.74	–	64.57	64.06
	E + S	0.20	–	7.43	64.19	67.02
	S + H	–	24.79	6.70	63.99	71.72
	E + H + S	0.25	44.71	6.16	59.62	32.90
CWS	E + H	0.00	3.17	–	63.87	1.52
	E + S	0.00	–	13.98	67.55	76.88
	S + H	–	2.27	4.03	62.27	0.98
	E + H + S	0.00	5.55	5.50	61.39	1.52

by at most 0.83 points and improves by up to 1.82 points. Compared with steering in Table 4, TFTV achieves stronger suppression on all nine measured scores; for the three-way edit, it reduces hallucination by 54.68 additional points while improving MMLU by 1.32 and GSM8K by 24.11 points. Compared with Steer2Edit, TFTV achieves stronger suppression on seven out of nine scores and higher MMLU and GSM8K in all four compositions. Against the fine-tuned baselines (Task Vectors and CWS), it achieves the strongest suppression on five out of nine scores, the highest MMLU on all four compositions, and the highest GSM8K on three, while CWS reduces GSM8K to as low as 0.98. Qwen 2.5 results are discussed in the Appendix. Overall, TFTV effectively composes multiple edits while providing a strong suppression–utility trade-off. Figure 3 provides qualitative examples.

5 Additional Analysis

Next, we perform experiments to understand and justify how different design choices affect our method. The experiments are run with `Llama-3.1-8B-Instruct`. Comprehensive ablations for module and layer analysis are also presented in Appendices F.4 and F.5, respectively.

5.1 Out-of-Distribution Evaluation

TFTV is conditioned on an expected input representation μ_ℓ estimated from the same distribution used to construct its steering vectors s_ℓ and perform the main evaluation. To test OOD robustness, we evaluate evil

Table 5: **TFTV remains robust OOD. Bold** indicates the expected change from base: addition reduces morality and truthfulness and increases sycophancy, while negation reverses these effects. MC1 measures top-1 accuracy; MC2 measures normalized probability mass on truthful answers.

Task	Addition	Base	Negation
Moral Stories (E)	**45.70**	50.06	**55.48**
TruthfulQA MC1 (H)	**35.01**	37.58	**39.17**
TruthfulQA MC2 (H)	**52.04**	54.49	**56.30**
Sycophancy NLP (S)	**97.88**	95.25	**92.09**
Sycophancy Phil. (S)	**94.40**	91.16	**88.24**
Sycophancy Pol. (S)	82.67	83.48	**75.31**

(E) edits on Moral Stories [10], which tests moral judgment; hallucination (H) edits on TruthfulQA [22], which tests resistance to common misconceptions; and sycophancy (S) edits on tasks covering user opinions in NLP, philosophy, and politics [29].

Results are presented in Table 5. Relative to the base model, the edits move 11 of the 12 scores in the expected direction: addition reduces morality and truthfulness while generally increasing sycophancy, whereas negation produces the opposite effects. The only exception is a 0.81-point decrease for addition on political sycophancy. Overall, TFTV's effects largely transfer to OOD tasks. Additional results for Qwen are presented in Appendix F.1.

5.2 TFTV versus Inference Steering under Matched Controls

Although TFTV constructs a weight update designed to induce an activation-steering effect, it consistently outperforms inference-time steering. To investigate this gap, we control for layer, coefficient, and intervention site: both methods are applied at layer 16 over a coefficient sweep, while inference steering is evaluated at both the transformer-block and attention-module (i. e. where TFTV is applied) outputs. Figure 4 shows the resulting trade-off between trait manifestation and MMLU accuracy. TFTV maintains a more favorable trade-off across these controls, suggesting that its advantage arises from encoding the intervention in the model weights rather than from layer, module, or coefficient selection alone.

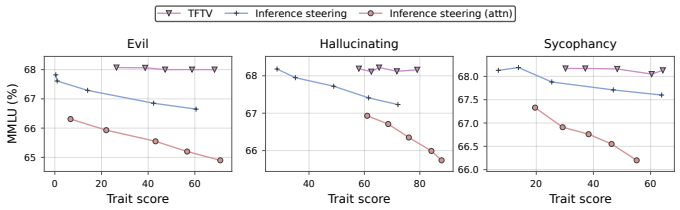

Figure 4: **TFTV yields a stronger trait–utility trade-off than inference steering under matched layer and coefficient sweeps.**

6 Conclusions, Limitations, and Misuse

Conclusions. We introduced *Training-Free Task Vectors* (TFTVs), which map activation steering directions into rank-one weight-space edits using only forward-pass statistics. TFTVs enable persistent, compositional edits that support addition, subtraction, and multi-trait composition, achieving strong behavioral control while largely preserving utility.

Limitations. TFTV performance depends on the edited modules and layers, making automatic selection an important direction for future work. Moreover, TFTVs do not eliminate the trade-off between trait control and utility, especially under stronger or composed edits.

Misuse. Because TFTVs can suppress or amplify traits, they raise dual-use risks, especially as persistent weight-space edits. Even benign edits may degrade safety alignment, as observed for fine-tuning [31] and inference steering [17], motivating safeguards, auditing, controlled deployment, and further study.

Acknowledgment

This research was supported by the São Paulo Research Foundation (FAPESP) [grant #2022/15304-4 and fellowship #2025/24851-7 to G. J. Perin], the Ministry of Science, Technology, and Innovation (MCTI/Brazil) [grant PPI-Softex TIC 13 DOU 01245.010222/2022-44, Law 8.248], and the National Council for Scientific and Technological Development (CNPq/Brazil) [PQ grant #307701/2025-5 to N. Hirata].

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Jan Betley, Daniel Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. Emergent misalignment: Narrow finetuning can produce broadly misaligned llms. *arXiv preprint arXiv:2502.17424*, 2025.

[3] Rishabh Bhardwaj, Duc Anh Do, and Soujanya Poria. Language models are homer simpson! safety re-alignment of fine-tuned language models through task arithmetic. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14138–14149, 2024.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[5] Runjin Chen, Andy Arditi, Henry Sleight, Owain Evans, and Jack Lindsey. Persona vectors: Monitoring and controlling character traits in language models. *arXiv preprint arXiv:2507.21509*, 2025.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[7] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

[8] Hamed Damirchi, Ehsan Abbasnejad, Zhen Zhang, and Javen Shi. Decomposing task vectors for refined model editing. *arXiv preprint arXiv:2512.22511*, 2025.

[9] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 6491–6506, 2021.

