Achieving Stronger Generation via Simple Contrastive Tuning

Zhimeng Wang[1], **Pinzheng Wang**[1], **Juntao Li**[1*], **Yibin Chen**[2], **Min Zhang**[1]
[1]Harbin Institute of Technology, Shenzhen, China
[2]Huawei Technologies
zmwang03@gmail.com; chenyibin4@huawei.com;
zhangmin2021@hit.edu.cn;

Abstract

Instruction tuning is widely used to unlock the abilities of Large Language Models (LLMs) in following human instructions, resulting in substantial performance improvements across various downstream tasks. Furthermore, contrastive decoding methods are employed to enhance instruction-tuned models. To further explore the potential of contrastive decoding, we introduce the Contrastive Tuning and Decoding (CTD) framework, which enhances model performance without requiring additional data or significant computational resources. When performing Contrastive Tuning, we optimize a correction model by targeting discrepancies between the original outputs and labels. During Contrastive Decoding, the correction model adjusts the logits of the SFT model using the same input to ensure better adherence to instructions. With the lightweight CTD framework, we refine the behavior of instruction-tuned models, improving their performance on the challenging SUPNATINST dataset with unfamiliar data distributions across various models and prompt formats.

1 Introduction

Recent years (Zhang et al., 2023b) have witnessed remarkable progress in large language models (LLMs). Some LLMs, such as LLaMA (Touvron et al., 2023a), GPT-3 (Brown et al., 2020), and Mistral (Jiang et al., 2023), have acquired general abilities for solving various tasks. Instruction tuning is a crucial technique to enhance the capabilities and controllability of LLMs (Zhang et al., 2023b) by further fine-tuning LLMs with instruction-formatted data, resulting in better performance on downstream tasks.

However, there is a concern that instruction-tuned models show significant improvements mainly on tasks related to the instruction datasets and may not perform as well on others (Gudibande et al., 2023). This limitation indicates that instruction-tuned models struggle to effectively generalize to tasks with entirely different data distributions. Additionally, in practical scenarios, instruction tuning often faces challenges with limited data and tends to overfit, leading to performance degradation after multiple training epochs (Xue et al., 2024; Muennighoff et al., 2024).

To further enhance the performance of LLMs on text generation tasks, some researchers have explored decoding methods in a contrastive manner (Chuang et al., 2023; Kim et al., 2023; Shi et al., 2024; Li et al., 2022). These methods focus on modifying the model's initial output distribution with a specific distracted distribution. Although existing contrastive decoding methods do not require additional training, they lack robustness across various models and scenarios and are sensitive to specific hyperparameters (Kim et al., 2023).

To address this issue and further explore the potential of instruction-tuned models, we propose a novel Contrastive Tuning and Decoding (CTD) framework to achieve stronger generation without additional data or much computational resource, as shown in Figure 1. With a Supervised Fine-Tuned (SFT) model derived from a pre-trained model as the original model, we initialize the correction model using the SFT model and fine-tune it with parameter-efficient methods, utilizing data sampled from the SFT dataset to obtain additional parameters for correction. Finally, we apply Contrastive Decoding, which leverages the differences between the original and correction model's outputs. This process results in significant performance improvements with minimal resource requirements, as we use prompt tuning for contrastive tuning, which is lightweight. We are the first to explore employing contrastive tuning before contrastive decoding, which optimizes the correction model from a unique perspective, resulting in more reliable im-

* Corresponding author.

Findings of the Association for Computational Linguistics: EACL 2024, pages 8986–8999

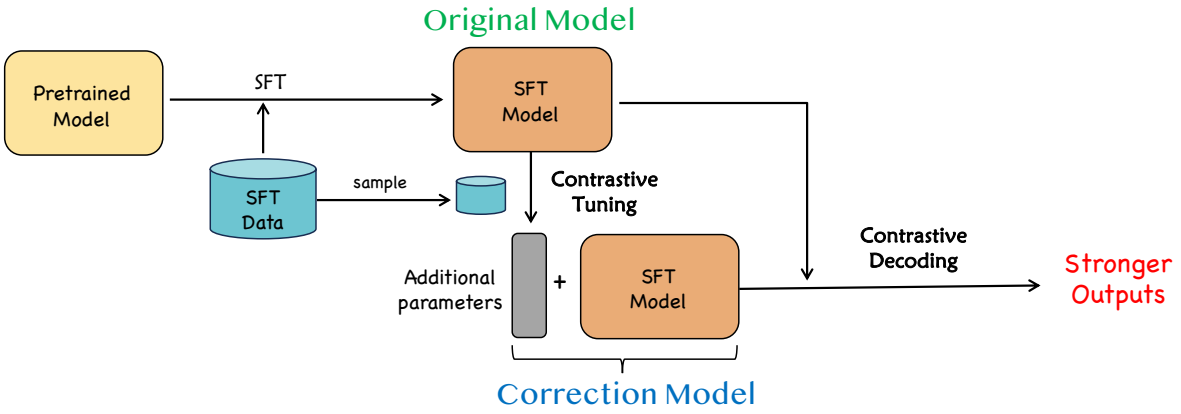


Figure 1: Overview of the Contrastive Tuning and Decoding (CTD) framework. Given an instruction-tuned model as the original model, we employ low-cost Contrastive Tuning with data sampled from the SFT dataset to obtain additional parameters for the correction model. Then, we use Contrastive Decoding to refine the original model's predictions, resulting in stronger outputs.

provements on challenge tasks with unfamiliar data distribution.

In summary, our contributions are as follows: **(1)** We introduce Contrastive Tuning, which develops a specific correction model for the original SFT model. **(2)** We propose Contrastive Decoding with a tunable prefix, which is efficient as it introduces almost no inference latency. **(3)** Our method achieves stronger generation by integrating Contrastive Tuning and Contrastive Decoding into the CTD framework.

2 Related Work

Instruction Tuning Instruction tuning is a method for fine-tuning pre-trained LLMs on a collection of formatted instances presented in natural language (Wei et al., 2021). This approach enables LLMs to follow human instructions (Wei et al., 2022) and perform specific tasks without requiring demonstrations, even for unseen tasks (Chung et al., 2024). However, in a data-constrained scenario, many prior works on LLM (Chung et al., 2024; Brown et al., 2020) show that training an LLM with multiple epochs of repeated data leads to overfitting. Our work also employs multiple rounds of tuning with repeated data and achieves better performance.

