Textual Self-Attention Network: Test-Time Preference Optimization Through Textual Gradient-Based Attention

Shibing Mo[12*], **Haoyang Ruan**[1*], **Kai Wu**,[1†] **Jing Liu**[12]

[1]School of Artifcial Intelligence, Xidian University
[2]Guangzhou Institute of Technology, Xidian University
kwu@xidian.edu.cn

Abstract

Large Language Models (LLMs) have demonstrated remarkable generalization capabilities, but aligning their outputs with human preferences typically requires expensive supervised fine-tuning. Recent test-time methods leverage textual feedback to overcome this, but they often critique and revise a single candidate response, lacking a principled mechanism to systematically analyze, weigh, and synthesize the strengths of multiple promising candidates. Such a mechanism is crucial because different responses may excel in distinct aspects (e.g., clarity, factual accuracy, or tone), and combining their best elements may produce a far superior outcome. This paper proposes the Textual Self-Attention Network (TSAN), a new paradigm for test-time preference optimization that requires no parameter updates. TSAN emulates self-attention entirely in natural language to overcome this gap: it analyzes multiple candidates by formatting them into textual keys and values, weighs their relevance using an LLM-based attention module, and synthesizes their strengths into a new, preference-aligned response under the guidance of the learned textual attention. This entire process operates in a textual gradient space, enabling iterative and interpretable optimization. Empirical evaluations demonstrate that with just three test-time iterations on a base SFT model, TSAN outperforms supervised models like Llama-3.1-70B-Instruct and surpasses the current state-of-the-art test-time alignment method by effectively leveraging multiple candidate solutions.

Code — https://github.com/Explorermomo/TSAN-main
Extended version — http://arxiv.org/abs/2511.06682

Introduction

Aligning LLMs with human values and preferences has become a cornerstone of modern AI research, ensuring these powerful systems are helpful, harmless, and honest (Ji et al. 2023). The dominant paradigms for achieving this alignment, such as reinforcement learning from human feedback (RLHF) (Ouyang et al. 2022) and direct preference optimization (DPO) (Rafailov et al. 2023), are performed during the training phase (Wu et al. 2024). These methods embed preferences directly into the model's parameters, producing statically-aligned AI models. While effective, this approach is computationally intensive and lacks the flexibility to adapt to new or evolving preferences without expensive retraining cycles (Wang et al. 2024).

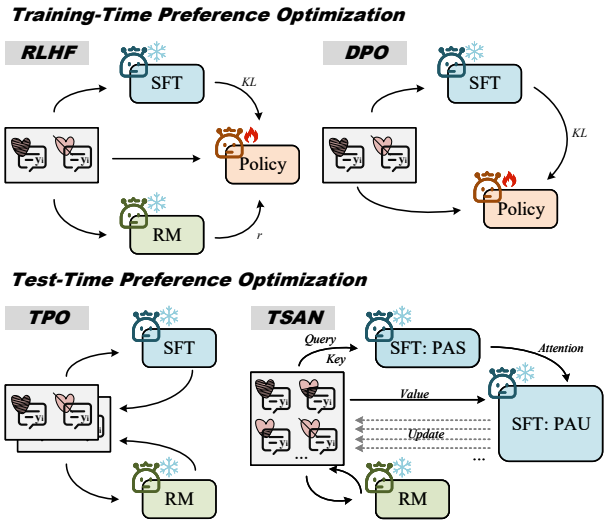

Figure 1: Comparison of Textual Self-Attention Network with existing preference optimization methods (e.g., RLFH, DPO, and TPO), where PAS and PAU represent textual attention scores and textual aggregation updates, respectively.

This inflexibility has given rise to an emerging area of test-time self-improvement (Raschka 2024). These methods seek to align LLM outputs on-the-fly, trading additional test-time computation for higher-quality, better-aligned responses during the generation process (Dong, Teleki, and Caverlee 2024). However, many existing test-time preference optimization techniques face significant limitations. Rudimentary methods like Best-of-N sampling (Lightman et al. 2023) rely on a simple scalar reward signal to select the best response from a pool of candidates. This scalar reward is a notorious information bottleneck and is highly susceptible to reward hacking, where the model learns to exploit proxy signals (e.g., generating verbose or formulaic answers) to achieve a high score without genuinely improving

*These authors contributed equally.
†Corresponding author.

quality (Khalaf et al. 2025).

More sophisticated approaches have begun to leverage richer, textual feedback to overcome the limitations of scalar rewards. Frameworks like test-time preference optimization (TPO) (Li et al. 2025) and critique & revise (Jin et al. 2023) generate natural language critiques and use them as textual gradients to iteratively improve a single response. As shown in figure 1, although this is an important step forward, a key limitation remains: these methods still operate in a linear, unstructured fashion. They typically critique one candidate response and then attempt to revise it, lacking a principled mechanism to systematically analyze, weigh, and synthesize the strengths of multiple promising candidate outputs simultaneously. This single-path revision process misses the opportunity to combine the best aspects of several strong alternatives, fundamentally limiting its optimization potential.

