

# Capability Saliency Vector: Fine-grained Alignment of Loss and Capabilities for Downstream Task Scaling Law

Qiming Ge<sup>♡★\*</sup>, Shuhao Xing<sup>♡★\*</sup>, Songyang Gao<sup>♡</sup>, Yunhua Zhou<sup>♡</sup>,  
Yicheng Zou<sup>♡†</sup>, Songyang Zhang<sup>♡</sup>, Zhi Chen<sup>♡</sup>, Hang Yan<sup>♡</sup>, Qi Zhang<sup>★</sup>,  
Qipeng Guo<sup>♡</sup>, Kai Chen<sup>♡</sup>

<sup>♡</sup>Shanghai AI Laboratory

<sup>★</sup>College of Computer Science and Artificial Intelligence, Fudan University  
qmge22@m.fudan.edu.cn, zouyicheng@pjlab.org.cn

## Abstract

Scaling law builds the relationship between training computation and validation loss, enabling researchers to effectively predict the loss trending of models across different levels of computation. However, a gap still remains between validation loss and the model’s downstream capabilities, making it untrivial to apply scaling law to direct performance prediction for downstream tasks. The loss typically represents a cumulative penalty for predicted tokens, which are implicitly considered to have equal importance. Nevertheless, our studies have shown evidence that when considering different training data distributions, we cannot directly model the relationship between downstream capability and computation or token loss. To bridge the gap between validation loss and downstream task capabilities, in this work, we introduce **Capability Saliency Vector**, which decomposes the overall loss and assigns different importance weights to tokens to assess a specific meta-capability, aligning the validation loss with downstream task performance in terms of the model’s capabilities. Experiments on various popular benchmarks demonstrate that our proposed Capability Saliency Vector could significantly improve the predictability of language model performance on downstream tasks.

## 1 Introduction

Large language models have demonstrated impressive performance across a wide range of tasks, but this achievement comes with the trade-off of significant computational demands. To mitigate the computational burden during the training of large models, researchers have developed the scaling law (Bahri et al., 2024; Hoffmann et al., 2022; Kaplan et al., 2020; Muennighoff et al., 2024), a framework that predicts how variations in model size and

data scale impact the validation loss of LLMs. This allows researchers to use smaller models to predict the validation loss of larger ones, reducing the cost of hyperparameter tuning and trial-and-error experimentation. However, in practical applications, a model’s capabilities are typically assessed using a variety of downstream benchmarks, and validation loss does not always correlate directly with performance on these tasks. It is not uncommon to encounter instances where models with similar validation losses exhibit significantly different downstream task performance (Liu et al., 2023). Consequently, validation loss alone is not always a reliable indicator of a model’s true capabilities.

To address this issue, some researchers have chosen to bypass validation loss, focusing instead on directly establishing the scaling relationship between computational resources and task performance (Gadre et al., 2024). To minimize the effects of data distribution shifts, they adjusted the amount of computation and model sizes while keeping the pre-training dataset fixed across all experimental settings. However, we argue that the impact of data distribution shifts cannot be overlooked, as they may substantially influence model performance on downstream tasks. As demonstrated by our experiments (Section 3.1), the same amount of computation can yield markedly different outcomes when models are trained on distinct data distributions.

Another line of research focuses on developing alternative metrics that better correlate with downstream task performance. For instance, Ruan et al. (2024) proposed the Observational Scaling Law (OSL), leveraging open-source benchmark results and applying principal component analysis (PCA) to explore various dimensions of model capabilities. While these approaches encompass a broad range of model families that are pre-trained on different data distributions, they depend heavily on the diversity and scope of the benchmarks used, which can incur substantial evaluation costs.

<sup>\*</sup>Equal contribution. This work was done when Qiming Ge was an intern at Shanghai AI Laboratory.

<sup>†</sup>Corresponding Author

From the discussions above, we identify two important but underexplored questions that this work seeks to address:

- Is it reasonable to predict downstream capabilities directly based on computational cost?
- Why is it challenging to establish a clear relationship between validation loss and a model’s capabilities?