Contrastive Decoding & Instructive Decoding The idea of using contrast to improve the text generation performance of LLMs has been studied in various way (Yona et al., 2023; Li et al., 2022; Kim et al., 2023; Liu et al., 2021a; Shi et al., 2024). Our work is mainly motivated by Contrastive Decoding (CD) (Li et al., 2022) and Instructive Decoding (ID) (Kim et al., 2023). CD contrasts expert LLMs with amateur LLMs by taking the difference of model log probabilities to improve generation quality without training. It uses larger LLMs as experts and smaller LLMs as amateurs. However, Instructive Decoding is based on the effect of different instructions. They contrast normal instructions with noise instructions to adjust the logits of the next token prediction and achieve considerable performance.

Prompt-tuning Prompt-tuning (Lester et al., 2021) is a Parameter-Efficient Fine-Tuning (PEFT) method that can efficiently adapt large models over various downstream tasks (Han et al., 2024). It adds a learnable soft prompt (also called a continuous prompt) before the input of models. During training, only soft prompts are updated, and the model parameters are frozen. Prompt-tuning works well on many tasks, but it doesn't perform as effectively as fine-tuning (Liu et al., 2021b) and has been replaced by other PEFT methods such as Lora (Hu et al., 2021), Qlora (Dettmers et al., 2024), Llama adapter (Zhang et al., 2023a), etc. In our work, we use prompt tuning to show the feasibility of tuning in a contrastive way, as this approach is sufficiently straightforward.

3 Motivation

We draw inspiration from Instructive Decoding, which suggests that distracted logits from the SFT model itself can be used to refine its original output (Kim et al., 2023). However, this approach is based on intuition, and the improvement is not stable (Kim et al., 2023). It has been observed that

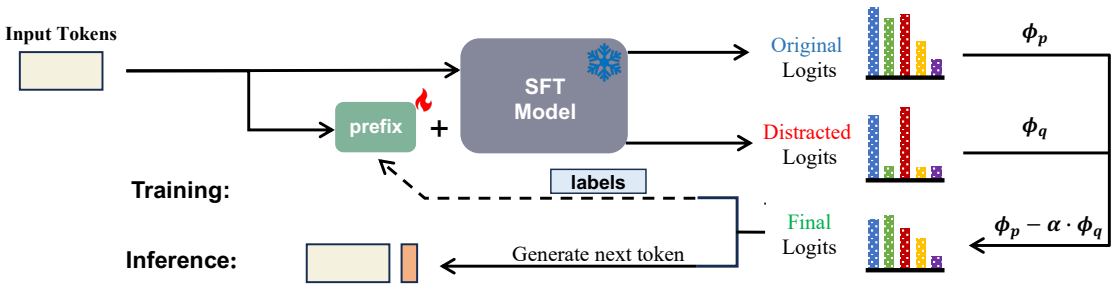


Figure 2: Illustration of Contrastive Tuning and Contrastive Decoding with a soft prefix. The instruction-tuned model serves as the original model, with the soft prefix added to create the correction model. Both models receive the same input. During training and inference, we leverage the differences between the original and corrected logits, optimizing the soft prefix throughout the training process.

Instructive Decoding has misaligned training and decoding objectives. Therefore, we design a novel optimization objective to align the language modeling objectives between training and decoding, enhancing the model's ability to correct its own outputs more effectively. So Contrastive Tuning and Decoding framework can correct high-confidence errors, which Instructive Decoding alone cannot achieve. By incorporating prompt tuning, we avoid storing two sets of model parameters, reducing memory requirements.

4 Methodology

4.1 Overall Framework

We start from the idea that the model can correct its output with just a specific instruction motivated by Instructive Decoding (Kim et al., 2023). In our exploration setting, the correction model is developed from the Supervised Fine-Tuned (SFT) models. Our target is to efficiently construct a correction model based on the given SFT model and leverage these two models to achieve stronger model predictions. Specifically, given a pre-trained autoregressive language model, we treat its SFT version M_Φ on the instruction dataset $\mathcal{D}$ as the original model, where Φ denotes model parameters. The Contrastive Tuning and Decoding (CTD) framework first performs low-cost contrastive tuning on the instruction-tuned model M_Φ to obtain a correction model $M_{\Phi+\theta}$ using an instruction dataset $\mathcal{D}'$ sampled from $\mathcal{D}$. Then, CTD utilizes Contrastive Decoding to achieve stronger model predictions based on the given original and correction model. It refines the behavior of LLMs, achieving stable improvements without extra data through low-cost training. We further introduce the Contrastive Tuning and Instructive Decoding in Section 4.2 and 4.3, as illustrated in Figure 2.

4.2 Contrastive-Tuning

Given a text sequence $\boldsymbol{t}$ of length N, $\boldsymbol{t}_{<i} = (t_1, t_2, \ldots, t_{i-1})$ denotes the sequences proceding the ith token $(i < N)$. The causal language modeling (CLM) objective for tuning a language model parameterized by Φ is defined as minimizing the negative log-likelihood:

$$p_\Phi(t_i|t_{<i}) = \text{SOFTMAX}[z_i],$$

$$\mathcal{L}_{\text{CLM}} = -\mathbb{E}_{t\sim\mathcal{C}}\left[\sum_i \log p_\Phi(t_i|t_{<i})\right],$$

where $p_\Phi(t_i|t_{<i})$ is the predicted probability for token t_i derived from output logits z_i by SOFTMAX function.

For Contrastive Tuning, we employ prompt tuning (Lester et al., 2021; Liu et al., 2021b) by introducing a tunable prefix denoted as θ while keeping the main model frozen. The original model generates logits z_i without θ, and a modified logits z_i^- with θ. We consider the original model with θ as the correction model and treat the logits z_i^- as learnable noise. In order to minimize the noise in the original outputs, we compute the difference between z_i and z_i^-. To optimize the parameter θ for correction, we modify the conventional CLM loss as follows:

$$p_{\Phi,\Phi+\theta}(t_i|t_{<i}) = \text{SOFTMAX}[z_i - \alpha \cdot z_i^-],$$

$$\mathcal{L}_{\text{CLM-CTD}} = -\mathbb{E}_{t\sim\mathcal{C}}\left[\sum_i \log p_{\Phi,\Phi+\theta}(t_i|t_{<i})\right],$$

where α represents the contrast intensity, as shown in the logits subtraction part of Figure 2. Specifically, θ is a tunable prefix for contrast, making this process lightweight and easy to perform.