To address this gap, we introduce the TSAN, a novel test-time preference optimization method that operationalizes the principles of a self-attention mechanism within the textual domain. Rather than merely critiquing and revising a single output, TSAN formalizes a structured process for aggregating information from a set of high-quality candidate responses. It treats the user's prompt as a query (Q) and a curated set of strong initial answers as textual keys (K) and values (V). The framework then uses an LLM to generate textual attention scores — a natural language analysis of each key's relevance and merit with respect to the query. Guided by this rich, contextual attention signal, TSAN synthesizes a new, superior response by performing an aggregation update that integrates the most salient features from the Values. This entire process is enclosed in an iterative optimization loop driven by textual gradient descent, allowing the model to progressively refine its output for better alignment with user preferences. In summary, TSAN distinguishes itself from previous works through some key features:

- We introduce a new paradigm for test-time alignment focused on the principled, compositional synthesis of an optimal response from multiple candidates. This moves beyond the prevailing approach of selecting or revising a single response, establishing a more structured method to systematically analyze and integrate the strengths of diverse solutions.
- To realize this paradigm, we propose the TSAN, an innovative framework that operationalizes a query-key-value self-attention mechanism entirely within the textual domain. By generating natural language attention scores to weigh the merits of each candidate, TSAN effectively processes multiple inputs to synthesize a superior, preference-aligned output without any parameter updates.
- Extensive experiments have demonstrated that by structurally designing the refinement process through a textual attention mechanism, TSAN can provide a more robust and effective test-time alignment method.

Related Work

This work is situated at the intersection of LLM preference optimization, test-time alignment, and the emerging field of textual feedback mechanisms.

Preference Optimization Paradigms

Research in LLM alignment is broadly divided into two paradigms. Training-time alignment methods instill preferences by modifying model weights. This began with the complex, multi-stage pipeline of RLHF, which involves training a separate reward model and using reinforcement learning (like PPO (Schulman et al. 2017)) to optimize the policy. Due to its instability and high computational cost, the field has largely shifted towards simpler and more stable methods like DPO (Rafailov et al. 2023). DPO and its successors, such as RLOO (Zhang et al. 2024), CPO (Xu et al. 2024), and ORPO (Hong, Lee, and Thorne 2024), reframe the preference learning problem as a classification task, thereby eliminating the need for an explicit reward model and a complex reinforcement learning loop. However, all these methods produce a static model that cannot be adjusted post-training.

In contrast, test-time alignment aims to provide this missing flexibility by intervening during the generation process without updating model weights. These techniques range from decoding strategies and sampling methods to more complex iterative refinement loops. As shown in figure 1, TSAN falls squarely into this paradigm, offering a novel mechanism for on-the-fly alignment.

Test-Time Self-Improvement

The field of test-time self-improvement has explored several strategies for enhancing model outputs at test time.

Sampling and Search. The most straightforward approach is Best-of-N sampling (Lightman et al. 2023), where N candidate responses are generated, and the one with the highest score from a reward model is selected. More advanced techniques, such as Tree-of-Thoughts (Yao et al. 2023) and TreeBoN (Qiu et al. 2024), employ tree-based search to explore the solution space more effectively. However, these methods typically rely on scalar rewards and are thus vulnerable to reward hacking.

Iterative Refinement. For more nuanced correction, iterative methods have been proposed. The critique & revise framework involves generating an initial response, critiquing it, and then generating a revision based on the critique (Li et al. 2025; Jin et al. 2023). TPO formalizes this concept by treating the critique as the textual gradient (Yuksekgonul et al. 2025) that guides optimization. As shown in figure 1, TSAN builds upon this iterative foundation but introduces a key distinction: instead of a linear loop of critiquing and revising a single candidate, TSAN implements a structured, attention-based mechanism to synthesize an improved response from a whole set of strong candidates. More related work can be found in **Appendix A**.

Preliminary

The goal of preference optimization in LLMs is to align the output of a policy π_θ with human preferences. This objective is typically formalized as maximizing a scoring function s

over a preference dataset D, which consists of a prompt x, a chosen response y_w, and a rejected response y_l:

$$\max_{\pi} E_{(x,y_w,y_l)\sim\mathcal{D}}\big[s(\pi, x, y_w, y_l)\big] \tag{1}$$

Training-Time Preference Optimization

Mainstream alignment methods achieve this during the training phase by updating the model parameters θ. RLHF is a seminal work in this area. It first trains a reward model (RM), r_ϕ, to fit human preferences. Then, it uses reinforcement learning to maximize this reward for the policy model, while a KL-divergence term penalizes its deviation from a reference model π_{ref}:

$$\max_{\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}\big[r_\phi(x,y) - \beta\cdot KL\big(\pi_\theta(y|x)\,\big|\big|\,\pi_{\text{ref}}(y|x)\big)\big] \tag{2}$$

Test-Time Preference Optimization

In contrast to methods that update model weights, test-time optimization intervenes on the model's output during the inference phase. An emerging direction in this area is the use of textual gradients. This paradigm does not compute numerical gradients $\nabla_\theta L(\theta)$. Instead, it leverages the LLMs' own capabilities to iteratively revise its output text y guided by natural language critiques. This process can be abstracted as:

$$y_{i+1} \leftarrow \text{Revise}(y_i, \nabla_{\text{text}}), \quad \text{where } \nabla_{\text{text}} = \text{Critique}(y_i, L_{\text{text}}) \tag{3}$$

The TSAN we propose provides a novel and structured framework for applying textual gradients under this test-time optimization paradigm.

Method

To address the insufficiency of existing test-time preference optimization methods in integrating information from multiple candidate answers, we propose the TSAN. TSAN is to elevate the self-attention mechanism from the numerical level to the textual level at test time. Through a structured, LLM-driven process, it enables the systematic analysis, weighting, and synthesis of multiple high-quality candidate answers, thereby generating a more optimal response.