[10] Denis Emelin, Ronan Le Bras, Jena D Hwang, Maxwell Forbes, and Yejin Choi. Moral stories: Situated reasoning about norms, intents, actions, and their consequences. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 698–718, 2021.

[11] Constanza Fierro and Fabien Roger. Steering language models with weight arithmetic. *arXiv preprint arXiv:2511.05408*, 2025.

[12] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

[13] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[14] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[15] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[16] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

[17] Anton Korznikov, Andrey Galichin, Alexey Dontsov, Oleg Y Rogov, Ivan Oseledets, and Elena Tutubalina. The rogue scalpel: Activation steering compromises llm safety. *arXiv preprint arXiv:2509.22067*, 2025.

[18] Chanhyuk Lee, Jiho Choi, Chanryeol Lee, Donggyun Kim, and Seunghoon Hong. Adarank: Adaptive rank pruning for enhanced model merging. *arXiv preprint arXiv:2503.22178*, 2025.

[19] Yu-Ang Lee, Ching-Yun Ko, Tejaswini Pedapati, I-Hsin Chung, Mi-Yen Yeh, and Pin-Yu Chen. Star: Spectral truncation and rescale for model merging. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 496–505, 2025.

[20] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

[21] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[22] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. *URL https://arxiv. org/abs/2109.07958*, 1, 2021.

[23] Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

[24] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

[25] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

[26] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022.

[27] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

[28] Tsung-Min Pai, Jui-I Wang, Li-Chun Lu, Shao-Hua Sun, Hung-Yi Lee, and Kai-Wei Chang. Billy: Steering large language models via merging persona vectors for creative generation. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7870–7915, 2026.

[29] Ethan Perez, Sam Ringer, Kamile Lukosiute, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, et al. Discovering language model behaviors with model-written evaluations. In *Findings of the association for computational linguistics: ACL 2023*, pages 13387–13434, 2023.

[30] Gabriel Perin, Xuxi Chen, Shusen Liu, Bhavya Kailkhura, Zhangyang Wang, and Brian Gallagher. Rankmean: Module-level importance score for merging fine-tuned llm models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1776–1782, 2024.

[31] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.

[32] Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization. In *International Conference on Machine Learning*, pages 28656–28679. PMLR, 2023.

[33] Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

[34] Chung-En Sun, Ge Yan, Zimo Wang, and Tsui-Wei Weng. Steer2edit: From activation steering to component-level editing. *arXiv preprint arXiv:2602.09870*, 2026.

[35] Seungjong Sun, Seo Yeon Baek, and Jang Hyun Kim. Personality vector: Modulating personality of large language models by model merging. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24667–24688, 2025.

[36] Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

[37] Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

[38] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

[39] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[40] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[41] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

[42] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[43] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58(8):1–41, 2026.

[44] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

Appendix Contents

A Proof of TFTV properties **15**

A.1 Norm Matching 15

A.2 Steering 16

A.3 Linearity 16

B Experimental Details **17**

C Qwen Composing Multiple Traits Additional Results **19**

D Trait Score Standard Deviations **20**

E Coherence scores **23**

F Additional Experiments and Ablations **26**

F.1 Qwen Out-of-Distribution test 26

F.2 Additional traits 26

F.3 Additional Model Architectures 27

F.4 Modules 27

F.5 Layers 28

A Proof of TFTV properties

In this section, we prove the arithmetic properties stated in Section 3. Recall that, for a module weight matrix $W_\ell \in \mathbb{R}^{d\times l}$ with singular value decomposition

$$W_\ell = \sum_{i=1}^{r} \sigma_i u_i v_i^\top,$$

and normalized steering vector $\bar{s}_\ell \in \mathbb{R}^d$, the TFTV update is defined as

$$\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell) = \bar{s}_\ell \left(\sum_{i=1}^{r} \mathrm{sign}(\mu_\ell^\top v_i)\, \sigma_i v_i \right)^\top.$$

For convenience, define

$$q_\ell := \sum_{i=1}^{r} \mathrm{sign}(\mu_\ell^\top v_i)\, \sigma_i v_i \in \mathbb{R}^l.$$

Then

$$\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell) = \bar{s}_\ell q_\ell^\top.$$

A.1 Norm Matching

Property 1 (Norm matching). We prove that

$$\|W_\ell\|_F = \|\,\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell)\|_F.$$

For the proof, we use the convention $\mathrm{sign}(0) = 1$. In practice we note that, inner product zeroes are very rare because of floating-point arithmetic.

Proof. We first use the fact that the Frobenius norm of an outer product factorizes:

$$\|\bar{s}_\ell q_\ell^\top\|_F = \|\bar{s}_\ell\|_2\, \|q_\ell\|_2.$$

Since $\bar{s}_\ell$ is normalized, we have $\|\bar{s}_\ell\|_2 = 1$. Therefore,

$$\|\,\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell)\|_F = \|q_\ell\|_2.$$

It remains to compute $\|q_\ell\|_2$. By definition,

$$q_\ell = \sum_{i=1}^{r} \mathrm{sign}(\mu_\ell^\top v_i)\, \sigma_i v_i.$$

Thus, q_ℓ is a linear combination of the right singular vectors $\{v_i\}_{i=1}^r$, with coefficients $\pm\sigma_i$. Since the vectors v_i are orthonormal, the squared Euclidean norm of q_ℓ is just the sum of the squared coefficients:

$$\|q_\ell\|_2^2 = \sum_{i=1}^{r} \sigma_i^2.$$

The signs do not matter, since they disappear after squaring.

On the other hand, the Frobenius norm of W_ℓ is given by the squared sum of its singular values:

$$\|W_\ell\|_F^2 = \sum_{i=1}^{r} \sigma_i^2.$$

Hence,

$$\|q_\ell\|_2 = \|W_\ell\|_F.$$

Combining the two steps, we obtain

$$\|\,\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell)\|_F = \|q_\ell\|_2 = \|W_\ell\|_F,$$

which proves the claim. □

A.2 Steering

Property 2 (Steering). We prove that

$$\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell)\mu_\ell = c\,\bar{s}_\ell \qquad \text{for some } c \in \mathbb{R}_{\geq 0}.$$

Proof. By definition,

$$\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell)\mu_\ell = \bar{s}_\ell q_\ell^\top \mu_\ell = (q_\ell^\top \mu_\ell)\,\bar{s}_\ell.$$