By learning to refine the original output with learnable noise, we obtain a correction model that can be used to adjust the original model's predictions. In Section 7.3, we also conduct a pilot experiment with LoRA to obtain the parameter θ, demonstrating the generalization of Contrastive Tuning.

4.3 Contrastive Decoding

Motivated by Instructive Decoding (Kim et al., 2023) and Contrastive Decoding (Li et al., 2022), we propose Contrastive Decoding with soft prefix. We develop a correction model based on the soft prefix θ after Contrastive Tuning. During generation with Contrastive Decoding, we still take the difference between the original model and the correction model with the same input, resulting in stronger model predictions, as described in Algorithm 1. Unlike other methods that only decode in a contrasting manner, our approach achieves stronger predictions by aligning the CLM objective between training and contrastive decoding.

5 Experiments

5.1 Experiment Settings

Models Our approach focuses primarily on instruction-tuned models. In this study, we evaluate our method on Alpaca-7B (Taori et al., 2023), Mistral-7B-SlimOrca (Lian et al., 2023b), DeciLM-7B-instruct (Team, 2023), Llama3-8b-alpaca, claude2-alpaca-7B and claude2-alpaca-13B (Chen et al., 2023).

The Alpaca-7B and Llama3-8b-alpaca are trained from Llama-7B and Llama-3-8B on the Alpaca dataset (Taori et al., 2023). claude2-alpaca-7B and claude2-alpaca-13B is trained from Llama-2-7B (Touvron et al., 2023a) and Llama-2-13B (Touvron et al., 2023b) on claude2_alpaca dataset (Chen et al., 2023). Mistral-7B-SlimOrca and DeciLM-7B-instruct are trained from Mistral-7B-v0.1 (Jiang et al., 2023) and DeciLM-7B (Team, 2023) on the SlimOrca dataset (Lian et al., 2023c), a subset of OpenOrca dataset (Mukherjee et al., 2023; Lian et al., 2023a). All the models mentioned above are derived from publicly available checkpoints on Huggingface. These instruction-tuned models encompass three different prompt formats: Alpaca format (Taori et al., 2023), ChatML format, and another commonly used format, detailed in Appendix C. In our experiments, greedy decoding is primarily employed for these models.

Algorithm 1 Contrastive Decoding with Soft Prefix

Input: Original model $\mathcal{M}_\Phi$, correction model $\mathcal{M}_{\Phi+\theta}$, input text sequence I, target sequence length T and intensity coefficient β.

Initialize $t \leftarrow 1$
while $t < T$ **do**
  $z_t \leftarrow \mathcal{M}_\Phi(y_t|I, y_{<t})$
  $\tilde{z}_t \leftarrow \mathcal{M}_{\Phi+\theta}(y_t|I, y_{<t})$
  $y_t = \arg\max(\text{SOFTMAX}[z_t - \beta * \tilde{z}_t])$
  set $t \leftarrow t + 1$
end while

Evaluation and Baselines We examine the models and compare the Contrastive Tuning and Decoding (CDT) framework with two baselines: 1) Original decoding (OD) using greedy decoding strategy and 2) Instructive Decoding (ID). We choose the opposite as the noisy instructions, as it consistently outperforms other Instructive Decoding settings (Kim et al., 2023).

Following Instructive Decoding (Kim et al., 2023), we also utilize SUPNATINST (Wang et al., 2022) to assess the model's performance on unseen task generalization using Rouge-L metrics (Lin, 2004). We evaluate the model on 119 tasks from SUPNATINST, categorized into 12 groups, as outlined in Appendix B.

Additionally, DROP (Discrete Reasoning Over Paragraphs) (Dua et al., 2019) and AlpacaEval (Li et al., 2023) are also used for further evaluations.

DROP is a benchmark where models need to extract relevant information from English text paragraphs and then perform discrete reasoning steps on them. We use the exact match as the metric for DROP, and the results are shown in Appendix E.

AlpacaEval is a widely used benchmark for evaluating large language models (LLMs) on their ability to follow instructions and align with human preferences (Li et al., 2023). It assesses the preference likelihood of an LLM-based evaluator favoring a model's output compared to a GPT-4 baseline, offering a cost-effective alternative to manual human preference annotations. In our evaluation, we utilize the gpt-4-1106-preview version as the automated evaluator.

Training We perform Contrastive Tuning on the above models for 3 epochs using 4 A100-40G GPUs. We use 20 tokens to initialize the soft contrastive prefix. The intensity coefficient α is set to 0.3 for training. The data for Contrastive Tuning is sampled from the same dataset used

Table 1: Zero-shot Rouge-L scores on unseen tasks from a subset of the SUPNATINST dataset are evaluated with different models using Original Decoding (OD), Instructive Decoding (ID), and our Contrastive Tuning and Decoding (CTD) framework.

