Transferable Post-training via Inverse Value Learning

Anonymous ACL submission

Abstract

As post-training processes utilize increasingly large datasets and base models continue to grow in size, the computational demands and implementation challenges of existing algorithms are escalating significantly. In this paper, we propose modeling the changes at the logits level during post-training using a separate neural network (i.e., the value network). After training this network on a small base model using demonstrations, this network can be seamlessly integrated with another pre-trained models during inference, enabling them to achieve similar capability enhancements. We systematically investigate the best practices for this paradigm in terms of pre-training weights and connection schemes. We demonstrate that the resulting value network has broad transferability across pre-trained models of different parameter sizes within the same family, models undergoing continuous pre-training within the same family, and models with different vocabularies across families. In certain cases, it can achieve performance comparable to full-parameter fine-tuning. Furthermore, we explore training methods to enhance transferability, which effectively improve the transfer performance of the value model across models of various parameter scales and prevent overfitting to the base model used during training.

1 Introduction

Post-training of large language models (LLMs), which can be refers to as fine-tuning a pre-trained model for helpfulness, honesty, and harmlessness (Askell et al., 2021), is becoming increasingly important (Dubey et al., 2024). A well-designed post-training process can maximize the potential injected during pre-training while remaining aligned with human intentions and values.

However, post-training has become increasingly intricate (Dubey et al., 2024; Yang et al., 2024; Adler et al., 2024). A well-designed post-training procedure typically consists two main stages: supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022). Each stage involving iterative data synthesis, quality control, and training pipelines (Cao et al., 2024), which collectively introduce significant complexity to the model training workflow. Furthermore, as model sizes continue to grow, the processes of both sampling and training larger-scale models are becoming significantly more resource-intensive.

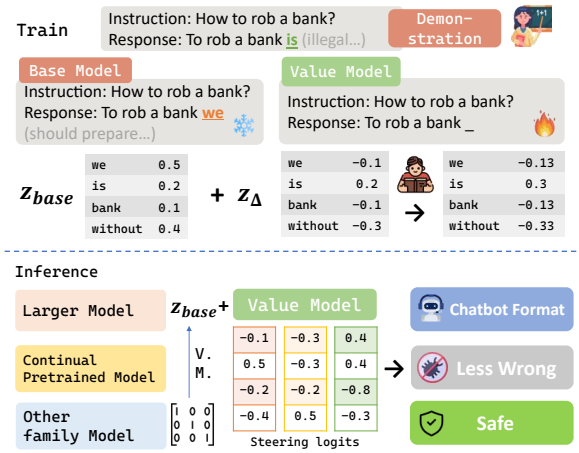


Figure 1: Illustration of Inverse Value Learning. V.M. denotes Vocabulary Mapping.

To address these challenges, we propose a novel framework for transferable post-training that efficiently adapts models of varying sizes and families with minimal reconfiguration. This framework distill the post-training process to another network by leveraging the *logits space* as a shared interface for model adaptation and capability transfer. Unlike parameter or representation spaces, the logits space remains more consistent across different models and can serve as a universal channel for communication (Hinton, 1999). By adjusting the base model's logits during decoding, we can adapt any other base models to new tasks without modifying

its original parameters. This allows for efficient model adaptation, avoiding the resource-intensive process of full fine-tuning.

Specifically, we propose *transferable post-training via inverse value learning*, a framework that captures changes at the logits level using a separate neural network, referred to as the value network. This network is trained on a smaller base model using demonstration data and is designed to predict the residual logits that capture the necessary adaptations for post-training process. During inference, the value network can be seamlessly integrated with various pre-trained base models, enabling them to adopt the learned adjustments without requiring further training, thus facilitating efficient transfer of model capabilities.

We systematically investigate practical implementation for this framework. Specifically, we first study the importance of pre-trained weights in modeling the residual logits, then we examine two architectures for integrating the value network with pre-trained models: Cascade and Residual connections. Our experiments reveal that the residual connection scheme, where the value network predicts residuals based solely on previous text inputs, demonstrates superior transferability and efficiency. To improve the generalization ability of the value network, we incorporate regularization techniques, such as norm constraints, to mitigate overfitting to the base model during training. Additionally, we introduce a vocabulary mapping algorithm to facilitate effective cross-vocabulary transfer.

We assess our approach on multiple datasets and tasks, including general instruction following, zero-shot capabilities, and few-shot learning scenarios. Our results show that the value network exhibits board transferability across pre-trained models of varying parameter sizes within the same model family, models undergoing continual pre-training, and even across different model families. In certain cases, its performance closely approximates that of full-parameter fine-tuning, highlighting the method's potential for efficient and practical application without the need for extensive model fine-tuning.

Our main contributions are as follows:

1) To address the complexity and resource demands of the post-training process, we propose a transferable post-training framework based on the logits space. By modeling the residuals in the logits space, we can transfer post-training capabilities across models of varying sizes and families without altering the original parameters and additional training.