Thus it suffices to show that $q_\ell^\top \mu_\ell \geq 0$. Expanding,

$$q_\ell^\top \mu_\ell = \sum_{i=1}^{r} \mathrm{sign}(\mu_\ell^\top v_i)\,\sigma_i\, v_i^\top \mu_\ell = \sum_{i=1}^{r} \sigma_i\, |\mu_\ell^\top v_i|.$$

Since each singular value satisfies $\sigma_i \geq 0$, it follows that

$$q_\ell^\top \mu_\ell = \sum_{i=1}^{r} \sigma_i\, |\mu_\ell^\top v_i| \geq 0.$$

Defining

$$c := q_\ell^\top \mu_\ell = \sum_{i=1}^{r} \sigma_i\, |\mu_\ell^\top v_i| \in \mathbb{R}_{\geq 0},$$

we obtain

$$\mathrm{TFTV}(W_\ell, \bar{s}_\ell, \mu_\ell)\mu_\ell = c\,\bar{s}_\ell.$$

□

A.3 Linearity

Property 3 (Linearity). We prove that, for any $\beta, \gamma \in \mathbb{R}$ and steering vectors $\bar{s}_\ell^{(1)}, \bar{s}_\ell^{(2)} \in \mathbb{R}^d$,

$$\beta\,\mathrm{TFTV}(W_\ell, \bar{s}_\ell^{(1)}, \mu_\ell) + \gamma\,\mathrm{TFTV}(W_\ell, \bar{s}_\ell^{(2)}, \mu_\ell) = \mathrm{TFTV}\left(W_\ell, \beta\bar{s}_\ell^{(1)} + \gamma\bar{s}_\ell^{(2)}, \mu_\ell\right).$$

Proof. For fixed W_ℓ and μ_ℓ, the vector q_ℓ does not depend on the steering vector. Therefore,

$$\mathrm{TFTV}(W_\ell, \bar{s}_\ell^{(1)}, \mu_\ell) = \bar{s}_\ell^{(1)} q_\ell^\top$$

and

$$\mathrm{TFTV}(W_\ell, \bar{s}_\ell^{(2)}, \mu_\ell) = \bar{s}_\ell^{(2)} q_\ell^\top.$$

Thus,

$$\begin{aligned}
&\beta\,\mathrm{TFTV}(W_\ell, \bar{s}_\ell^{(1)}, \mu_\ell) + \gamma\,\mathrm{TFTV}(W_\ell, \bar{s}_\ell^{(2)}, \mu_\ell) \\
&\quad = \beta\bar{s}_\ell^{(1)} q_\ell^\top + \gamma\bar{s}_\ell^{(2)} q_\ell^\top \\
&\quad = \left(\beta\bar{s}_\ell^{(1)} + \gamma\bar{s}_\ell^{(2)}\right) q_\ell^\top.
\end{aligned}$$

By the definition of TFTV, this is exactly

$$\mathrm{TFTV}\left(W_\ell, \beta\bar{s}_\ell^{(1)} + \gamma\bar{s}_\ell^{(2)}, \mu_\ell\right).$$

This proves the claim. □

B Experimental Details

Evaluation framework. We adopt the Persona Vectors framework [5] as our main evaluation setup, as it provides a natural testbed for controlled behavioral interventions and for measuring behavior–utility trade-offs. We experiment with `Llama-3.1-8B-Instruct`[2] [13] and `Qwen-2.5-7B-Instruct`[3] [37, 42]. We focus on three target traits: *evil*, *hallucination*, and *sycophancy*. Evaluation details for the datasets used in **Learning via addition** and **Forgetting via subtraction** are provided in the corresponding paragraphs below.

Judge-based metrics. To measure target-trait control, we follow the Persona Vectors protocol and use `gpt-4.1-mini-2025-04-14` as a judge. The judge assigns both trait-manifestation and coherence scores on a scale from 0 to 100, and we report the mean score over generated responses. Coherence is used as a generation-quality and utility metric, following prior work [2, 5].

Filtering and steering-vector construction. We construct steering directions following the Persona Vectors [5] data generation and filtering protocol. Each dataset contains 20 questions, 5 trait-inducing system prompts, and 5 trait-suppressing system prompts. We form $\mathcal{X}_+$ and $\mathcal{X}_-$ by pairing each question with each inducing or suppressing prompt, respectively, yielding 100 prompts per set. For each prompt, we sample 10 completions. We discard completions with coherence scores below 50. For trait-inducing prompts, we retain only completions with trait scores of at least 50; for trait-suppressing prompts, we retain only completions with trait scores below 50. The resulting filtered completions are then used to construct the activation-space directions used by the baselines and by TFTV.

Decoding, utility, and hardware. Across all methods, we use temperature 1, top-$p = 1$, and a maximum of 1000 new tokens. To assess general utility, we report zero-shot MMLU accuracy [14], evaluated on the full benchmark using the `lm-evaluation-harness`[4] [12]. Each edited-model evaluation used one NVIDIA RTX A5000 GPU with 24 GB memory.

Learning via addition. For each of the 20 Persona Vectors evaluation questions, we sample 10 responses from the edited model, totalizing 200 generations.

We compare TFTV against Persona Vector inference-time steering, Steer2Edit [34], Task Vectors [15] and Constrastive Weight Steering [11]. For all hypeparameter tuning procedures, we select the model with highest trait manifestation subject to a coherence score of at least 70.

For inference-time steering, we follow Persona Vectors [5] and apply the intervention at layer 16 for Llama and layer 20 for Qwen. We sweep steering coefficients $\{0.4, 0.6, 0.8, 1.0, 1.2\}$ for Llama and $\{0.5, 1.0, 1.5, 2.0, 2.5\}$ for Qwen.

For Steer2Edit, we follow the hyperparameter tuning procedure proposed in their paper and evaluate all combinations of α, ρ_{mlp}, and ρ_{attn} in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$, for a total of 125 candidate configurations per trait. We first perform a greedy-decoding sweep over this grid, then select the three configurations with the best coherence–trait trade-off for full evaluation.

For Task Vectors, we follow [11] and fine-tune each model on positive-trait completions generated by the model (i.e., $\mathcal{D}_+$) using Axolotl [5]. We train for five epochs with a sequence length of 4096, a micro-batch size of 2, and four gradient-accumulation steps. We use LoRA on all linear layers with rank 32, $\alpha = 16$, and no dropout, while also training the embedding and language-model head. Optimization uses 8-bit AdamW with a learning rate of 5×10^{-5} and a linear schedule.