Model	Methods	Rouge-L												
		Overall	TG	CR	TE	QR	CEC	DAR	AC	KT	DT	WA	OE	GEC
Alpaca-7b	OD	35.97	23.13	26.22	**43.37**	51.10	50.52	30.32	38.74	32.41	39.07	21.55	29.11	79.89
	ID	37.01	25.09	26.60	42.34	58.67	**50.89**	30.43	39.48	33.03	39.01	**22.78**	28.82	79.52
	CTD	**38.73**	**26.49**	**28.59**	42.84	**60.19**	50.62	**32.21**	**46.93**	**36.09**	**39.13**	22.24	**29.60**	**80.30**
Mistral-7B-SlimOrca	OD	53.58	32.38	53.72	66.43	**58.02**	70.47	64.23	54.96	41.54	41.44	44.21	58.63	87.66
	ID	54.10	33.44	53.79	66.65	58.00	70.87	65.42	56.76	42.18	**41.62**	43.89	58.10	**88.35**
	CTD	**56.57**	**34.46**	**56.49**	**69.38**	57.81	**71.92**	**71.85**	**61.26**	**46.36**	41.17	**47.48**	**66.35**	87.72
DeciLM-6b-instruct	OD	53.41	34.23	43.87	70.03	62.90	69.20	68.25	36.38	42.84	46.58	56.91	67.35	**86.47**
	ID	54.49	34.44	45.81	71.15	**63.17**	69.91	69.36	39.67	41.74	**46.75**	58.40	68.77	85.98
	CTD	**60.46**	**35.84**	**60.30**	**76.65**	61.25	**73.38**	**74.45**	**57.61**	**51.67**	46.13	**61.72**	**70.36**	86.25
Llama3-8b-alpaca	OD	32.38	25.88	18.98	35.76	57.85	45.32	21.06	23.95	24.77	40.31	27.17	31.54	88.56
	ID	32.55	26.07	18.73	36.01	**58.27**	44.76	21.91	23.97	24.78	40.24	**27.73**	33.54	88.62
	CTD	**36.05**	**27.93**	**23.75**	**39.74**	58.12	**50.36**	**29.01**	**33.09**	**25.96**	**40.56**	27.35	**37.97**	**88.64**
claude2-alpaca-7B	OD	34.46	28.23	25.31	33.19	58.96	52.47	23.46	35.80	27.07	40.51	21.42	21.01	**86.44**
	ID	35.08	28.80	**25.57**	**34.80**	59.37	54.69	24.07	35.78	27.92	39.98	21.14	20.69	86.07
	CTD	**35.72**	**28.88**	24.97	33.77	**60.07**	**55.17**	**26.33**	**39.97**	**30.91**	**40.53**	**21.50**	**22.71**	85.92
claude2-alpaca-13B	OD	40.36	33.07	28.65	37.43	59.66	59.99	34.03	45.28	39.09	43.87	**34.34**	26.63	**88.81**
	ID	40.14	33.09	31.06	34.85	61.44	60.34	34.20	44.16	38.87	43.68	34.09	26.17	87.45
	CTD	**45.11**	**35.35**	**34.67**	**45.11**	**63.53**	**62.10**	**37.88**	**55.97**	**41.64**	**44.13**	33.98	**35.38**	87.26

by the instruction-tuned model. The batch size is set as 4×8. We optimize the prefix using the AdamW (Loshchilov and Hutter, 2017) optimizer with a learning rate of 9e-4.

5.2 Performance on Unseen Task generalization

Result Overview Table 1 displays the results of applying the Contrastive Tuning and Decoding (CTD) framework to these instruction-tuned models. CTD consistently outperforms the original decoding and the Instructive Decoding methods, particularly for the DeciLM-7b-instruct and claude2-alpaca-13B models. CTD achieves remarkable performance in adhering to instructions when facing these unseen, challenging tasks.

Winning Rate We also assess the tasks in which CTD outperformed the baseline in the SUPNATINST dataset, measured by the Rouge-L score, as depicted in Figure 3. CTD consistently outperforms original decoding and Instructive Decoding methods across various tasks, model sizes, and prompt formats.

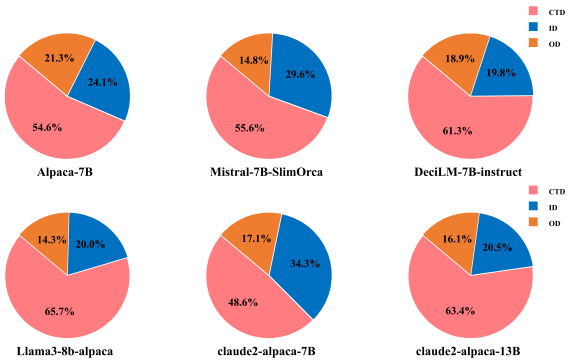


Figure 3: Comparative winning rates among Original Decoding (OD), Instructive Decoding (ID), and Contrastive Tuning and Decoding (CTD) across 119 tasks in the SUPNATINST dataset.

Text Generation Quality We evaluate our CTD framework on the AlpacaEval dataset with gpt-4-1106-preview as the evaluator, and the results are shown in Figure 4. The CTD framework performs

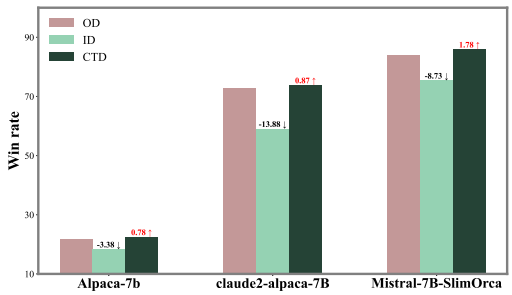


Figure 4: Winning rates of Original Decoding (OD), Instructive Decoding (ID), and Contrastive Tuning and Decoding (CTD) compared to baseline in the AlpacaEval dataset.

better than both Original Decoding (OD) and Instructive Decoding (ID) on this benchmark, highlighting its robustness and effectiveness in improving text generation quality. While ID performs reasonably well on the SUPNATINST tasks, its question-answering quality deteriorates. In contrast, CTD consistently enhances generation quality, making it suitable for real-world applications.

5.3 Implementation Details

Training Epochs and Dataset Proportion The choice of epochs for Contrastive Tuning is critical, as excessive training can lead to overfitting. As shown in Figure 5, one or two epochs of contrastive

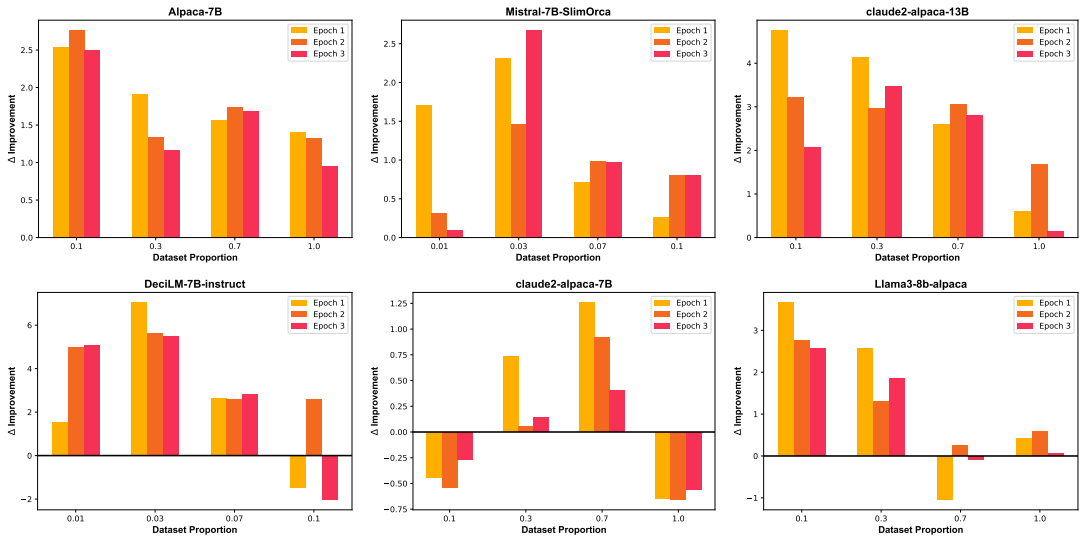


Figure 5: Improvement of Contrastive Tuning and Decoding (CTD) framework compared to Original Decoding (OD) for different models on the SUPNATINST dataset across various sampled data proportions and training epochs.

tuning yield the best results for most models, with performance dropping after the third epoch.