2) We systematically studied the implementation of this framework, including pretrained weights, connection schema, regularization methods and cross vocabulary transfer algorithm. Through comprehensive experiments across diverse settings and datasets, we demonstrate the effectiveness of our proposed framework and board transferability of the trained value models.

2 Problem Statement and Background

The post-training of LLMs has become increasingly important for enhancing model performance and adapting to specific tasks (Dubey et al., 2024). However, as model sizes continue to grow, the computational and time costs of training the entire model have also increased significantly. In some cases, updating all the weights of the model is impractical, prompting researchers to explore alternative methods, such as updating additional low-rank parameters (Hu et al., 2022) or interfering the representations (Wu et al., 2024) while frozen all the original pre-trained weights.

A key challenge is that the parameter space and representation space of different models cannot be directly shared due to differences in structure, hidden states dimensions, and etc. In contrast, the logits space has better sharing properties and can serve as a communication channel for model interaction, enabling the transfer of capabilities between models (Hinton, 1999). Specifically, the impact of post-training on the logits of a pretrained model can be formally expressed as:

$$\log p_{\text{post}}(y_t|\mathbf{x}, \mathbf{y}_{<t}) = \log p_{\text{base}}(y_t|\mathbf{x}, \mathbf{y}_{<t}) + \boxed{\log p_{\Delta}(y_t|\mathbf{x}, \mathbf{y}_{<t})}$$

where $\log p_{\Delta}$ represents the incremental change in the logits space during post-training. A series of studies have focused on exploring $\log p_{\Delta}$. Mitchell et al. (2023) and Liu et al. (2024) obtained $\log p_{\Delta}$ by comparing an existing instruction-following model with its corresponding base pre-trained model, i.e., $\log p_{\Delta}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) = \log \frac{p_{\pi^*}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})}{p_{\text{base}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})}$. This difference can be regarded as an advantage function (Mitchell et al., 2023), or an implicit reward signal (Rafailov et al.,

2024). Furthermore, Khanov et al. (2024) directly uses predefined reward functions as $\log p_\Delta$ to guide the Top-K candidates of $\log p_{\text{base}}$ during decoding, achieving controllable text generation.

Different from the inference-time fusing methods and rely on additional outcome-based reward signal. In this paper, we focus on how to model $\log p_\Delta$ using a separate neutral network plugged during training, thereby efficiently achieving the transfer and enhancement of model capabilities.

3 Preliminary Study

3.1 Task Definition

Let x denote an input sample. We utilize a pre-trained model $\pi_{\text{base}}(\boldsymbol{x};\theta_1)$ with fixed parameters θ_1 that outputs logits $\boldsymbol{z}_{\text{base}}$:

$$\boldsymbol{z}_{\text{base}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t}) = \pi_{\text{base}}(\boldsymbol{x};\theta_1). \tag{1}$$

To adapt this model to a new task without altering θ_1, we introduce a delta model $\pi_\Delta(\circ;\theta_2)$ with parameters θ_2, producing adjustment logits:

$$\boldsymbol{z}_\Delta(y_t|\boldsymbol{x},\boldsymbol{y}_{<t}) = \pi_\Delta(\circ;\theta_2). \tag{2}$$

The final logits are computed by combining the pre-trained model's logits with the delta model's output:

$$\begin{aligned}\boldsymbol{z}_{\text{post}}(y_t|\mathbf{x},\mathbf{y}_{<t}) &= \text{stop_gradient}(\boldsymbol{z}_{\text{base}}(y_t|\mathbf{x},\mathbf{y}_{<t}))\mathbf{W} \\ &\quad + \boldsymbol{z}_\Delta(y_t|\mathbf{x},\mathbf{y}_{<t}).\end{aligned} \tag{3}$$

where stop_gradient$(\cdot)$ indicates that gradients are not propagated through $\boldsymbol{z}_{\text{base}}$ during back-propagation, and $\boldsymbol{W} \in \mathbb{R}^{n\times m}$ is a learnable transition matrix to mapping the size-n pre-trained vocabulary to the size-m vocabulary of the value model.

Let p_l represent the label probabilities. Our objective is to minimize the Cross-Entropy loss function:

$$\mathcal{L} = \text{CE}(\boldsymbol{z}_{\text{post}}, p_l). \tag{4}$$

By optimizing θ_2, we aim to align $\log p_{\text{post}}$ with the target label distribution p_l, thereby enhancing performance on the new task while keeping the pre-trained parameters θ_1 unchanged. Importantly, this process is equivalent to inversely learning an action-value function $Q(s,a)$ from demonstrations and π_{base}, where the states are $y_{<t}$ and the actions are y_t. In this context, actions with a higher $\boldsymbol{z}_{\text{post}}$ indicates that the value function encourages these actions at state s based on the training demonstrations. Therefore we refer to this process as ***inverse value learning***.

Advantages. This task formulation has a set of advantages, including:

1. The decoupling of delta logits allows for more thorough investigation when errors occur, enabling us to attribute them to either the pre-training process or the post-training process.