The overall framework of TSAN comprises three core phases: (1) Candidate generation and textual QKV construction; (2) Textual attention calculation; (3) Aggregate and update with alternating optimization. The entire process operates within an optimization loop driven by textual gradients.

Candidate Generation and Textual QKV Construction

Given a user input x, the TSAN process is initiated.

1. **Candidate Generation**: TSAE first utilize a policy model π_θ to generate a set of N diverse candidate responses $\{y_1, y_2, \ldots, y_N\}$ for the input x. This step initially explores the possible solution space.
2. **Reward Model Scoring and Selection**: Next, using an independent RM to score these N candidates, obtaining their respective preference scores. Then cache each candidate response and its score, and sort them in descending order based on their scores:

$$Y = \{(y_i, R(y_i))\}_{i=1}^{N} \tag{4}$$

3. **Textual QKV Construction**: Inspired by the self-attention mechanism, we reframe the optimization problem as an interaction of query, key, and value in the textual domain.
   (a) **Textual Query (Q)**: The user's original input x is directly used as the textual query, Q_{text}.
   (b) **Textual Key (K) and Value (V)**: Selecting the top-k candidate responses from the cache and concatenate them into a single, numbered string. This string serves simultaneously as the textual key, K_{test}, and the textual value, V_{test}. In TSAN, K_{test} is used to evaluate the relevance of each candidate answer to the query, while V_{test} acts as the carrier for the content of these answers, which will ultimately be synthesized. Formally:

$$Q_{\text{text}} = x \tag{5}$$

$$K_{\text{test}} = V_{\text{test}} = \text{Top-}k\big(\{(y_i, R(y_i))\}_{i=1}^{N}\big) \tag{6}$$

Textual Attention Score

In standard self-attention mechanisms, attention scores are numerical matrices calculated by computing the dot product of query vectors and key vectors. However, in TSAN, we use specially configured LLMs to generate a natural language format analysis report, which we refer to as **textual attention scores** (AS_{text}).

Specifically, we construct an attention score model, the $\text{PAS}_{\text{model}}$, which is an LLM configured with a specific system prompt P_{att}. This prompt instructs the model to role-play as an attention mechanism, with the following task:

> "*Analyze the relevance between the query Q and each candidate key K_i. Evaluate the key characteristics of each K_i regarding aspects such as performance and accuracy, and provide a step-by-step explanation. Finally, perform a qualitative summary of these characteristics to produce a coherent textual description that synthesizes all insights.*"

By feeding the textual query Q_{text} and the textual key K_{text} as input into this model, its output is the textual attention score, AS_{text}. This AS_{text} is a piece of text rich with associative information; it not only assesses relevance but also points out the advantages and disadvantages of each candidate answer, providing high-quality, interpretable guiding signals for the subsequent aggregation step.

$$AS_{\text{text}} = \text{PAS}_{\text{model}}(Q_{\text{text}}, K_{\text{text}}) \tag{7}$$

Aggregate Update

After obtaining the textual attention score, the next step is to utilize this score to guide a model in synthesizing the merits of all candidate answers (i.e., the textual value, V_{text}) to generate a completely new and more optimal response. We call this process **textual aggregate update**.

Similar to the previous step, we construct an **aggregate update model** $\text{PAU}_{\text{model}}$, which is configured by a system prompt P_{agg}. This prompt instructs the model:

"*You are an aggregation and update mechanism. Your task is, based on the textual attention score (AS_{text}) that analyzes the relationship between Q and each K_i, and in conjunction with the corresponding V_i, to integrate the information contained within these values and synthesize a new, rich output.*"

The inputs Q_{text}, AS_{text}, and V_{text} are fed into this model. The model's output, a synthetically refined answer, is then cached and denoted as y_{agg}. Conceptually, this step is equivalent to the process in self-attention where attention weights are applied to the value vectors, which are then summed to produce an output. In TSAN, however, this is an inference process based entirely on textual understanding and generation.

$$y_{\text{agg}} = \text{PAU}_{\text{model}}(Q_{\text{text}}, AS_{\text{text}}, V_{\text{text}}) \tag{8}$$

$$Y = \{(y_j, R(y_j))\}_{j=1}^{N} \cup y_{\text{agg}} \tag{9}$$

Iterative Optimization

During the test-time optimization process, the preceding steps are embedded into an iterative optimization loop. Driven by textual gradients, the system continuously adjusts the output to better align with the preferences of the reward model. This process is composed of the following parts:

1. **Textual Loss Calculation**: We use a prompt P_l to express the loss function, where an LLM_L is prompted to critique the previously generated answers $\{y_{\text{agg}}^i\}_{i=1}^M$ with respect to the original user prompt x. The process can be formalized as:

$$L_{\text{text}} = LLM_L(P_l, x, y_{\text{agg}}) \tag{10}$$

where the prompt P_l instructs LLM_L:

"*You are supposed to evaluate the model's response. Please assess the strengths and weaknesses of the model's answer,then refine the PAS_prompt to instruct the PAS_Model to focus more on certain aspects and ensure better alignment of attention,and refine the PAU_prompt to guide the PAU_Model on how to utilize the attention analysis results from the PAS_Model to generate a better answer from the existing answers. Step by step.*"