To address the first question, we explored the impact of different pre-training data distributions on the ability to model downstream tasks. Our experiments show that, under the same computational budget, different pre-training data distributions can lead to varying downstream capabilities. Therefore, it is not reasonable to predict downstream task performance directly based on computational cost. To answer the second question, we revisit the concept of validation loss, a metric used to evaluate a model’s cumulative penalty for next-token predictions. This metric inherently assumes that all tokens contribute equally to the model’s training process. However, previous studies have demonstrated that tokens may differ in their learnability and difficulty (Lin et al., 2024; Xia et al., 2022). Inspired of this, we introduce the concept of **Capability Saliency Vector (CSV)**, which quantifies the capabilities required for downstream tasks by assigning different importance weights to tokens. By computing a weighted average of the loss for these tokens, this metric can replace the raw validation loss and demonstrate a stronger correlation with downstream task performance. Furthermore, we propose an optimization algorithm to automatically derive the Capability Saliency Vector for any given text. Experiments conducted on six popular benchmarks show that our approach consistently achieves a high correlation across different model series and significantly enhances the predictability of language model performance on downstream tasks.

Overall, our contributions are threefold:

- Our experiments indicate that downstream scaling laws cannot be directly modeled using computation or the average token loss on the validation set.
- We introduce the Capability Saliency Vector and present an algorithm to automatically derive it. This allows us to quantify a model’s capabilities by assigning different importance

weights to the loss of specific tokens, thereby establishing a strong correlation between validation loss and downstream task performance.

- Through experiments on six benchmarks for different model capabilities, we show that our method consistently achieves high correlation across various model series and significantly improves the predictability of model performance on these benchmarks. The predicted mean squared error was maintained within the range of  $1e-3$ .

## 2 Related work

### 2.1 Scaling Law for Downstream Tasks

Scaling law establishes the power-law relationships between computation  $C$ , model parameters  $N$ , the number of training tokens  $D$ , and cross-entropy loss on the validation set. However, there is still a gap between validation loss and downstream task performance. Recent studies have tried to investigate the scaling laws governing downstream task performance. Owen (2024) investigated the predictability of model scaling on BBH (Srivastava et al., 2022) and MMLU (Hendrycks et al., 2020) tasks, proposing that downstream task performance follows a sigmoidal function and can be predicted. Their findings also indicate that prediction errors decrease as more models are incorporated into the fitting process. Ruan et al. (2024) introduced an alternative observational approach, applying low-rank decomposition to publicly available model evaluation results to extract sub-dimensions representing the model capabilities. These sub-dimensions were then used to establish scaling laws that relate computational cost to downstream task performance. Gadre et al. (2024) examined the scaling laws of overtraining models on specific pre-training data distributions. They identified a power-law relationship between validation loss and downstream task performance, demonstrating predictability after pre-training on a given data distribution.

However, the aforementioned studies primarily focused on the relationship between computational cost and downstream task performance, without addressing the fact that different pre-training data distributions can result in different downstream performances for the same computational cost. Isik et al. (2024) explored the impact of pre-training data distribution on downstream performance in translation tasks. Their findings revealed that different

pre-training distributions significantly influence the scaling behavior of downstream tasks, indicating that data distribution should be considered when evaluating model scaling laws.

## 2.2 Efficient Benchmark Prediction

As large language models continue to advance, numerous benchmarks have been developed to assess their diverse capabilities. To comprehensively evaluate the performance of a pre-trained model, it is necessary to measure its scores across various downstream tasks. However, conducting a full downstream evaluation is often time-consuming, prompting recent efforts to improve its efficiency. For example, Ye et al. (2023) and Perlitz et al. (2023) have explored reducing the number of task examples in benchmarks like BBH (Srivastava et al., 2022) and HELM (Liang et al., 2022) to streamline the evaluation process.

Recent studies, such as Polo et al. (2024) and Kipnis et al. (2024), have explored leveraging Item Response Theory (Lord and Novick, 2008; Baker, 2001) to extract model features from open-source evaluations and compress validation sets. This allows for the reconstruction of benchmark scores using these compressed subsets. Similarly, Pacchiardi et al. (2024) applied a feature extraction approach and trained an evaluator to directly predict performance on individual samples. Zhang et al. (2024) demonstrated that incorporating task performance across different model families, alongside additional model and task information, can enhance prediction accuracy. However, these methods still rely on accuracy metrics for part of the data, and it remains uncertain whether downstream task performance (e.g., accuracy) can be directly predicted based on downstream scaling laws.