2. It is feasible to either Weak-to-strong (Burns et al., 2024) or Strong-to-weak generations by introducing different amount of parameters to parameterize π_{base} and π_Δ (Mitchell et al., 2023). Since the delta model can be trained with small models and plugged to the larger models in Weak-to-strong setting ($||\theta_1|| > ||\theta_2||$), the overall training time can be reduced compared to training a strong model directly.

3. The formulation is compatible with a variety of loss functions, including KL-based teacher-student distillation, pairwise optimization and standard cross-entropy adapted in this paper.

4. Operating in the logit space allows for cross-model guidance in a plug-and-play manner, enabling capability transfer across models with the same or different vocabularies, thus minimizing the need for redundant training.

Limitations. This formulation can introduce additional inference cost, as it requires obtaining logits from both the pre-trained and value models, resulting in extra computational overhead. Nevertheless, with more computational resources available and residual architecture adapted, which will be discussed in Section 4.1, this will not introduce additional inference latency. Moreover, techniques like speculative decoding (Chen et al., 2023; Leviathan et al., 2023) could be employed to further optimize inference time. We leave this optimization to future work.

3.2 Linear Probes Are Insufficient for Inverse Value Learning

Given the recent prominence of the surface alignment hypothesis (Zhou et al., 2023a; Lin et al., 2024), which posits that language models acquire the majority of their knowledge during pre-training. We have the reason to investigate whether a minimal set of parameters could sufficiently model the transformation between pre-trained and fine-tuned logits. To address this question, we conducted a preliminary experiment utilizing a single linear layer (see Appendix A.1 for details) to model

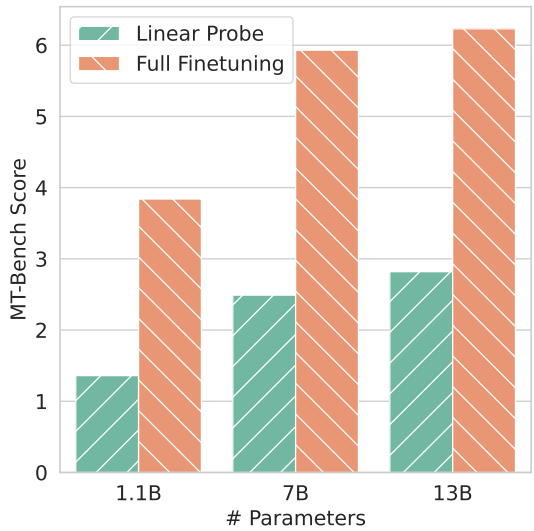


Figure 2: MT-Bench scores of models with different sizes using linear probes for modeling p_Δ and full fine-tuning.

π_θ. We trained this model on ShareGPT (Chiang et al., 2023), a typical instruction-tuning dataset, and evaluated its efficacy.

As shown in Figure 2. Our findings indicate that a single linear probe is insufficient to effectively model such a transformation. Specifically, while the model demonstrated proficiency in emulating the assistant's style, it failed to capture and retrieve the task-specific knowledge required for problem-solving. We further categorized typical errors, as presented in Figure 5 in the Appendix. These results suggest that the relationship between pre-trained and fine-tuned logits is more complex than initially hypothesized, potentially involving deeper neutral network for more complex transformations or requiring a more sophisticated architectural modification.

Therefore, we further explored whether utilizing deeper networks could contribute to better modeling of this differential. Specifically, we employed a randomly initialized 7B Llama architecture as π_Δ, subjected to the same training procedure to guide the alignment of 1.1B pre-trained Llama Model. We observe that while it marginally outperforms the shallow linear probe in instruction-following capability, it still fell significantly short of full fine-tuning performance. Consequently, we believe that employing pre-trained weights to predict the delta logits is necessary.

4 Methods

4.1 Connection Schemes

Based on the formulation in Section 3.1, we propose two distinct architectures to transform a pre-trained model into a value network and integrate it with a pre-trained model, namely **Cascade** and **Residual**, as illustrated in Figure 3 (a). It is important to note that both approaches maintain the internal structure of the LLM unchanged, focusing solely on the connection schemes between the pre-trained model and the value network.

Cascade. In the cascade architecture, the value model receives both the logits information from the pre-trained model and text embedding as the inputs. The fusion of these inputs is formulated as:

$$\boldsymbol{h}_{\text{final}} = p_{\text{base}} \boldsymbol{W}_e \oplus \boldsymbol{h}_{\text{embed}}$$

where p_{base} represents the pre-trained probabilities, $\boldsymbol{W}_e$ is the pre-trained embedding matrix, and $\boldsymbol{h}_{\text{embed}}$ denotes the original text embedding, $\oplus$ is the fusing operator (set as addition in this work).