It can be seen that it not only to evaluate y_{agg} but also to provide feedback for improving the TSAN framework itself. Therefore, the resulting textual loss, L_{text}, is a structured critique containing feedback for various learnable components of the framework. We can formally represent this output as a tuple:

$$L_{\text{text}} = (\nabla_{y_{\text{agg}}}, \nabla_{P_{\text{att}}}, \nabla_{P_{\text{agg}}}) \tag{11}$$

2. **Gradient Computation**: A prompt P_{grad} is used to transform the textual loss L_{text} described above into an update instruction, thereby forming a textual gradient:

$$\frac{\partial \mathcal{L}}{\partial y_{\text{agg}}} = LLM(P_{\text{grad}}(L_{\text{text}})) \tag{12}$$

Algorithm 1: Textual Self-Attention Network (TSAN)

Require: Query x, policy model π_θ, reward model RM, max iterations T

Ensure: The final optimized answer y_{final}

// Initialization
$Y_{cand} \leftarrow$ Generate N candidate responses $\{y_1, \ldots, y_N\}$ in parallel using π_θ.
$K_{text}^{(0)}, V_{text}^{(0)} \leftarrow$ Evaluate Y_{cand} using RM, select the Top-k, and concatenate into a string.
$Q_{text} \leftarrow x$
for $i = 0$ to $T - 1$ **do**
  // Textual Attention Calculation
  $AS_{text}^{(i)} \leftarrow \text{PAS\_model}(Q_{text}, K_{text}^{(i)})$
  // Aggregate and Update
  $y_{agg} \leftarrow \text{PAU\_model}(Q_{text}, AS_{text}^{(i)}, V_{text}^{(i)})$
  // Textual Gradient Optimization
  $\nabla_{y_{agg}}^{(i)} \leftarrow L_{text}(y_{agg})$ // Generate critique
  $\{y_{\text{agg}}^{i\,\prime}\}_{i=1}^{M} \leftarrow \text{Optimizer.step}(\nabla_{y_{agg}}^{(i)})$ // Generate new candidate set based on the critique
  // Update candidates
  $Y_{cand} \leftarrow \{y_{\text{agg}}^{i\,\prime}\}_{i=1}^{M} \cup y_{agg} \cup Y_{cand}$
  // Update K and V for the next iteration
  $K_{text}^{(i+1)}, V_{text}^{(i+1)} \leftarrow$ Evaluate Y_{cand} using RM and select the Top-k.
end for
$y_{final} \leftarrow$ Select the highest-scoring response from all candidates generated across all iterations.
Return y_{final}

3. **Variable Optimization**: Finally, a prompt P_{update} is used to update the variables to generate an optimized set $\{y_{\text{agg}}^{i\,\prime}\}_{i=1}^{M}$, analogous to a gradient descent update rule:

$$\{y_{\text{agg}}^{i\,\prime}\}_{i=1}^{M} = LLM\left(P_{\text{update}}(\frac{\partial \mathcal{L}}{\partial y_{\text{agg}}})\right) \tag{13}$$

where the M parallel outputs can be considered as the outputs of M attention heads. These newly generated answers are re-scored by the reward model and are used to update the textual key K_{text} and textual value V_{text} for the next iteration.

$$Y = \{(y_j, R(y_j))\}_{j=1}^{N} \cup y_{agg} \cup \left\{y_{\text{agg}}^{i\,\prime}\right\}_{i=1}^{M} \tag{14}$$

The entire process (from textual attention score calculation to the optimizer steps) is repeated until a preset maximum number of iterations, max_iters, is reached. Besides, We adopt the vanilla prompts for P_{grad} and P_{update} from TextGrad (Yuksekgonul et al. 2025). The complete steps are detailed in **Algorithm 1**.

Experiments

Experimental Setup

Models. This research contrasts two policy model categories: unaligned and aligned, with the core distinction lying in whether preference optimization (e.g., RLHF or DPO)

Model	AlpacaEval 2 LC(%)	AlpacaEval 2 WR(%)	Arena-Hard 2	HH-RLHF	BeaverTails	XSTest	MATH-500
Llama-3.1-70B-DPO (IQ4)	9.47	15.79	6.3	-2.94	-7.30	64.1	25.8
Llama-3.1-70B-Instruct (IQ4)	**21.73**	18.18	6.8	**-2.45**	-7.50	69.2	24.0
Llama-3.1-70B-SFT (IQ4)	3.01	4.91	5.5	-6.65	-10.07	75.2	22.0
SFT - TPO (k4-T3)	17.95	<u>20.18</u>	6.0	-2.96	-6.67	<u>76.6</u>	**32.0**
SFT - TPO (k4-T5)	18.28	**22.15**	6.0	-2.96	<u>-6.53</u>	<u>76.6</u>	**32.0**
SFT - TSAN (k4-M4-T3)	<u>18.57</u>	17.05	<u>8.4</u>	<u>-2.88</u>	-6.63	**78.8**	<u>28.2</u>
SFT - TSAN (k4-M4-T5)	<u>18.57</u>	17.05	**8.5**	<u>-2.88</u>	**-6.45**	**78.8**	<u>28.2</u>

Table 1: Unaligned Model Performance Comparison on Multiple Benchmarks, where the IQ4 indicates INT4 quantization, k indicates the number of candidate answers sampled, M indicates the number of textual attention heads, and T indicates the number of TSAN iterations. The **bold** and <u>underlined</u> numbers indicate the best and second-best performances, respectively.