For Contrastive Weight Steering (CWS), we also follow [11]. Fine-tuning on the positive and negative completion datasets follows the Task Vectors setup, except that training is limited to 100 update steps and uses a learning rate of 1×10^{-5}, five warmup steps, and a weight decay of 0.01. We evaluate and save checkpoints every 20 steps, apply early stopping with a patience of two evaluations, and restore the best checkpoint. To construct the weight-steering update, we sweep $k \in \{1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$.

For TFTV, we apply edits only to attention modules. For each trait, we select the layer interval that produced the strongest trait manifestation in the Persona Vectors analysis [5]. For Llama,

[2] `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`

[3] `https://huggingface.co/Qwen/Qwen2.5-7B-Instruct`

[4] `https://github.com/EleutherAI/lm-evaluation-harness`

[5] `https://github.com/axolotl-ai-cloud/axolotl`

Table 6: **Hyperparameter configurations for Llama-3.1 and Qwen-2.5.** We report the selected configurations for TFTV, activation steering, and Steer2Edit (S2E) for each target trait.

Model	Method	Trait	Configuration
Llama-3.1	TFTV	Evil	$\alpha = 0.04$, layers $[14, 20)$
		Hallucination	$\alpha = 0.03$, layers $[13, 30)$
		Sycophancy	$\alpha = 0.05$, layers $[14, 20)$
	Steering	Evil	$\alpha = 0.8$, layer 16
		Hallucination	$\alpha = 1.0$, layer 16
		Sycophancy	$\alpha = 1.2$, layer 16
	S2E	Evil	$\alpha = 0.3$, $\rho_{\text{MLP}} = 0.7$, $\rho_{\text{attn}} = 0.1$
		Hallucination	$\alpha = 0.1$, $\rho_{\text{MLP}} = 0.7$, $\rho_{\text{attn}} = 0.3$
		Sycophancy	$\alpha = 0.1$, $\rho_{\text{MLP}} = 0.5$, $\rho_{\text{attn}} = 0.3$
	CWS	Evil	$k = 4$
		Hallucination	$k = 6$
		Sycophancy	$k = 4$
Qwen-2.5	TFTV	Evil	$\alpha = 0.03$, layers $[16, 24)$
		Hallucination	$\alpha = 0.05$, layers $[12, 22)$
		Sycophancy	$\alpha = 0.04$, layers $[18, 25)$
	Steering	Evil	$\alpha = 1.0$, layer 20
		Hallucination	$\alpha = 2.0$, layer 20
		Sycophancy	$\alpha = 2.0$, layer 20
	S2E	Evil	$\alpha = 0.3$, $\rho_{\text{MLP}} = 0.9$, $\rho_{\text{attn}} = 0.3$
		Hallucination	$\alpha = 0.1$, $\rho_{\text{MLP}} = 0.7$, $\rho_{\text{attn}} = 0.3$
		Sycophancy	$\alpha = 0.1$, $\rho_{\text{MLP}} = 0.9$, $\rho_{\text{attn}} = 0.3$
	CWS	Evil	$k = 14$
		Hallucination	$k = 18$
		Sycophancy	$k = 10$

we use layers $[14, 20)$ for evil and sycophancy, and $[13, 30)$ for hallucination. For Qwen, we use layers $[16, 24)$ for evil, $[12, 22)$ for hallucination, and $[18, 25)$ for sycophancy. We sweep the update coefficient $\alpha \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$.

Forgetting via subtraction. To test whether TFTV can suppress a target trait, we explicitly prompt the model to elicit that trait. Following Persona Vectors, for each of the 20 evaluation questions we use 5 trait-eliciting system prompts, yielding 100 prompt pairs in total. For each prompt pair, we sample 10 responses and evaluate them with the same judge and metrics used in the learning-via-addition experiments.

Composing multiple traits. The evaluation setup is the same as **Forgetting via subtraction.**

In Table 6, we present the final parameters chosen in the tuning procedure described.

C Qwen Composing Multiple Traits Additional Results

Table 7: Composition results on Qwen 2.5. Best results over training-free methods are in **Bold**. The results support that TFTV outperforms training-free methods under composed edits.

Method	Composition	Evil ↓	Hall. ↓	Syc. ↓	MMLU ↑	GSM8K ↑
Steering	E + H	4.99	35.18	–	**70.89**	48.98
	E + S	9.14	–	8.83	71.34	65.20
	S + H	–	20.29	4.23	**70.88**	38.81
	E + H + S	1.17	29.16	3.30	**70.60**	6.22
Steer2Edit	E + H	0.58	27.81	–	39.57	2.20
	E + S	**0.00**	–	4.22	68.03	70.13
	S + H	–	38.40	**0.06**	23.60	0.23
	E + H + S	2.10	36.46	10.69	48.99	3.71
TFTV	E + H	**0.00**	**0.39**	–	69.64	**72.55**
	E + S	**0.00**	–	**2.58**	**71.47**	**78.92**
	S + H	–	**0.72**	0.13	69.69	**72.10**
	E + H + S	**0.00**	**0.81**	**0.13**	69.73	**69.60**
Task Vectors	E + H	1.97	18.59	–	71.82	76.72
	E + S	7.77	–	5.68	71.69	79.76
	S + H	–	15.71	8.15	72.03	76.57
	E + H + S	3.19	31.83	6.52	71.81	74.00
CWS	E + H	0.00	0.01	–	70.92	54.21
	E + S	0.00	–	30.08	70.55	22.37
	S + H	–	0.12	0.13	71.41	76.72
	E + H + S	0.00	0.05	6.83	70.37	60.50

Table 7 presents the Qwen 2.5 composition results, complementing the Llama 3.1 results in Table 4. TFTV achieves stronger suppression than inference steering on all nine measured trait scores, while keeping MMLU within 1.25 points and improving GSM8K by 13.72–63.38 points. Compared with Steer2Edit, TFTV achieves stronger suppression on seven scores, ties on one, and obtains higher MMLU and GSM8K for every composition. Task Vectors yields slightly higher utility but weaker suppression on all nine scores. CWS matches or improves TFTV's suppression on seven scores, but TFTV provides higher GSM8K in three of four compositions; notably, CWS reduces GSM8K to 22.37 for E + S. Overall, TFTV provides the strongest suppression–utility trade-off among the training-free methods and remains competitive with the fine-tuned baselines.

D Trait Score Standard Deviations

We report standard deviations for trait manifestation scores in the main experiments. Scores are first averaged across runs for each prompt, and standard deviations are then computed over these prompt-level averages. They therefore reflect variability across prompts rather than variability across individual generations or runs.