Contrastive Tuning is efficient due to its minimal training epochs and parameters, as well as the requirement for only a small amount of data sampled from the original SFT dataset, shown in Figure 5. We sample data proportions ranging from 0.1 to 1.0 for Alpaca-7B, claude2-alpaca-13B, claude2-alpaca-7B, and llama3-8b-alpaca. For Mistral-7B-SlimOrca and DeciLM-7B-instruct, data proportions of 0.01 to 0.1 are sampled. Typically, using data proportions of 0.1 or 0.01 leads to significant improvements. It's important to note that overfitting can occur with excessive data due to our limited tunable parameters.

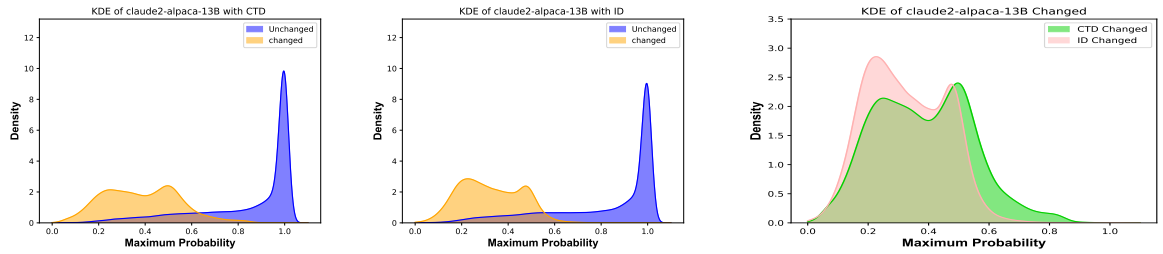
The Coefficient α The coefficient α controls the intensity of contrast in both training and decoding. The relationship between the coefficient α in Contrastive Tuning and Decoding is significant. We train all models with $\alpha = 0.3$, and during decoding, larger α values initially show better performance, shown in Appendix D. As α continues to increase, performance improves until α reaches 0.5, after which performance begins to decline.

However, there are some exceptions. The performance of claude2-alpaca-7B does not decline with larger α values, and DeciLM-7B-instruct achieves its best performance at $\alpha = 0.3$. Theoretically, the performance should be optimal when the training and inference α values are the same. We believe these anomalies arise due to differences between the initial SFT and our Contrastive Tuning settings (as we do not have access to the original training code), leading to some discrepancies. Nevertheless, this also demonstrates the robustness of our method.

5.4 Analysis

To further illustrate the impact of the correction model on the original instruction-tuned model, we analyze the prediction tokens that are changed or unchanged by the correction model. Figure 6 shows the density of the maximum probability from the token distribution of original predictions on SUPNATINST tasks, highlighting tokens that are changed or remain unchanged by the correction model. We notice that confident base predictions typically stay unchanged, whereas those lacking confidence are frequently modified through Contrastive Decoding. This tendency is consistent with Instructive Decoding (ID) (Kim et al., 2023), demonstrating how Contrastive Decoding influences the original outputs.

However, the Contrastive Tuning and Decoding (CTD) framework performs significantly better than Instructive Decoding. We compare the density of the original probabilities of the changed tokens between CTD and ID. The KDE plot for ID shows that the maximum probabilities of the changed tokens are confined to a narrower range, whereas the KDE plot for CTD spans a wider range of probabilities. When it comes to high-confidence false predictions, Instructive Decoding fails to correct them, whereas CTD effectively adjusts these predictions, leading to stronger generation. This indicates



(a) CTD and ID: Unchanged vs. Changed

(b) Changed Maximum Probability: CTD vs. ID

Figure 6: Kernel density estimation (KDE) of predictions from the original model (claude2-alpaca-13B) on the SUPNATINST dataset. 'Maximum Probability' refers to the highest value in the token distribution derived from the original model. 'Changed' denotes tokens changed by the correction model, while 'Unchanged' represents tokens that remain unchanged after the correction.

a greater correction ability of the CTD framework, which we attribute to Contrastive Tuning.

Claude2-alpaca-13B

Method	Corrections Times	Number of tokens	Tokens >10 times
ID-before	6063	1318	70
CTD-before	11154	1411	116
ID-after	6063	1746	89
CTD-after	11154	2976	131

Figure 7: Token correction frequency and variety for Claude2-alpaca-13B on the SUPNATINST dataset. ID/CTD-before and ID/CTD-after represent corrections before and after contrastive decoding, where Correction Times indicates the total number of corrected tokens, and Number of Tokens refers to the distinct tokens affected.

Figure 7 shows the frequency of token changes and the variety of tokens corrected for Claude2-alpaca-13B on the SUPNATINST task. ID changes the original prediction 6063 times, correcting 1318 unique tokens to 1746 unique tokens. In contrast, CTD makes 11154 corrections, adjusting 1411 unique tokens to 2976 unique tokens. This demonstrates that CTD not only performs more changes on token predictions but also covers a wider range of token types.

6 Ablation Study

6.1 Processed by Two sets of Model Parameters

The CTD framework processes the input using two sets of main model parameters (the original model and the correction model), achieving better outputs than vanilla decoding, which uses only one set of model parameters. Although CTD employs an additional process with two sets of model parameters, it significantly outperforms Instructive Decoding, which also uses two sets of model parameters but shows less improvement and even degradation, as shown in Figure 1. This demonstrates that the effectiveness of CTD is not merely due to the additional process with two sets of model parameters but rather the strength of Contrastive Tuning.

6.2 Impact of the Additional Prefix

We compare Prompt Tuning (Liu et al., 2021b; Lester et al., 2021) with the CTD framework, both using a tunable 20-token prefix, to explore the impact of the additional prefix on the model's performance.