Alternatively, one may posit that the input derived from sampled text is not critical, and that the value model can effectively predict $\boldsymbol{z}_\Delta$ solely based on the logits from the pre-trained model. In this case:

$$\boldsymbol{h}_{\text{final}} = p_{\text{base}} \boldsymbol{W}_e$$

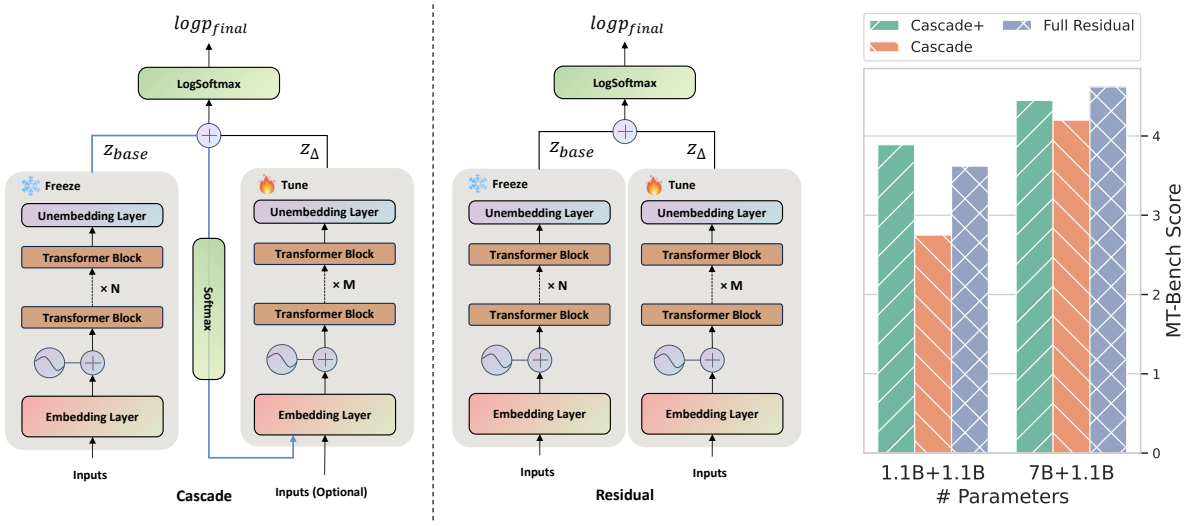
The cascade model takes $\boldsymbol{h}_{\text{final}}$ as input and outputs the value scores:

$$\boldsymbol{z}_\Delta(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) = \pi_\Delta(\boldsymbol{h}_{\text{final}}; \theta_{\text{cascade}}).$$

Full Residual. The residual architecture specifies that the value model has no access to the logits information of the pre-trained model. Instead, it predicts the residual for the next token position based solely on the previous tokens. This design allows the value model to begin forwards without waiting for the pre-trained model's inference to complete. However, this approach may potentially limit transferability, as it cannot steer predictions based on logits from the inference-time pre-trained model which it is plugged into, which may differ significantly from the training-time one.

The full residual model takes previous tokens $\boldsymbol{x}$ as input and outputs the value scores:

$$\boldsymbol{z}_\Delta(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) = \pi_\Delta(\boldsymbol{x}; \theta_{\text{residual}}).$$



(a) (b)

Figure 3: Cascade and Residual connection schemes for Inverse Value Learning. (a) Illustration of the two connection schemes. (b) MT-Bench scores for 1.1B models trained on the 1.1B base and generalized to the 7B base model using different connection schemes. "Cascade+" refers to the cascade value model where logits information from the pre-trained models is fused with the original token embeddings (marked as optional in Figure (a)) and used as input for the value model.

4.2 Plug-and-play transferability

Once a value model is trained on one backbone, it can be plugged into any other pre-trained models without further training, potentially achieving similar performance and behaviors as if it were post-trained. This capability stems from the feature of logits space steering (Mitchell et al., 2023). To further expand the boundaries of transferability, we investigate techniques to mitigate overfitting to the training-time backbone and methods for cross-vocabulary model transfer.

4.2.1 Regularization via Norm Constraint

Inevitably, the model tends to overfit to the backbone it is trained with, thereby reducing its generalization capability on other pre-trained models. Applying appropriate regularization techniques during training can mitigate this issue. For example, we can directly constrain the sparsity of delta logits by incorporating an L1 norm term in the loss function. Specifically, this can be expressed as:

$$\mathcal{L} = \mathrm{CE}(\boldsymbol{z}_{\text{post}}, p_l) + \lambda \|\boldsymbol{z}_\Delta\|_1$$

where λ is a hyperparameter controlling the strength of regularization.

4.2.2 Vocabulary mapping

To adpat value-guided decoding to pre-trained and value models trained on different vocabularies and tokenizers, such as plugging a Llama-2-based value model to a Llama-3 family pre-trained model, a vocabulary mapping algorithm is essential. We adopt the `MinED` mapping algorithm (Wan et al., 2024) for per step mapping of the logits, which identifies mapping relationships between two sequences by minimizing their edit distance. Based on these mapping relationships, we can derive a vocabulary mapping matrix, which is then normalized and regularized to ensure the mapped logits values are not biased. During each decoding step, this mapping matrix is employed to transform the vocabulary space as stated in equation 3. The details of the algorithm and corresponding evaluation is shown in Appendix B.