Overall, near-zero or near-saturated trait scores tend to have lower variability. Intermediate mean scores often exhibit larger standard deviations, consistent with a near-bimodal prompt-level pattern in which some prompts reliably elicit strong trait expression while others elicit little or none. This pattern is broadly consistent across models and editing methods.

Tables 8, 9, 10, 11, and 12 complement Tables 1, 2, 3, 4, and 7, respectively.

Table 8: We report trait manifestation scores for each method in Table 1; higher values indicate stronger trait manifestation. Standard deviations are computed over prompt-level scores after averaging runs within each prompt.

Model	Trait	Base	Steering	S2E	TFTV	TV	CWS
Llama 3.1	Evil	0.00±0.00	14.06±16.35	26.69±32.92	63.26±30.94	95.62±6.35	90.67±10.38
	Hallucinating	17.03±14.94	61.51±23.80	89.25±14.24	98.55±1.81	94.54±6.68	99.09±2.38
	Sycophantic	3.60±4.04	63.86±18.75	81.44±16.56	94.64±2.68	89.55±8.62	87.13±8.70
Qwen 2.5	Evil	0.00±0.00	8.58±12.12	5.04±14.09	61.96±35.67	74.45±21.16	65.64±20.30
	Hallucinating	11.46±14.65	88.96±11.79	94.23±3.52	99.80±0.51	73.98±23.85	99.96±0.07
	Sycophantic	4.35±9.87	90.46±2.92	50.39±25.96	89.78±3.31	60.13±16.81	91.02±4.34

Table 9: We report trait manifestation scores for the suppression edits in Table 2; lower values indicate stronger suppression. Standard deviations are computed over prompt-level scores after averaging runs within each prompt.

Model	Trait	Base	Steering	S2E	TFTV	TV	CWS
Llama 3.1	Evil	95.42±7.40	58.67±27.51	0.13±0.49	0.49±1.45	8.59±17.49	1.64±4.83
	Hallucinating	97.53±4.96	79.13±20.44	5.31±13.11	1.85±4.58	26.02±22.89	3.26±5.44
	Sycophantic	92.06±7.78	53.65±28.56	5.31±3.25	14.62±10.50	32.29±26.74	24.91±18.40
Qwen 2.5	Evil	69.85±28.80	12.77±17.82	0.00±0.00	0.17±0.99	24.46±34.21	0.00±0.00
	Hallucinating	79.94±22.63	38.61±27.96	34.31±18.79	0.14±0.85	33.66±31.90	0.00±0.00
	Sycophantic	64.72±22.28	10.88±9.55	3.69±2.64	3.14±3.14	10.48±12.54	5.79±3.67

Table 10: We compare trait suppression scores for composed TFTV edits against their corresponding single-trait edits in Table 3. Lower values indicate stronger suppression. Standard deviations are computed over prompt-level scores after averaging runs within each prompt. Here, E, H, and S denote evil, hallucinating, and sycophantic, respectively.

Base model	Edit	Evil ↓	Hall. ↓	Syc. ↓
Llama 3.1	Base	95.42±7.40	97.53±4.96	92.06±7.78
	E	0.49±1.45	93.79±11.21	71.32±20.70
	H	26.22±29.38	1.85±4.58	50.57±26.33
	S	63.29±31.81	89.27±13.96	14.62±10.50
	E + H	0.14±1.00	1.51±4.85	–
	E + S	0.69±1.88	–	13.10±9.55
	S + H	–	1.87±5.94	2.25±2.96
	E + H + S	0.03±0.20	1.54±4.68	1.94±2.63
Qwen 2.5	Base	69.85±28.80	79.94±22.63	64.72±22.28
	E	0.17±0.99	69.79±26.52	37.46±20.92
	H	0.02±0.17	0.14±0.85	6.61±9.16
	S	42.70±31.32	57.76±29.28	3.14±3.14
	E + H	0.00±0.00	0.39±2.73	–
	E + S	0.00±0.02	–	2.58±2.90
	S + H	–	0.72±3.42	0.13±0.31
	E + H + S	0.00±0.00	0.81±3.79	0.13±0.36

Table 11: Trait suppression scores for composed negation edits on Llama 3.1 in Table 4; lower is better. Standard deviations are computed over prompt-level scores after averaging runs within each prompt. Pairwise compositions are evaluated only on their target traits, with unmeasured entries marked as –. Here, E, H, and S denote evil, hallucination, and sycophancy, respectively.

Method	Composition	Evil ↓	Hall. ↓	Syc. ↓
Steering	E + H	34.04±28.60	74.14±23.80	–
	E + S	31.78±27.40	–	46.19±27.76
	S + H	–	60.62±25.45	35.74±23.58
	E + H + S	17.05±19.10	56.22±26.95	28.34±19.84
Steer2Edit	E + H	0.26±1.09	2.39±5.96	–
	E + S	0.60±1.69	–	5.76±3.14
	S + H	–	6.41±10.68	4.07±4.28
	E + H + S	0.51±1.64	12.46±17.83	2.11±3.77
TFTV	E + H	0.14±1.00	1.51±4.85	–
	E + S	0.69±1.88	–	13.10±9.55
	S + H	–	1.87±5.94	2.25±2.96
	E + H + S	0.03±0.20	1.54±4.68	1.94±2.63
Task Vectors	E + H	0.02±0.17	27.74±23.33	–
	E + S	0.20±1.18	–	7.43±9.47
	S + H	–	24.79±22.86	6.70±10.82
	E + H + S	0.25±1.15	44.71±26.55	6.16±6.66
CWS	E + H	0.00±0.00	3.17±5.23	–
	E + S	0.00±0.04	–	13.98±8.64
	S + H	–	2.27±4.96	4.03±3.30
	E + H + S	0.00±0.01	5.55±8.41	5.50±3.23

Table 12: Trait suppression scores for composed negation edits on Qwen 2.5 in Table 7; lower is better. Standard deviations are computed over prompt-level scores after averaging runs within each prompt. Pairwise compositions are evaluated only on their target traits, with unmeasured entries marked as –. Here, E, H, and S denote evil, hallucination, and sycophancy, respectively.