As shown in Figure 8, prompt tuning with a 20-token prefix tends to either overfit or underfit, resulting in poor performance on the SUPNATINST dataset. However, our CTD framework, which employs a 20-token prefix for Contrastive Tuning, consistently improves performance with minimal training data.

In summary, the combination of Contrastive Decoding and Contrastive Tuning is essential for achieving optimal results. Simply using contrastive decoding or adding a tunable prefix alone does not yield significant improvements.

6.3 Training Data

Contrastive Tuning does not inject new knowledge into the model; rather, it teaches the model to correct its own outputs with learnable noise. This process can be viewed as aligning the original language modeling objective with Contrastive Decoding methods.

We use data sampled from the original SFT

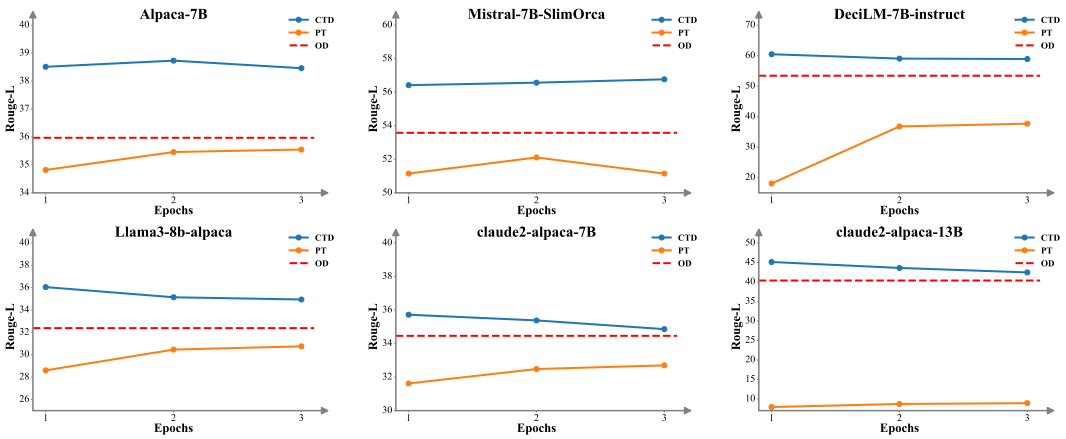


Figure 8: Impact of the tunable 20-token prefix for Prompt Tuning (PT) and the Contrastive Tuning and Decoding (CTD) framework on the SUPNATINST dataset, with Original Decoding (OD) as the baseline. The same data proportion and training settings are used across methods.

dataset for Contrastive Tuning to avoid introducing unfamiliar data that may hinder optimization.

To illustrate this, we conducted an experiment using additional data for Contrastive Tuning on the Alpaca-7B. The results, shown in Figure 9, indicate that attempting to introduce new knowledge through contrastive tuning leads to performance degradation.

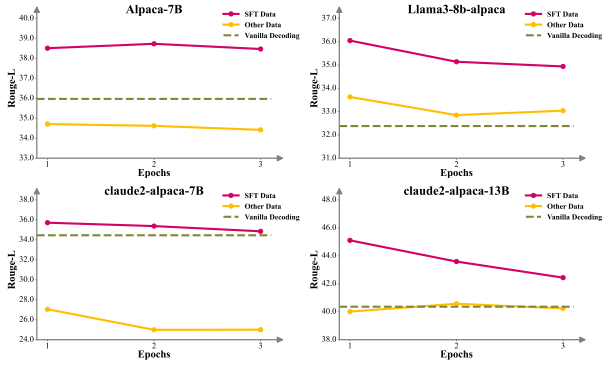


Figure 9: Different datasets used for contrastive tuning. SFT Data refers to tuning with previous SFT data, while Other Data refers to tuning with unknown out-of-distribution data.

7 Discussion

7.1 Training Cost & Inference Latency

Although our CDT framework is effective, the training cost is critical for its practical application. Firstly, we achieve data efficiency by sampling a small amount of data from the SFT data. Secondly, we are committed to finding the most parameter-efficient methods that use the least amount of memory. Therefore, we employ prompt tuning for Contrastive Tuning, which uses the least memory among parameter-efficient methods. When performing Contrastive Decoding, we just double the input batch and put the tunable prefix before half of the input, which only need one set of main model parameters in the GPU memory. In contrast, other PEFT methods, such as LoRA (Hu et al., 2021), require storing two sets of model parameters in memory simultaneously.

We use 20 tokens to initialize the input prefix for both 7B, 8B, and 13B models, which tunes about 0.0012%, 0.0009%, and 0.0008% of the model parameters.

Compared to the original decoding method, Contrastive Decoding introduces only 20 tokens of latency and memory that doubles the input batch sizes, which is equal to Instructive Decoding (Kim et al., 2023) and affordable.

7.2 Difference between Contrastive-Tuning and Contrastive learning

To further understand Contrastive Tuning, it is necessary to discuss the differences between Contrastive Tuning and Contrastive learning (Gao et al., 2021).

Contrastive learning enables models to map similar instances close together in a latent space while pushing apart those that are dissimilar by the Contrastive learning loss (Gao et al., 2021). Both supervised and self-supervised contrast learning need designed tasks for similar and dissimilar data.

In contrast, Contrastive Tuning aims to create a correction model that modifies the instruction-tuned model's output from the same input and computes the cross-entropy loss with the labels.

7.3 LoRA for Contrastive Tuning

To further demonstrate the effectiveness of the Corrector Framework, we employ LoRA (Hu et al., 2021) for Contrastive Tuning, with the results presented in Table 2.

CTD-LoRA also performs well on MMLU (Hendrycks et al., 2021) and SUPNATINST, showing the potential of our CTD framework. However, CTD-LoRA requires more memory because LoRA Contrastive Tuning needs two sets of main model parameters compared to the prompt tuning CTD, making it not as efficient as our Contrastive prompt tuning.

	MMLU	SUPNATINST
OD	33.25	35.46
CTD-LoRA	**35.60**	**37.29**

Table 2: Performance of CTD-LoRA on MMLU and SUPNATINST

8 Conclusion

In this paper, we propose the Contrast Tuning and Decoding (CTD) framework for instruction-tuned models to achieve stronger generation with minimal cost and ensure better generalization on challenging tasks with unfamiliar data distribution. Compared to Instructive Decoding, we demonstrate the necessity of Contrastive Tuning. Through our CTD framework, instruction-tuned models with different models, prompt formats and SFT data show stable improvement on various text generation tasks.