is applied. Specifically, for the unaligned model, we adopt Llama-3.1-Tulu-3-70B-SFT (IQ4 XS) (Lambert et al. 2024), obtained by supervised fine-tuning (SFT) on the foundation model Llama-3.1-70B (Grattafiori et al. 2024). For aligned models, we employ two publicly available models: Llama-3.1-70B-Instruct (IQ4 XS) (Grattafiori et al. 2024) and the smaller-scale Mistral-Small-Instruct-2409 (Jiang et al. 2023). For RM, we use FsfairX-LLaMA3-RM-v0.1 (Dong et al. 2024) for all policy models. Additionally, we follow the settings of TPO (Li et al. 2025), training an on-policy aligned model (termed Llama-3.1-70B-DPO) based on Llama-3.1Tulu-3-70B-SFT using UltraFeedback (Cui et al. 2023). More optimize Details can be found in **Appendix B1.**

Evaluation Benchmarks. We evaluate our model using a comprehensive set of benchmarks covering multiple facets, including instruction following (AlpacaEval 2 (Li et al. 2023) and Arena-Hard 2 (Li et al. 2024)), general preference alignment (HH-RLHF (Bai et al. 2022)), safety (BeaverTails (Ji et al. 2023) and XSTest (Röttger et al. 2023)), and mathematical capabilities (MATH-500 (Lightman et al. 2023)). More details about the dataset can be found in **Appendix B2**.

Experimental Results

We evaluate the effectiveness of our proposed TSAN by applying it to both unaligned and aligned base models. The results, detailed in table 1 and table 2, demonstrate that TSAN serves as a powerful and versatile test-time optimization framework, significantly enhancing model performance across a wide array of benchmarks.

Benchmark Performance

Unaligned Models. As presented in table 1, applying TSAN to the base supervised fine-tuned model (Llama-3.1-70B-SFT) yields substantial improvements, elevating its capabilities to be highly competitive with training-time alignment methods like DPO and TPO.

Most notably, on instruction-following benchmarks, TSAN brings dramatic gains. On AlpacaEval 2, it boosts the length-controlled win rate (LC) from a mere 3.01% to 18.57% and the raw WR from 4.91% to 17.05%. This demonstrates a vastly improved ability to adhere to user instructions. Similarly, on the more challenging Arena-Hard 2 benchmark, TSAN improves the score from 5.5% to 8.5%, surpassing the performance of the SFT-TPO model (6.0%).

In terms of preference alignment and safety, TSAN consistently steers the SFT model towards better-aligned and safer outputs. The average reward on HH-RLHF improves from -6.65 to -2.88, and on BeaverTails from -10.07 to -6.45. This indicates that TSAN effectively guides the model to generate responses that are better aligned with the reward model's learned human preferences. On XSTest, TSAN increases the compliance accuracy from 75.2% to 78.8%. Furthermore, TSAN enhances the model's reasoning capabilities, as shown by the improvement on MATH-500 from 22.0% to 28.2%.

When compared with SFT-TPO, a strong training-time baseline, our test-time SFT-TSAN method proves to be highly competitive. While TPO achieves a higher raw win rate on AlpacaEval 2 and a stronger score on MATH-500, TSAN outperforms TPO on Arena-Hard 2, XSTest, and achieves slightly better reward scores on HH-RLHF and BeaverTails. This highlights that TSAN can achieve a comparable level of alignment to training-time methods without requiring any gradient-based updates or extensive training data. And RM scores result can be found in **figure A1** in **Appendix C**.

Aligned Models. As shown in table 2, TSAN serves as a potent test-time enhancement for already-aligned models of varying scales. On the large-scale Llama-3.1-70B-Instruct, it delivers substantial gains across all benchmarks, notably improving the win rate on AlpacaEval 2 (WR from 18.18% to 23.19%) and Arena-Hard 2 (from 6.8% to 10.4%). The most striking improvements are in reasoning and safety, where TSAN boosts accuracy on MATH-500 from 24.0% to a remarkable 38.0% and on XSTest from 69.2% to 81.7%.

To demonstrate the generality of our method, we also applied TSAN to the smaller-scale Mistral-Small-Instruct-2409 model. A consistent pattern of significant improvement emerges. TSAN boosts Mistral-Small's performance on AlpacaEval 2 (WR from 16.10% to 25.38%) and Arena-Hard 2 (from 9.1 to 10.3), while also enhancing its MATH score to 32.2%. This demonstrates that TSAN is a model-agnostic

Model	AlpacaEval 2 LC(%)	AlpacaEval 2 WR(%)	Arena-Hard 2	HH-RLHF	BeaverTails	XSTest	MATH-500
Llama-3.1-70B-Instruct (IQ4)	21.73	18.18	6.8	-2.45	-7.50	69.2	24.0
+ TSAN (k4-M4-T3)	**26.51**	**23.19**	8.1	-1.73	-6.63	**81.7**	**38.0**
+ TSAN (k4-M4-T5)	**26.51**	**23.19**	**10.4**	**-1.52**	**-6.44**	**81.7**	**38.0**
Mistral-Small-Instruct-2409	15.76	16.10	9.1	-3.11	-6.95	72.0	27.6
+ TSAN (k4-M4-T3)	**22.57**	24.10	**10.3**	**-2.46**	-6.25	**74.4**	**32.2**
+ TSAN (k4-M4-T5)	**22.57**	**25.38**	**10.3**	**-2.46**	**-6.13**	**74.4**	**32.2**

Table 2: Aligned Model Performance Comparison on Multiple Benchmarks, where the IQ4 indicates INT4 quantization, k indicates the number of candidate answers sampled, M indicates the number of textual attention heads, T indicates the number of TSAN iterations and + *TSAN* indicates the use of TSAN based on the Llama-3.1-70B-Instruct (IQ4) model or the Mistral-Small-Instruct-2409 model. The **bold** indicate the best performances.