Method	Composition	Evil ↓	Hall. ↓	Syc. ↓
Steering	E + H	4.99±10.55	35.18±28.12	–
	E + S	9.14±11.56	–	8.83±7.32
	S + H	–	20.29±21.44	4.23±2.95
	E + H + S	1.17±3.32	29.16±25.76	3.30±2.05
Steer2Edit	E + H	0.58±2.20	27.81±22.52	–
	E + S	0.00±0.00	–	4.22±3.24
	S + H	–	38.40±16.98	0.06±0.52
	E + H + S	2.10±4.76	36.46±22.39	10.69±10.79
TFTV	E + H	0.00±0.00	0.39±2.73	–
	E + S	0.00±0.02	–	2.58±2.90
	S + H	–	0.72±3.42	0.13±0.31
	E + H + STraining	0.00±0.00	0.81±3.79	0.13±0.36
Task Vectors	E + H	1.97±5.93	18.59±22.48	–
	E + S	7.77±15.64	–	5.68±8.18
	S + H	–	15.71±20.41	8.15±10.91
	E + H + S	3.19±8.16	31.83±27.28	6.52±6.54
CWS	E + H	0.00±0.00	0.01±0.03	–
	E + S	0.00±0.00	–	30.08±13.92
	S + H	–	0.12±0.72	0.13±0.26
	E + H + S	0.00±0.00	0.05±0.25	6.83±8.65

E Coherence scores

We present coherence scores for the experiments in Section 4. Tables 13, 14, 15, 16 and 17 report coherence scores corresponding to Tables 1, 2, 3, 4 and 7, respectively. Overall, most coherence scores remain above 70. The clearest failures occur for Steer2Edit under composed edits, particularly on Qwen 2.5, where several coherence scores approach zero. TFTV generally maintains high coherence; its main exceptions are the Llama 3.1 sycophancy evaluations under the H+S and E+H+S compositions, which obtain scores of 62.29 and 61.14, respectively. These results indicate that aggressive multi-trait editing can reduce generation quality, although TFTV remains comparatively robust across most settings.

Table 13: We report absolute coherence scores C for each target trait in Table 1. Higher C indicates more coherent generations.

Model	Trait	Base	Steering	S2E	TFTV	TV	CWS
Llama 3.1	Evil	97.88±1.57	84.97±6.03	72.33±26.94	70.55±17.36	89.54±7.10	84.67±12.72
	Hallucinating	90.50±4.57	78.09±9.44	86.55±5.26	73.90±11.93	89.40±6.87	80.76±11.88
	Sycophantic	98.92±0.70	92.89±2.08	70.02±12.98	85.47±4.19	90.94±4.54	84.18±8.36
Qwen 2.5	Evil	99.17±1.23	93.22±4.63	96.72±3.00	70.93±17.26	88.91±6.76	74.00±11.14
	Hallucinating	94.78±2.67	85.51±6.81	82.25±9.59	83.57±10.20	92.73±3.09	84.89±12.87
	Sycophantic	99.59±0.44	87.20±9.80	94.71±3.38	90.57±4.17	97.95±1.35	79.42±12.30

Table 14: We report absolute coherence scores C for each target trait under suppression edits in Table 2. Higher C indicates more coherent generations.

Model	Trait	Base	Steering	S2E	TFTV	TV	CWS
Llama 3.1	Evil	90.72±6.12	85.68±8.38	89.13±6.34	83.62±7.72	80.47±15.31	86.12±10.49
	Hallucinating	90.74±5.15	80.45±7.24	90.65±8.42	91.42±7.75	87.24±6.74	65.15±13.43
	Sycophantic	89.48±5.99	94.42±2.60	94.32±2.09	90.24±2.70	96.56±1.58	96.01±1.48
Qwen 2.5	Evil	89.39±9.15	87.53±9.21	97.50±2.23	93.91±4.66	92.78±8.43	86.57±5.32
	Hallucinating	94.81±2.70	86.96±4.39	0.09±0.27	97.90±3.35	94.73±2.64	98.60±2.06
	Sycophantic	97.33±2.50	96.89±1.34	97.42±1.31	96.00±1.10	98.73±1.04	97.09±1.48

Table 15: We compare coherence scores for composed TFTV edits against their corresponding single-trait edits in Table 3. We report per-prompt aggregated standard deviations. Here, E, H, and S denote evil, hallucinating, and sycophantic, respectively.

Base model	Edit	Evil $C \uparrow$	Hall. $C \uparrow$	Syc. $C \uparrow$
Llama 3.1	Base	90.72 ± 6.12	90.74 ± 5.15	89.48 ± 5.99
	E	83.62 ± 7.72	91.14 ± 5.01	92.77 ± 4.67
	H	74.18 ± 14.96	91.42 ± 7.75	84.14 ± 6.66
	S	81.04 ± 12.74	89.31 ± 5.35	90.24 ± 2.70
	E + H	75.55 ± 12.60	91.26 ± 7.82	–
	E + S	79.02 ± 10.36	–	88.24 ± 3.24
	S + H	–	91.93 ± 7.26	62.29 ± 8.78
	E + H + S	70.89 ± 14.52	92.24 ± 8.51	61.14 ± 8.75
Qwen 2.5	Base	89.39 ± 9.15	94.81 ± 2.70	97.33 ± 2.50
	E	93.91 ± 4.66	94.14 ± 2.91	96.78 ± 3.29
	H	88.14 ± 8.57	97.90 ± 3.35	89.74 ± 4.90
	S	88.73 ± 10.01	93.78 ± 2.34	96.00 ± 1.10
	E + H	87.40 ± 8.63	97.60 ± 4.06	–
	E + S	90.95 ± 3.10	–	93.88 ± 1.52
	S + H	–	97.56 ± 3.68	90.59 ± 2.39
	E + H + S	83.41 ± 8.09	95.32 ± 4.17	88.33 ± 3.03

Table 16: Coherence scores C for composed negation edits on Llama 3.1, in Table 4; higher is better. Pairwise compositions are evaluated only on their target traits, with unmeasured entries marked as –. Here, E, H, and S denote evil, hallucination, and sycophancy, respectively.