5 Experiments

5.1 Training Datasets

The experiments are primary conducted based on the following two training datasets:

ShareGPT is a dialog dataset collected from `sharegpt.com`, a website collecting user-shared dialog with ChatGPT. This dataset encompasses a wide range of topics, making it suitable as a general instruction tuning set focusing on the instruction following ability of models. While ShareGPT is the training set of Vicuna (Chiang et al., 2023), the

exact data has not been released. For our study, we utilize a version comprising approximately 86K dialogs.

InfinityInstruct-7M (Zhang et al., 2024a) is a large scale instruction tuning dataset specifically designed to enhance foundational reasoning abilities in code and math. This collection consists of 7M carefully curated instructions, aggregated and filtered from a variety of open-source datasets.

5.2 Pre-trained Backbones

We employed TinyLlama (Zhang et al., 2024b) and the Llama-2 series (Touvron et al., 2023) as backbones for the 1.1B, 7B, 13B, and 70B parameter scales. To validate broader transferability, we also conduct tests on CodeLlama (Roziere et al., 2023), a model based on Llama-2 that undergoes additional continual pertaining on 20B of code related data. Furthermore, we use Llama-3 (Dubey et al., 2024) to test cross model family generalization ability.

The training details of the value models are shown at Appendix C.

5.3 Tasks

General Instruction Following. We employ MT-Bench (Zheng et al., 2023) as an open-ended instruction following and use GPT-4 to judge the response from the models.

Zero-shot capabilities. We select representative datasets in TÜLU-2 (Ivison et al., 2023) evaluation suite, including IFEval (Zhou et al., 2023b), ToxiGen (Hartvigsen et al., 2022), GSM8K (Cobbe et al., 2021), MBPP (Austin et al., 2021), HumanEval (Chen et al., 2021). For MBPP and HumanEval, we report the Pass@1 scores. We apply corresponding chat templates for the instruction-following models.

Few-shot capabilities. We select representative few-shot datasets originally employed in Open LLM Leaderboard v1[1] for testing the few shot capabilities, including ARC (25-shot) (Clark et al., 2018), TruthfulQA (0-shot) (Lin et al., 2021), MMLU (5-shot) (Hendrycks et al., 2020), Winograde (5-shot) (Sakaguchi et al., 2019), Hellaswag (10-shot) (Zellers et al., 2019), .

[1] https://huggingface.co/docs/leaderboards/en/open_llm_leaderboard/archive

5.4 Results

Which connection scheme enables better transfer for Inverse Value Learning? Prior to conducting extensive transfer testing, we first evaluate which of the proposed schemes is more conducive to modeling z_Δ. Using a 1.1B model fine-tuned as the value model, we study the effects of the two structures proposed in Figure 2(a) on both the 1.1B backbone (used during training) and the 7B backbone (used for inference-time transfer). As shown in Figure 2(b), changes in the model's connection scheme did not result in significant differences in general instruction-following ability, especially when transferring to other pre-trained models. Integrating logits from the base model slightly improves performance on the training backbone. However, when transfering to a larger backbone, there is a decrease in performance compared to the full residual structure. We attribute this to two potential factors: 1) Excessive input signals introduce noise to the cascade scheme, and 2) The value model in the cascade scheme no longer maintains a purely autoregressive form (i.e., it predicts the p_Δ from based on the pre-trained logits at the same time step), widening the gap between pre-training and fine-tuning. Consequently, we adapt the full residual scheme for subsequent experiments in Table 1, given its superior performance and efficiency.

Wide-ranging transferability. The inverse learned value model demonstrated board generalizability across models of different parameter scales, consistent with the findings of Mitchell et al. (2023). When a 1.1B parameter value model, trained on a 1.1B parameter backbone, is transferred to a 7B parameter base model, it retains approximately 80% of its instruction-following capability compared to full finetuning. Additionally, when a 7B parameter value model, jointly trained with a 7B base model, was transferred to a 13B parameter model, the instruction-following performance matched that of full finetuning. Same transferability can be scaled to 70B.

Table 1 further illustrates the potential of our approach in continued pre-training scenarios within the same model family. For instance, in the case of Codellama, the 1.1B value model can be directly applied to guide the base model in acquiring instruction-following capabilities while preserving its strong performance in coding tasks.