Limitations Our experiments show the feasibility of the Contrastive Tuning and Decoding (CTD) framework. However, there is much more to do, given our limited time and resources. Firstly, the impact of the intensity parameter α for contrastive tuning is not clear, nor is how it influences the optimization process. Secondly, we do not yet know the minimum data required for contrastive tuning sampled from SFT data to further simplify the CTD framework. Finally, we have only tried prompt tuning and LoRA for contrastive tuning. Other parameter-efficient tuning methods may have more potential.

Acknowledgments

We want to thank all the anonymous reviewers for their valuable comments. This work was supported by the National Science Foundation of China (NSFC No. 62206194 and 62276077), the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20220488), Young Elite Scientists Sponsorship Program by CAST (2023QNRC001), and Huawei Technologies.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Lichang Chen, Khalid Saifullah, Ming Li, Tianyi Zhou, and Heng Huang. 2023. Claude2-alpaca: Instruction tuning datasets distilled from claude. https://github.com/Lichang-Chen/claude2-alpaca.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*.

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Taehyeon Kim, Joonkee Kim, Gihun Lee, and Se-Young Yun. 2023. Distort, distract, decode: Instruction-tuned model can refine its response from noisy instructions. *arXiv preprint arXiv:2311.00233*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023a. Openorca: An open dataset of gpt augmented flan reasoning traces. `https://https://huggingface.co/Open-Orca/OpenOrca`.

Wing Lian, Bleys Goodson, Guan Wang, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023b. Mistralslimorca: Mistral-7b model instruct-tuned on filtered, corrected, openorcav1 gpt-4 dataset.

Wing Lian, Guan Wang, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023c. Slimorca: An open dataset of gpt-4 augmented flan reasoning traces, with verification.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021a. Dexperts: Decoding-time controlled text generation with experts and anti-experts. *arXiv preprint arXiv:2105.03023*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021b. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. 2024. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *Preprint*, arXiv:2306.02707.

Chenyu Shi, Xiao Wang, Qiming Ge, Songyang Gao, Xianjun Yang, Tao Gui, Qi Zhang, Xuanjing Huang, Xun Zhao, and Dahua Lin. 2024. Navigating the overkill in large language models. *arXiv preprint arXiv:2401.17633*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

DeciAI Research Team. 2023. Decilm-7b-instruct.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

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

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Fuzhao Xue, Yao Fu, Wangchunshu Zhou, Zangwei Zheng, and Yang You. 2024. To repeat or not to repeat: Insights from scaling llm under token-crisis. *Advances in Neural Information Processing Systems*, 36.

Gal Yona, Or Honovich, Itay Laish, and Roee Aharoni. 2023. Surfacing biases in large language models using contrastive input decoding. *arXiv preprint arXiv:2305.07378*.

Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023a. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023b. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

A Open-Source Models Utilized in Our Study

We provide a list of the open-source models used in our work, as shown in Table 3. These models are popular checkpoints from Huggingface, instruction-tuned on open-source datasets, and have demonstrated strong performance on downstream tasks. While we are unable to obtain the training code for these models, this limitation underscores the robustness and practicality of our approach.

B Overview of the SUPNATINST Dataset

SUPNATINST (Wang et al., 2022) is a large-scale dataset comprising over 1,600 natural language processing (NLP) tasks designed to enhance model generalization through declarative instructions.

Each task in SUPNATINST includes a 'Definition' prompt, which serves as an instructional guide. For zero-shot evaluations, only the 'Definition' is provided. Following the approach from Instructive Decoding (Kim et al., 2023), our experiments focus on the English portion of the dataset, evaluating 100 instances per task as outlined by (Wang et al., 2022). This subset consists of 119 evaluation tasks, categorized into the 12 groups shown in Table 4.

Abbreviation	Task Category
AC	Answerability Classification
CEC	Cause-Effect Classification
DT	Data-to-Text
GEC	Grammar Error Correction
CR	Coherence Resolution
KT	Keyword Tagging
DAR	Dialogue Act Recognition
OE	Overlap Extraction
QR	Question Rewriting
TE	Textual Entailment
TG	Title Generation
WA	Word Analogy

Table 4: Task Categories in the SUPNATINST Dataset

C Prompt Formats

Our experiments cover three prompt formats: Alpaca format, ChatML format, and another commonly used format, as shown in Table 6.

D The coefficient α

Figure 11 illustrates the impact of the coefficient α on model performance during Contrastive Decoding.

Model	Huggingface Model ID
Alpaca-7B	chavinlo/alpaca-native
DeciLM-7B-instruct	Deci/DeciLM-7B-instruct
Mistral-7B-SlimOrca	Open-Orca/Mistral-7B-SlimOrca
Llama3-8b-alpaca	lainshower/Llama3-8b-alpaca
claude2-alpaca-7B	umd-zhou-lab/claude2-alpaca-7B
claude2-alpaca-7B	umd-zhou-lab/claude2-alpaca-7B

Table 3: Open-Source Model used In Our Work

E Performance on DROP

DROP (Discrete Reasoning Over Paragraphs) (Dua et al., 2019) is a reading comprehension benchmark designed to challenge language models with 96,000 crowdsourced, adversarially created questions. Unlike previous datasets, DROP requires language models to perform discrete operations such as addition, counting, and sorting by resolving references across multiple positions within a paragraph. This demands a deeper and more comprehensive understanding of the paragraph's content.

We evaluate the Contrastive Tuning and Decoding framework on the DROP dataset, where it also outperforms both Original Decoding and Instructive Decoding, as shown in Figure 10 and Table 5.

Model	Methods	DROP Exact Match
Alpaca-7B	OD	14.01
	ID	14.44
	CTD	**16.39**
DeciLM-7B-instruct	OD	47.00
	ID	46.75
	CTD	**49.52**
claude2-alpaca-7B	OD	18.08
	ID	17.88
	CTD	**19.10**
claude2-alpaca-13B	OD	13.53
	ID	12.91
	CTD	14.51
Mistral-7B-SlimOrca	OD	46.11
	ID	46.48
	CTD	46.64

Table 5: Performance on DROP using Original Decoding (OD), Instructive Decoding (ID), and Contrastive Tuning and Decoding (CTD) framework with Exact Match.