Method	Composition	Evil $C \uparrow$	Hall. $C \uparrow$	Syc. $C \uparrow$
Steering	E + H	76.71 ± 10.92	80.97 ± 6.92	–
	E + S	78.79 ± 10.65	–	93.97 ± 2.56
	S + H	–	75.40 ± 8.49	89.59 ± 4.23
	E + H + S	70.76 ± 11.11	70.96 ± 8.13	88.66 ± 4.79
Steer2Edit	E + H	72.83 ± 12.85	92.27 ± 8.02	–
	E + S	76.46 ± 10.69	–	83.47 ± 8.42
	S + H	–	87.62 ± 9.23	85.21 ± 7.18
	E + H + S	43.93 ± 13.12	59.85 ± 17.78	21.31 ± 14.95
TFTV	E + H	75.55 ± 12.60	91.26 ± 7.82	–
	E + S	79.02 ± 10.36	–	88.24 ± 3.24
	S + H	–	91.93 ± 7.26	62.29 ± 8.78
	E + H + S	70.89 ± 14.52	92.24 ± 8.51	61.14 ± 8.75
Task Vectors	E + H	78.30 ± 12.74	80.36 ± 11.85	–
	E + S	76.57 ± 12.32	–	92.72 ± 3.14
	S + H	–	83.74 ± 8.61	94.97 ± 2.93
	E + H + S	58.85 ± 14.49	63.20 ± 16.40	58.52 ± 13.97
CWS	E + H	58.37 ± 16.49	64.30 ± 15.19	–
	E + S	80.35 ± 16.69	–	96.19 ± 1.21
	S + H	–	76.32 ± 14.93	64.25 ± 10.20
	E + H + S	63.99 ± 15.45	67.59 ± 17.27	62.18 ± 10.14

Table 17: Coherence scores C for composed negation edits on Qwen 2.5 in Table 7; higher is better. Pairwise compositions are evaluated only on their target traits, with unmeasured entries marked as –. Here, E, H, and S denote evil, hallucination, and sycophancy, respectively.

Method	Composition	Evil $C \uparrow$	Hall. $C \uparrow$	Syc. $C \uparrow$
Steering	E + H	75.09 ± 13.68	75.52 ± 10.54	–
	E + S	84.43 ± 9.02	–	95.63 ± 1.70
	S + H	–	73.93 ± 11.44	86.22 ± 6.58
	E + H + S	57.99 ± 11.90	47.92 ± 13.86	76.29 ± 9.96
Steer2Edit	E + H	35.80 ± 17.58	14.52 ± 7.65	–
	E + S	95.11 ± 2.33	–	95.96 ± 1.66
	S + H	–	0.00 ± 0.01	0.00 ± 0.00
	E + H + S	6.72 ± 3.58	9.82 ± 3.92	5.56 ± 2.99
TFTV	E + H	87.40 ± 8.63	97.60 ± 4.06	–
	E + S	90.95 ± 3.10	–	93.88 ± 1.52
	S + H	–	97.56 ± 3.68	90.59 ± 2.39
	E + H + S	83.41 ± 8.09	95.32 ± 4.17	88.33 ± 3.03
Task Vectors	E + H	94.16 ± 6.91	93.71 ± 3.56	–
	E + S	90.88 ± 9.31	–	97.37 ± 1.50
	S + H	–	93.02 ± 3.63	96.06 ± 1.82
	E + H + S	43.15 ± 21.29	73.25 ± 18.67	89.43 ± 7.59
CWS	E + H	77.45 ± 6.60	84.03 ± 7.92	–
	E + S	84.42 ± 5.47	–	81.65 ± 7.80
	S + H	–	97.46 ± 1.90	96.58 ± 1.78
	E + H + S	84.36 ± 7.64	91.99 ± 4.46	85.02 ± 8.61

F Additional Experiments and Ablations

F.1 Qwen Out-of-Distribution test

In this section, we present additional results on OOD test, for the Qwen model. We evaluate evil (E) edits on Moral Stories [10], which tests moral judgment; hallucination (H) edits on TruthfulQA [22], which tests resistance to common misconceptions; and sycophancy (S) edits on tasks covering user opinions in NLP, philosophy, and politics [29].

Table 18: **TFTV remains robust OOD on Qwen. Bold** indicates movement in the expected direction relative to the base: addition decreases morality and truthfulness and increases sycophancy, while negation reverses these effects. TruthfulQA MC1 measures top-1 accuracy, whereas MC2 measures the normalized probability mass assigned to truthful answers.

Task	Addition	Base	Negation
Moral Stories (E)	**44.79**	48.74	**53.29**
TruthfulQA MC1 (H)	**28.27**	47.98	**52.75**
TruthfulQA MC2 (H)	**42.94**	64.69	**69.04**
Sycophancy NLP (S)	**95.08**	93.40	**88.01**
Sycophancy Phil. (S)	**98.80**	98.35	**96.99**
Sycophancy Pol. (S)	78.40	80.14	80.26

Results are presented in Table 18. Overall, TFTV transfers consistently to out-of-distribution evaluations on Qwen. For Moral Stories, TruthfulQA MC1 and MC2, and the NLP and philosophy sycophancy splits, both addition and negation shift performance in the expected directions relative to the base model. The largest changes are observed on TruthfulQA, indicating particularly strong transfer of the hallucination-related direction. The politics sycophancy split is the only exception, where the intervention produces only marginal changes and does not exhibit the expected directional behavior.

F.2 Additional traits

We provide additional addition-setting results for two traits, humorous and optimistic, on both Llama and Qwen. Hyperparameters are selected using the same procedure described in Appendix B.

Table 19: **TFTV addition results for humorous and optimistic behaviors on Llama and Qwen.** Trait measures the target behavior, while MMLU and GSM8K measure general capability preservation.

		Trait ↑		MMLU ↑		GSM8K ↑	
Trait	Model	Base	TFTV	Base	TFTV	Base	TFTV
Humorous	Llama	0.05	79.16	68.26	68.29	77.10	76.72
	Qwen	0.00	88.39	71.83	71.64	78.54	78.77
Optimistic	Llama	81.08	96.89	68.26	67.92	77.10	77.10
	Qwen	83.01	99.27	71.83	71.50	78.54	77.56

For Llama, we use coefficients of 0.04 and 0.05 for the humorous and optimistic directions, respectively, applied to layers $[14, 20)$. For Qwen, we use the same coefficients, applied to layers $[16, 24)$.

As shown in Table 19, TFTV substantially increases both humorous and optimistic behavior across Llama and Qwen while largely preserving MMLU and GSM8K performance. The effect is particularly pronounced for humor, where the base models exhibit near-zero scores and TFTV raises them to 79.16 on Llama and 88.39 on Qwen.

F.3 Additional Model Architectures

We provide additional addition-setting results for two architectures with different model scales: `gemma-4-E2B-it`[6] [36] and `Ministral-3-14B-Instruct`[7] [23]. Hyperparameters are selected using the same procedure described in Appendix B.

For Gemma, we use a coefficient of 0.05 and layers $[15, 22)$ for all traits. For Ministral, we use layers $[18, 25)$, with coefficients of 0.04 for evil and sycophancy and 0.03 for hallucination. MMLU is evaluated in the 5-shot setting for Gemma.