Model	AlpacaEval 2 LC(%)	AlpacaEval 2 WR(%)	Arena-Hard 2	HH-RLHF	BeaverTails	XSTest	MATH-500
Qwen-3-Plus	**65.08**	68.38	47.3	**2.47**	-4.26	75.1	90.0
+ TSAN (k4-M4-T3)	57.56	69.90	**72.1**	1.96	-3.16	**81.9**	90.6
+ TSAN (k4-M4-T5)	57.56	**71.69**	**72.1**	1.96	**-3.05**	**81.9**	**90.8**

Table 3: TSAN Performance (Model Scores) on Qwen-3-Plus, k indicates the number of candidate answers sampled, M indicates the number of textual attention heads, T indicates the number of TSAN iterations and + *TSAN* indicates the use of TSAN based on the Qwen-3-Plus API. The **bold** indicate the best performances.

Model	AlpacaEval 2 LC(%)	AlpacaEval 2 WR(%)	Arena-Hard 2	HH-RLHF	BeaverTails	XSTest	MATH-500
gpt-oss 20B	61.60	70.27	72.5	1.63	-7.45	84.5	96.0
gpt-oss 20B + TPO (T5)	63.90	72.70	75.2	1.44	-6.20	86.0	95.0
gpt-oss 20B + TSAN (T5)	**67.90**	**75.90**	**82.8**	**1.68**	**-5.98**	**86.3**	**97.6**
gpt-oss 120B + TSAN (T3)	67.70	76.90	84.4	2.52	-6.54	85.0	**96.4**
gpt-oss 120B + TSAN (T5)	**72.60**	**78.90**	**87.4**	**2.57**	**-5.57**	**88.9**	**96.4**

Table 4: TSAN Performance (Model Scores) on gpt-oss model. Configurations: k indicates the number of candidate answers sampled (k=4 for all experiments), M indicates the number of textual attention heads (M=4 for all experiments), T indicates the number of iterations (noted in the table).

enhancement technique, capable of delivering substantial performance lifts across different model architectures and scales.

Furthermore, increasing iterations from T3 to T5 consistently yields better results for both models, especially on Arena-Hard 2 and in reward scores on HH-RLHF, confirming the value of continued optimization. In summary, TSAN proves to be a versatile post-hoc enhancement method, unlocking significant performance gains on SOTA aligned models without any additional training. And RM scores result can be found in **Appendix C**.

TSAN Performance on Qwen-3-Plus. Experimental results in table 3 show that a key advantage of TSAN is its exceptional portability. Because the entire optimization process operates in the textual domain without requiring access to model weights or gradients, it can be applied to powerful, closed-source models as a "plug-and-play" enhancement through simple API calls.

To validate this, we applied TSAN to Qwen-3-Plus, a SOTA proprietary LLM accessible only via API. The results show that TSAN significantly enhances Qwen-3-Plus's performance across a majority of benchmarks, demonstrating its practical utility. Most notably, on the challenging Arena-Hard benchmark, TSAN boosts the score from 47.3% to a remarkable 72.1%. Similarly, the model's safety, as measured by XSTest, improves from 75.1% to 81.9%.

TSAN Performance on gpt-oss. In addition, we conducted experiments on the gpt-oss series models, and the results are shown in table 4. On the gpt-oss 20B model, inserting TSAN achieved comprehensive improvements compared to inserting TPO. Notably, the performance of the gpt-oss 20B model with TSAN inserted is comparable to that of

the gpt-oss 120B model, which is highly advantageous for deployment in resource-constrained environments.

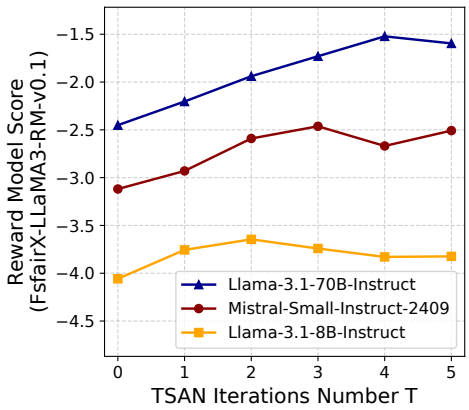

Figure 2: Test-time training curve of 'each aligned models + TSAN-k4-M4' Performance on the HH-RLHF dataset.

TSAN Performance on Small Model. To test TSAN's robustness, we experimented on the small-scale Llama-3.1-8B-Instruct model. Unlike methods such as TPO (Li et al. 2025), which can cause performance degradation on this model due to its weaker instruction-following capabilities, TSAN demonstrates a markedly different and positive optimization trajectory (Figure 2). The reward score consistently improves over the initial iterations, showing TSAN can successfully enhance models where other methods fail.

We attribute this resilience to TSAN's structured, scaffolded process. Instead of requiring the model to interpret abstract self-critiques, TSAN decomposes the task into first analyzing a set of concrete candidate answers (Keys) and then synthesizing a new response guided by that comparative analysis. This Query-Key-Value approach appears to lower the cognitive load on the policy model, providing a more robust improvement framework. This suggests TSAN is more broadly applicable, lowering the barrier for applying test-time optimization to a wider range of LLMs.