F Comparison and Integration of the CTD Framework, Instructive Decoding, and Prompt Tuning

Instructive Decoding is a variant of contrastive decoding that generally outperforms other contrastive methods. All contrastive decoding approaches aim to leverage distracting output logits to enhance the original output. However, a significant limitation of these methods is the lack of clear evidence explaining why distracting logits effectively influence the original output, as the concept has mostly been accepted based on intuition.

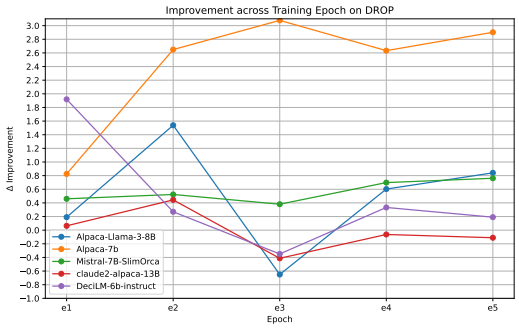


Figure 10: Impact of Training Epochs on Model Performance for the DROP Dataset

One key issue with contrastive decoding methods is the misalignment between language modeling objectives during training and decoding. To address this, our Contrastive Tuning technique bridges the gap between the training and decoding phases.

Our approach is unique in training a correction model with only a few parameters, based on the source model and guided by a clear optimization objective. This correction model refines the source model's output by verifying correct tokens and adjusting incorrect ones. Through contrastive tuning, the model gains this corrective ability—an innovation that has not been explored before. Prior methods, lacking such a well-defined objective, leave the distracting logits ambiguous about when to correct or affirm the original outputs.

Instructive Decoding is primarily effective for tasks that align closely with the distribution of the SFT data, which limits its improvement potential. By contrast, the CTD framework has been evalu-

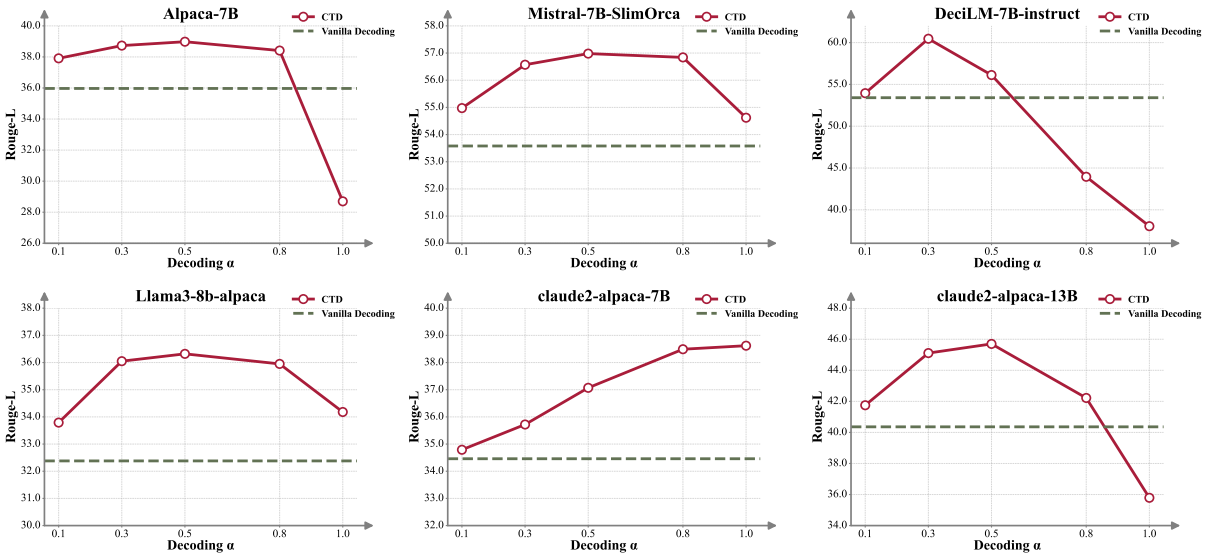


Figure 11: Impact of the coefficient α on decoding performance (measured by Rouge-L) for various models on the SUPNATINST dataset.

ated across multiple models trained with diverse SFT data and tested on more challenging and unfamiliar tasks. When applied to AlpacaEval, Instructive Decoding shows a performance drop compared to original decoding methods, revealing its weak generalization and limited practical value. Nevertheless, it provides useful insights by indicating that models can potentially recognize their own errors and self-correct. We are the first to unlock this potential through contrastive tuning.

Prompt tuning, a parameter-efficient fine-tuning (PEFT) method, is currently underutilized due to its comparatively lower performance. Within the CTD framework, prompt tuning serves as one method of implementing contrastive tuning. Preliminary experiments with LoRA in our study suggest that other PEFT methods also show promise.

However, we believe that prompt tuning is what makes our method stand out, as it enables contrastive tuning to be highly efficient. Prompt tuning allows the model to store only a single set of parameters in memory during both contrastive tuning and contrastive decoding. In contrast, methods like LoRA would require additional parameters and would need to store two sets of model parameters during contrastive tuning. With contrastive prompt tuning, updating as little as 0.0012% of the parameters for a 7B model leads to outstanding results. We attribute this efficiency to the model's inherent ability to self-correct (as demonstrated with Instructive Decoding), combined with our clear optimization objective that activates this capability at minimal cost.

G Decoding Strategy and Evaluation Metric

We employ greedy decoding due to the diverse nature of the SUPNATINST dataset, which consists of 119 tasks spanning generation, classification, and question-answering. While sampling-based decoding can enhance output diversity, its inherent randomness makes it less suitable for tasks like classification and question-answering.

Initially, we consider both Rouge-L and exact match metrics for evaluation. However, our research is primarily focused on out-of-distribution text generation, classification, and question-answering tasks. Even though some tasks may have a constrained output space, the outputs do not always perfectly match the reference answers. As a result, the exact match is excluded as it does not accurately reflect model performance in this context.

Prompt Format
Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. ### Instruction: {instruction} ### Input: {input} ### Response: {output}
<|im_start|>system {system_prompt}<|im_end|> <|im_start|>user {user_prompt}<|im_end|> <|im_start|>assistant {output}<|im_end|>
System: {system_prompt} ### User: {user_prompt} ### Assistant: {output}

Table 6: Prompt Formats Used In Our Work