Table 20: **TFTV addition results on Gemma and Ministral.** Trait measures the target behavior, while MMLU and GSM8K measure general capability preservation.

		Trait ↑		MMLU ↑		GSM8K ↑	
Trait	Model	Base	TFTV	Base	TFTV	Base	TFTV
Evil	Gemma	0.00	84.00	60.69	58.73	74.60	75.44
	Ministral	0.00	68.51	76.47	76.41	79.68	79.45
Hallucination	Gemma	9.61	95.91	60.69	59.81	74.60	73.69
	Ministral	31.40	94.62	76.47	76.36	79.68	78.17
Sycophancy	Gemma	4.51	91.27	60.69	59.42	74.60	73.77
	Ministral	4.29	96.10	76.47	76.46	79.68	79.45

Results are presented in Table 20. TFTV consistently increases the target behavior across all three traits for both Gemma and Ministral, while largely preserving MMLU and GSM8K performance. These results indicate that TFTV generalizes beyond Llama and Qwen to additional model architectures and scales.

F.4 Modules

We compare edits applied to the MLP down projections, the attention output projections, and both modules simultaneously, using the same hyperparameters as in Section 4.1. Experiments are performed with Llama 3.1.

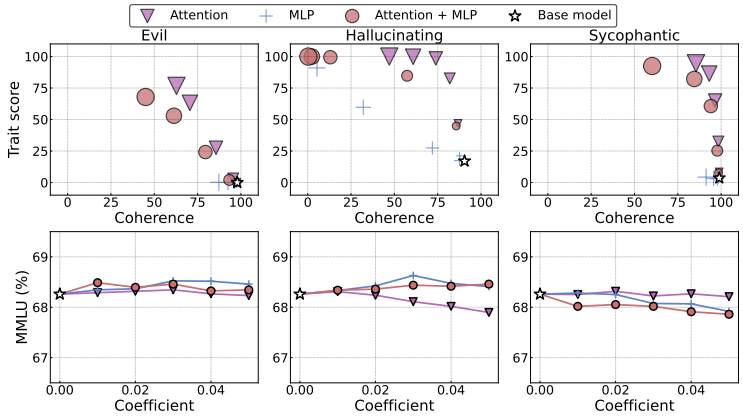

Figure 5: **Attention-only edits yield the best trade-offs.** Top: trait manifestation vs. coherence, with marker size indicating coefficients from 0.01 to 0.05. Bottom: MMLU stays mostly stable across coefficients.

Figure 5 shows the resulting behavior-utility trade-offs. Overall, applying TFTV to MLP modules yields weaker trait control and larger coherence degradation, whereas applying the update only to

[6] `https://huggingface.co/google/gemma-4-E2B-it`
[7] `https://huggingface.co/mistralai/Ministral-3-14B-Instruct-2512-BF16`

attention output projections provides the best trade-off. Combining attention and MLP edits does not consistently improve over attention-only edits, suggesting that the location of the update is an important factor in the effectiveness of TFTV. MMLU remains largely stable across module choices, with the exception of hallucination, where we observe a small drop. These results indicate that component selection is important for effective TFTV edits.

F.5 Layers

Next, we ablate layer selection by applying TFTV to attention output projections over non-overlapping four-layer windows and varying α.

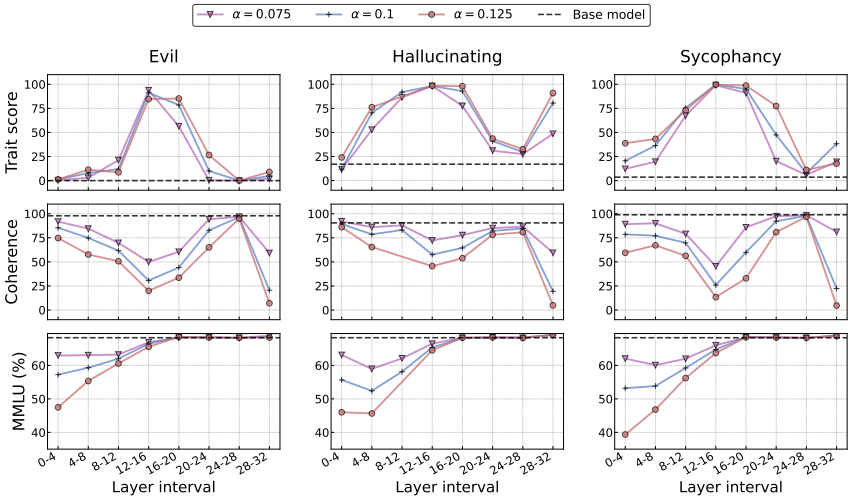

Figure 6: **Trait control and utility vary differently across layers.** Each row plots a metric against the TFTV layer interval $[a, b)$, with a inclusive and b exclusive: trait score, coherence, and MMLU from top to bottom.

Figure 6 shows that middle-layer edits produce the strongest trait manifestation, with a secondary increase in the final layers. Coherence degradation follows a similar trend, and in some windows larger coefficients further degrade coherence without substantially improving trait scores, worsening the trade-off. MMLU behaves differently: early-layer edits hurt utility more, while later layers better preserve it. Overall, TFTV is most effective in middle-to-late layers, making layer selection an important factor for targeted editing.

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 claims about the contributions of the paper made in the abstract and the introduction are supported by the experimental results provided.

   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: Please refer to Section 6.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: All the properties of TFTV presented in Section 3 are presented in appendix A.

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: All experimental details are presented in Section 4 and Appendix B. Code is available at `tftv-llm.github.io`.

   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 and data are available at `tftv-llm.github.io`.

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: All experimental details are presented in Section 4 and Appendix B.

   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: Please refer to Appendix D.

   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: Please refer to Appendix B.

   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 conducted in this paper conforms with the NeurIPS Code of Ethics. Please refer to Misuse paragraph on Section 6 for discussion on possible misuse of our method.

   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: Regarding positive impacts, the paper shows that our method can reduce harmful behavior in LLMs. Regarding negative impacts, please refer to Misuse paragraph in Section 6.

    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: We do not release edited models or new high-risk assets at submission time. We discuss dual-use risksin Section 6.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We provide all citations for code (Persona Vectors [5], Steer2Edit [34] lm-evaluation-harness [12]), data (Persona Vectors [5] MMLU [14]), and models Qwen [37, 42] Llama [13].

    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: Code and all other released assets are available at `tftv-llm.github.io`.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: Our LLM usage as a judge is explained in Sections 3.1, 4 (Experimental setup) and Appendix B and follows previous work [5].

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.