Methods	IF	Zero-Shot					Few-Shot				
	MT-Bench	IFEval	GSM8K	Toxicgen↓	MBPP	HumanEval	ARC	Hellaswag	MMLU	TruthfulQA	Winogrande
Tinyllama-1.1B											
Full	**3.84**	**16.1**	1.97	26.8	14.1	5.37	27.6	40	**26.4**	**34**	57.8
1.1B	3.62	15.2	2.43	**21.1**	10.6	8.17	28.4	44.1	25.2	30.4	57.5
1.1B norm	3.52	14.6	**2.65**	62.8	**17.9**	**15.4**	**31.4**	**44.9**	25.6	30	**59.6**
Llama-2-7B											
Full	5.93	23.8	10.6	49.8	26	17.6	**51.9**	59.5	**50.1**	**41.5**	**74.3**
1.1B	4.62	15.5	4.7	**44.3**	15.8	11.3	42.7	55	43.2	31.1	66.2
1.1B norm	4.55	15.3	5.61	62.8	20.4	15.4	48.5	57.9	44.2	31.8	72.5
1.1B infinity	4.06	17	**27.2**	65.3	**30.9**	**18.3**	48.3	58.6	45.8	30.2	71.2
7B	5.96	24.4	13.4	58.3	21.4	13.9	51	**61**	49.3	40.1	72
7B norm	**6.03**	**32.7**	13.5	45.2	23.4	14.8	**51.9**	60.9	49.6	39.9	70.6
Llama-2-13B											
Full	6.23	36.3	18.7	**5.03**	24.3	22.8	49.2	60.6	**53.9**	**41.8**	72.3
1.1B	4.74	15.5	6.67	39.7	16.4	13.6	46.1	57.2	52	29.3	70.1
1.1B norm	4.65	14.8	10.6	55.8	28.5	15.9	49.1	59.1	52.9	31.8	76
1.1B infinity	4.45	18.7	**30.3**	77	**33.1**	**23.7**	52.2	61.2	53.7	30.3	74.6
7B	**6.34**	**26.3**	18.3	56.8	24.9	19.6	56.1	63.6	53.4	37.9	74.4
7B norm	6.22	25.3	17.5	39.1	27.7	15.9	**56.6**	**63.6**	53.6	37.7	**74.8**
Llama-2-70B											
Full	**7.08**	**46.6**	35.6	**0**	36.6	**41.9**	**62.9**	67.8	**69.3**	**48.4**	80.5
1.1B	4.92	16.6	12.4	42.1	27.9	20.4	52.6	60.5	68.4	31.6	77.3
1.1B norm	5.37	16.8	24.9	70	37.8	28.6	62.8	66.7	67.9	37.1	**82.9**
1.1B infinity	4.86	20.1	**44**	71	**42.9**	33	58.6	66.4	68.5	36.1	81
7B	6.56	33.3	27.9	43.6	29.4	25.8	62.1	68.2	63.6	42.1	78.6
7B norm	6.48	34.6	28.8	27.5	33.1	25.1	62.8	**68.2**	64.1	41.9	79.3
Codellama-7B											
Full	5.47	**33.4**	15.4	83.2	**39.4**	**43.6**	40.6	48.3	41.6	34.4	64.5
1.1B	3.94	15.7	3.79	48.2	28.1	24	35.6	45.9	35.2	29.1	59.6
1.1B norm	3.99	15.9	4.85	62.3	**39.4**	31	39.4	46.9	38.1	29.5	64.4
1.1B infinity	3.51	19.8	**27.2**	65.2	31	18.4	38.9	49.1	38.5	29.6	64.7
7B	**5.64**	31.2	12	56.8	30.9	26.8	47.9	53.9	49.2	38.2	**68.4**
7B norm	5.54	31.2	16.8	**40.6**	33.2	25.1	**47.8**	**54.1**	**49.7**	**37.4**	68.3

Table 1: Performance comparison of full fine-tuning and inverse value learning on instruction following, zero-shot and few-shot tasks. Note that value models of various parameter scales are trained together with base models of corresponding scales. For instance, "1.1B" indicates a 1.1B value model trained on a 1.1B backbone model. We further use a pink cell tag to denote a transfer setting. "↓" symbol indicates that lower values are better.

The gap between inverse value learning and full fine-tuning narrows with the introduction of more direct supervision and larger value models. Encouragingly, when we evaluate the inverse value learning paradigm in the context of code and math post-training, which requires training on several million instruction-response pairs, we observe corresponding performance gains. Notably, these gains are transferable, even though the 1.1B backbone models perform poorly on code and math tasks. For instance, 1.1B value models, trained with 1.1B base models on `InfinityInstruct`, substantially improve the performance of 13B models on benchmarks such as GSM8K, MBPP, and HumanEval, outperforming other value models and full finetuning methods on the `ShareGPT` dataset. Furthermore, the overall performance of inverse value learning generally increases with the parameter scale of the value models, demonstrating the scalability of this paradigm.

Additional norm constraint can effectively prevent overfitting to the weak base models. Table 1 also reveals that the value models trained with the norm term, when generalizing to stronger models, typically maintain more robust reasoning and knowledge capability while sacrificing a small degree of instruction-following capability. This phenomenon is particularly pronounced when there is a significant disparity in parameter scale between the value model and the pre-trained model. Conse-

quently, the incorporation of a simple normalization term during the training of small value models can mitigate overfit ting to weak base model and enhance weak-to-strong generalization performance.