Finally, we provide some specific case studies on the application of TSAN to unaligned models and aligned models. For details of case studies, please refer to **Appendix D**.

Parameter Analysis

Candidate Sample Number. Figure 3 illustrates the effect of varying the number of candidate samples ($k \in \{2, 3, 4\}$) while keeping the number of attention heads fixed (M=4). The results clearly show a positive correlation between the number of candidates and the final performance.

Across all configurations, a single iteration of TSAN (T=1) provides a dramatic improvement over the baseline (T=0), with performance continuing to climb in subsequent iterations. Critically, increasing the number of candidate samples consistently yields a higher reward score at every iteration step. The SFT-TSAN-k4-M4 configuration achieves the best performance, followed by k3 and k2. This finding

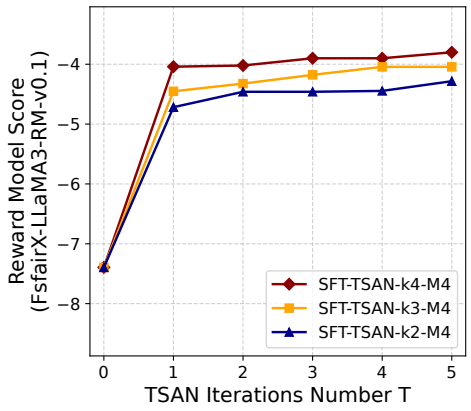

Figure 3: Experimental analysis results on MATH-500 dataset for candidate samples number.

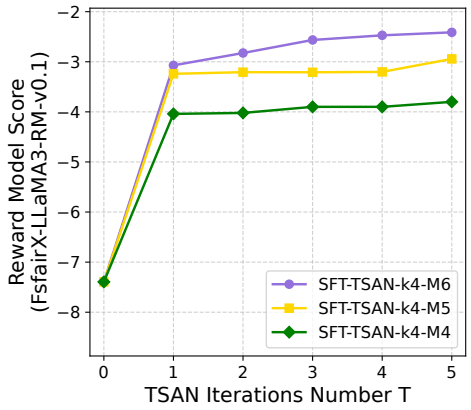

Figure 4: Experimental analysis results on MATH-500 dataset for attention heads number.

is significant and highlights a core advantage of our approach. In contrast to other methods that typically critique one candidate response and then attempt to revise it, lacking a principled mechanism to systematically analyze, weigh, and synthesize the strengths of multiple promising candidates, TSAN is explicitly designed to overcome this limitation.

Attention Heads Number. Figure 4 shows the results of varying the number of textual attention heads ($M \in \{4, 5, 6\}$) while holding the number of candidate samples constant ($k = 4$).

The SFT-TSAN-k4-M6 configuration consistently outperforms the M5 and M4 settings. This finding is aligned with the concept of multi-head attention in traditional transformers. In our framework, a larger M allows the textual gradient optimization step to produce a more diverse set of refined answers based on the critique. This greater diversity in the optimization trajectory prevents premature convergence and enables the model to explore more promising avenues for improvement, ultimately resulting in a higher-quality final output as judged by the reward model.

TSAN Computational Overhead Analysis

We analyze TSAN's computational cost using a PyTorch-based FLOPs counter[1]. The calculation setting is based on the TPO (Li et al. 2025). A full optimization cycle requires approximately **11.78 PFLOPs** per query. This is exceptionally efficient compared to training-time alignment, representing just **0.016%** of the cost to train a Llama-3.1-70B-DPO model (**72,840 PFLOPs**).

Crucially, this cost is only marginally higher than that of simpler test-time methods like TPO (**9.3 PFLOPs**). This near-equivalent efficiency is maintained despite TSAN's more sophisticated, multi-candidate synthesis mechanism. As our results demonstrate, this minimal computational overhead fuels a structured process that delivers substantially greater performance gains. TSAN thus provides a superior efficiency-to-performance profile, achieving SOTA test-time results without a costly computational trade-off.

Conclusion

In this paper, we introduced the TSAN, a novel test-time optimization framework designed to overcome the limitations of static alignment and unstructured revision methods. By emulating a query-key-value self-attention mechanism in the textual domain, TSAN systematically analyzes multiple candidate responses to synthesize a superior, preference-aligned output without updating any model parameters. Extensive experiments demonstrate that this approach enables a base model to outperform specially fine-tuned models and surpass the current SOTA in test-time alignment mothods. In summary, TSAN provides a structured new paradigm for achieving more dynamic, interpretable, and effective AI alignment, shifting the focus from simple response selection to principled, compositional synthesis.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62471371 and 62206205, in part by Natural Science Basic Research Program of Shaanxi under Grant 2025JC-QYCX-060, in part by the Young Talent Fund of Association for Science and Technology in Shaanxi, China under Grant 20230129, in part by the Guangdong High-level Innovation Research Institution Project under Grant 2021B0909050008, and in part by the Guangzhou Key Research and Development Program under Grant 202206030003.

References

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2204.05862*.

Cui, G.; Yuan, L.; Ding, N.; Yao, G.; Zhu, W.; Ni, Y.; Xie, G.; Liu, Z.; and Sun, M. 2023. UltraFeedback: Boosting Language Models with High-quality Feedback. *CoRR*.

Dong, H.; Xiong, W.; Pang, B.; Wang, H.; Zhao, H.; Zhou, Y.; Jiang, N.; Sahoo, D.; Xiong, C.; and Zhang, T. 2024. RLHF Workflow: From Reward Modeling to Online RLHF. *arXiv preprint arXiv:2405.07863*.