6 Discussions

Llama-2-70B +	1.1B	1.1B norm	1.1B curriculum
MT-Bench	4.92	5.37 (+0.45)	5.14 (+0.22)
IFEval	16.6	16.8 (+0.2)	15.7 (-0.9)
GSM8K	12.4	24.9 (+12.5)	15.8 (+3.4)
Toxicgen ↓	42.1	70 (-27.9)	40.9 (+1.2)
MBPP	27.9	37.8 (+9.9)	31.7 (+3.8)
HumanEval	20.4	28.6 (+8.2)	25.5 (+5.1)
ARC	52.6	62.8 (+10.2)	57.7 (+5.1)
Hellaswag	60.5	66.7 (+6.2)	64.3 (+3.8)
MMLU	68.4	67.9 (-0.5)	68.4 (+0.0)
TruthfulQA	31.6	37.1 (+5.5)	36.2 (+4.6)
Winogrande	77.3	82.9 (+5.6)	73.2 (-4.1)

Table 2: Comparison between Llama-2-70B based model guided by the standard 1.1B value model, the 1.1B model trained with a curriculum across 1.1B, 7B, and 13B models, and the model trained with a L1 normalization term.

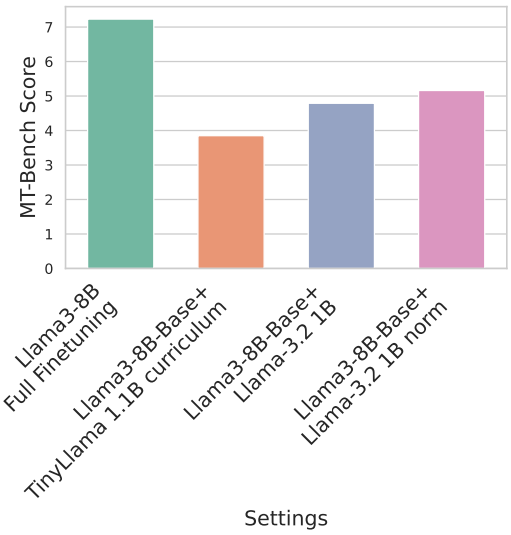


Figure 4: MT-Bench scores of Llama-3-8B fully fine-tuned, Llama-3-8B-base plugged with the TinyLlama 1.1B value model (cross vocabulary), and Llama-3-8B-base plugged with the Llama 3.1 value models (same vocabulary).

Can a multi-model curriculumly finetuning improves weak-to-strong generation? Given that all models in the Llama-2 series share the same vocabulary, we designed a curriculum to train the model sequentially on the 1.1B, 7B, and 13B versions, and then evaluated its generalization performance on the powerful 70B model. We maintained consistent training data volume and iteration counts across experiments. As shown in Table 2, our findings indicate that this multi-model curriculum training can also mitigate over-fitting to some extent, functioning similarly to regularization techniques. However, this method necessitates longer training periods and more complex model scheduling processes. Consequently, it does not offer significant advantages in weak-to-strong generation over simpler normalization methods in our observation.

Feasibility of cross-vocabuary transfer. We explored the transferability of model capabilities across different model families. We transferrs 1.1B parameter value models to Llama-3 using vocabulary mapping at decoding time, comparing it with Llama-3.2-1B guided and full fine-tuning. Interestingly, among all settings of the trained value model, only the curriculum-trained value model yielded acceptable generation ability in the cross-model and cross-vocabulary setting. However, as Figure 4 shows, while the transferred model grasped basic conversational abilities, it still underperformed compared to same-vocabulary value models and full fine-tuning. In addition to the generation ability of the value model, we attribute this performance gap partly to tokenization differences: source tokens can be mapped to multiple consecutive target tokens, which are unfamiliar to the source model in the following decoding steps, potentially degrading performance during decoding. While we've demonstrated the feasibility of cross-vocabulary value transfer, improving vocabulary mapping algorithms and cross-family model guidance remain promising areas for future research.

7 Conclusions

In this paper, we introduced a novel framework for transferable post-training via inverse value learning, which models residual logits adjustments using a separate value network. We systematically explored various aspects of this paradigm, including network architectures, pre-training weights, and connection schemes. Additionally, we addressed challenges related to overfitting and cross-vocabulary transfer by incorporating regularization techniques and a vocabulary mapping algorithm. Our results confirm that the value network exhibits broad transferability across models of different sizes and families, highlighting the potential of logits-space adjustments for efficient model adaptation.

Limitations

We've discussed the limitation of additional computational resources in Section 3.1. In addition, inverse value learning is not applicable when logits of models are unavailable, and there is a theoretical performance degradation when intergrate value models to pre-trained models with different vocabularies.

References

Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. 2024. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeffrey Wu. 2024. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. In *Forty-first International Conference on Machine Learning*.