Dong, X.; Teleki, M.; and Caverlee, J. 2024. A Survey on LLM Inference-Time Self-Improvement. *arXiv preprint arXiv:2412.14352*.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The LLaMa 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Hong, J.; Lee, N.; and Thorne, J. 2024. Orpo: Monolithic Preference Optimization without Reference Model. *arXiv preprint arXiv:2403.07691*.

Ji, J.; Liu, M.; Dai, J.; Pan, X.; Zhang, C.; Bian, C.; Chen, B.; Sun, R.; Wang, Y.; and Yang, Y. 2023. Beavertails: Towards Improved Safety Alignment of LLM via a Human-Preference Dataset. *Advances in Neural Information Processing Systems*, 36: 24678–24704.

Jiang, D.; Liu, Y.; Liu, S.; Zhao, J.; Zhang, H.; Gao, Z.; Zhang, X.; Li, J.; and Xiong, H. 2023. From Clip to Dino: Visual Encoders Shout in Multi-Modal Large Language Models. *arXiv preprint arXiv:2310.08825*.

Jin, D.; Mehri, S.; Hazarika, D.; Padmakumar, A.; Lee, S.; Liu, Y.; and Namazifar, M. 2023. Data-Efficient Alignment of Large Language Models with Human Feedback Through Natural Language. *arXiv preprint arXiv:2311.14543*.

Khalaf, H.; Verdun, C. M.; Oesterling, A.; Lakkaraju, H.; and Calmon, F. d. P. 2025. Inference-Time Reward Hacking in Large Language Models. *arXiv preprint arXiv:2506.19248*.

Lambert, N.; Morrison, J.; Pyatkin, V.; Huang, S.; Ivison, H.; Brahman, F.; Miranda, L. J. V.; Liu, A.; Dziri, N.; Lyu, S.; et al. 2024. Tulu 3: Pushing Frontiers in Open Language Model Post-Training. *arXiv preprint arXiv:2411.15124*.

Li, T.; Chiang, W.-L.; Frick, E.; Dunlap, L.; Wu, T.; Zhu, B.; Gonzalez, J. E.; and Stoica, I. 2024. From Crowdsourced Data to High-Quality Benchmarks: Arena-Hard and Benchbuilder Pipeline. *arXiv preprint arXiv:2406.11939*.

Li, X.; Zhang, T.; Dubois, Y.; Taori, R.; Gulrajani, I.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Alpacaeval: An Automatic Evaluator of Instruction-Following Models.

Li, Y.; Hu, X.; Qu, X.; Li, L.; and Cheng, Y. 2025. Test-Time Preference Optimization: On-the-Fly Alignment via Iterative Textual Feedback. *arXiv preprint arXiv:2501.12895*.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2023. Let's Verify Step by Step. In *International Conference on Learning Representations*.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training Language Models to Follow Instructions with Human Feedback. *Advances in neural information processing systems*, 35: 27730–27744.

[1]https://github.com/MrYxJ/calculate-flops.pytorch

Qiu, J.; Lu, Y.; Zeng, Y.; Guo, J.; Geng, J.; Wang, H.; Huang, K.; Wu, Y.; and Wang, M. 2024. Treebon: Enhancing Inference-Time Alignment with Speculative Tree-Search and Best-of-N Sampling. *arXiv preprint arXiv:2410.16033*.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *Advances in neural information processing systems*, 36: 53728–53741.

Raschka, S. 2024. *Build a Large Language Model (From Scratch)*. Simon and Schuster.

Röttger, P.; Kirk, H. R.; Vidgen, B.; Attanasio, G.; Bianchi, F.; and Hovy, D. 2023. Xstest: A Test Suite for Identifying Exaggerated Safety Behaviours in Large Language Models. *arXiv preprint arXiv:2308.01263*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

Wang, Z.; Bi, B.; Pentyala, S. K.; Ramnath, K.; Chaudhuri, S.; Mehrotra, S.; Mao, X.-B.; Asur, S.; et al. 2024. A Comprehensive Survey of LLM Alignment Techniques: RLHF, RLAIF, PPO, DPO and More. *arXiv preprint arXiv:2407.16216*.

Wu, S.; Fung, M.; Qian, C.; Kim, J.; Hakkani-Tur, D.; and Ji, H. 2024. Aligning LLMs with Individual Preferences via Interaction. *arXiv preprint arXiv:2410.03642*.

Xu, H.; Sharaf, A.; Chen, Y.; Tan, W.; Shen, L.; Van Durme, B.; Murray, K.; and Kim, Y. J. 2024. Contrastive Preference Optimization: Pushing the Boundaries of LLM Performance in Machine Translation. *arXiv preprint arXiv:2401.08417*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *Advances in neural information processing systems*, 36: 11809–11822.

Yuksekgonul, M.; Bianchi, F.; Boen, J.; Liu, S.; Lu, P.; Huang, Z.; Guestrin, C.; and Zou, J. 2025. Optimizing Generative AI by Backpropagating Language Model Feedback. *Nature*, 639(8055): 609–616.

Zhang, X.; Du, C.; Pang, T.; Liu, Q.; Gao, W.; and Lin, M. 2024. Chain of Preference Optimization: Improving Chain-of-Thought Reasoning in LLMs. *Advances in Neural Information Processing Systems*, 37: 333–356.