Boxi Cao, Keming Lu, Xinyu Lu, Jiawei Chen, Mengjie Ren, Hao Xiang, Peilin Liu, Yaojie Lu, Ben He, Xianpei Han, Le Sun, Hongyu Lin, and Bowen Yu. 2024. Towards scalable automated alignment of llms: A survey. *arXiv preprint arXiv:2406.01252*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

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. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3309–3326, Dublin, Ireland. Association for Computational Linguistics.

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

GE Hinton. 1999. Products of experts. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99.(Conf. Publ. No. 470)*, volume 1, pages 1–6. IET.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. 2024. ARGS: Alignment as reward-guided search. In *The Twelfth International Conference on Learning Representations*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. 2024. The unlocking spell on base LLMs: Rethinking alignment via in-context learning. In *The Twelfth International Conference on Learning Representations*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. 2024. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*.

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D Manning. 2023. An emulator for fine-tuning large language models using small language models. *arXiv preprint arXiv:2310.12962*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: your language model is secretly a reward model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

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

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. In *The Twelfth International Conference on Learning Representations*.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024. Reft: Representation finetuning for language models. *arXiv preprint arXiv:2404.03592*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Bo-Wen Zhang, Yan Yan, Lin Li, and Guang Liu. 2024a. Infinitymath: A scalable instruction tuning dataset in programmatic mathematical reasoning. *arXiv preprint arXiv:2408.07089*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024b. Tinyllama: An open-source small language model. *Preprint*, arXiv:2401.02385.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023b. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A The details of the Linear Probes

A.1 Probe Architecture

In this work, we adapt the MLP layer from the LLaMA architecture as a linear probe, which can be formally expressed as:

$$\mathbf{y} = \mathbf{W}_{\text{gate}}(\sigma(\mathbf{W}_{\text{gate}}\mathbf{x}+\mathbf{b}_{\text{gate}})\odot(\mathbf{W}_{\text{up}}\mathbf{x}+\mathbf{b}_{\text{up}}))+\mathbf{b}_{\text{gate}}$$

where $\sigma(\cdot)$ is the activation function, and $\odot$ denotes the element-wise product.

A.2 Failure mode

We demonstrated two representative failure case of using Linear layer to model the $z_{\triangle}$ in Figure 5.

B Vocab mapping algorithm

We adapt the `MinED` algorithm for vocab mapping. The detailed algorithm is shown at Algorithm 1. We conducted a preliminary evaluation of the alignment matrix by computing the overlap ratio between the target sequences mapped using this matrix and the golden target sequences on the training set, achieving an average overlap ratio of 0.42.

C Training Details

The majority of the training is performed on an 8 × A100-80G node. We adapt the hyperparameter configuration as outlined in Table 3.

Algorithm 1 Vocab Alignment

Require: Base tokenizer T_b with vocabulary V_b
Value tokenizer T_v with vocabulary V_v
Training data $\mathcal{D} = \{\text{text}_1, \text{text}_2, \ldots, \text{text}_N\}$
Ensure: Normalized token alignment matrix $M \in \mathbb{R}^{|V_b| \times |V_v|}$

Initialize count matrix $C \leftarrow \mathbf{0}_{|V_b| \times |V_v|}$
for each text $\in \mathcal{D}$ **do**
  Tokenize text using T_b to obtain sequence $\mathbf{t}_b$
  Tokenize text using T_v to obtain sequence $\mathbf{t}_v$
  Remove special prefixes from tokens in $\mathbf{t}_b$ and $\mathbf{t}_v$
  Compute cost matrix based on edit distances between tokens in $\mathbf{t}_b$ and $\mathbf{t}_v$
  Perform Dynamic Time Warping (DTW) to find the optimal alignment path $\mathcal{P}$
  for each aligned token pair $(t_{b,i}, t_{v,j}) \in \mathcal{P}$ **do**
    Increment count: $C[t_{b,i}, t_{v,j}] \leftarrow C[t_{b,i}, t_{v,j}] + 1$
  end for
end for
Normalize and sparsify the count matrix C to obtain the mapping matrix M
return M

Representative failure modes for linear probes

Instruct: Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions.
Response: Sure, here are some more travel blog post ideas:
1. A review of a recent trip to Hawaii
2. A trip to the Canary Islands, Spain
3. A trip to Egypt, where you can visit the pyramids and the Sphinx
4. A trip to Costa Rica, where you can hike in the rainforests and see the wildlife
5. A trip to South Africa, where you can visit the wildlife parks and take a safari

Instruct: How many integers are in the solution of the inequality $|x + 5| < 10$
Response: The solution is 10.

Figure 5: Failure modes for linear probes.

Hyperparameters	Value
optimizer	AdamW
learning rate (1.1B / 7B)	1e-4 / 1e-5
warm up ratio	0.04
lr schedular type	cosine
# train epochs	3
global batch size (1.1B / 7B)	32 / 16
max length	4096
λ	1.0

Table 3: The training configuration for 1.1B and 7B value models.