## 3 Method

In this section, we first explore whether downstream task performance can be directly predicted using computation or average token loss. Then, we introduce **Capability Salience Vector (CSV)**, which automatically assign different importance weights to validation token loss. This aligns the loss of tokens with the specific capabilities required for downstream tasks.

### 3.1 Impact of Pre-training Data Distribution on Modeling Downstream Scaling Law

Previous work on downstream scaling laws has primarily explored whether the performance of large

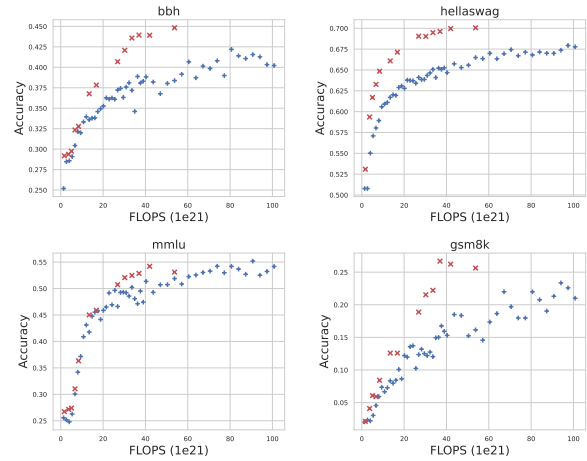


Figure 1: Relationship between computation (FLOPs) and performance (accuracy) across various downstream tasks is depicted, with red and blue points indicating model checkpoints trained on two different data distributions. Under the same computational budget, models trained on different data distributions exhibit varying downstream task performance.

models on downstream tasks can be predicted using pretrained small models. These studies follow a common setting, where small models trained on the same pretraining data distribution are used to predict the performance of larger models. However, during the pretraining process, researchers may need to continuously adjust the data composition to compare the impact of different data distributions on model performance. In this framework, applying downstream scaling laws would require training multiple small models under different configurations with the same data distributions, making the experiments significantly more costly.

In this section, we first investigate whether downstream scaling laws can be directly modeled using computational cost. Specifically, we use InternLM2.5 as the base architecture and train two series of models on significantly different pretraining data distributions which is the same pretraining data sources as InternLM2.5. These sources include a diverse mix of web data, code data, math data, and other domain-specific datasets. We only adjusted the proportions of different subsets within this pretraining corpus. We then select checkpoints at different levels of computational cost to evaluate downstream performance.

Figure 1 illustrates the performance of two models with different training data distributions on various downstream tasks under different computational budgets. We observe that for tasks such as

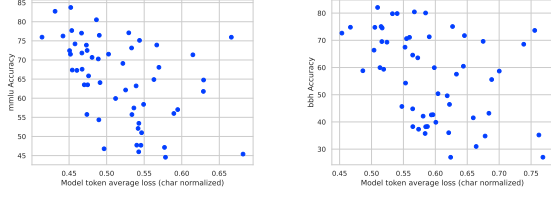


Figure 2: The relationship between average token loss and downstream task performance.

Hellaswag and BBH, models trained with the same computational cost exhibit differing downstream task performance. This indicates that a direct function from computation to downstream task performance cannot be reliably modeled. Therefore, when predicting downstream task performance, we must account for differences in the models’ training data distributions. Moreover, language models exhibit emergent capabilities in certain tasks, making it difficult to predict their performance using smaller models trained on the same pretraining data distribution. Before reaching the emergence threshold—when computational resources are insufficient and loss remains relatively high—the performance of small models on downstream tasks becomes difficult to measure (Du et al., 2024).

This raises an important question: Is there a suitable metric that satisfies the following conditions?

- The metric provides continuous measurements and is independent of emergent capabilities.
- The metric reflects differences in data distribution and can be predicted during pretraining.

Inspired by traditional scaling laws, we consider validation loss as a potential candidate. Validation loss is continuous and serves as an indicator of model capability. Therefore, we aim to model the relationship between validation loss and downstream performance. To explore this, we measure the downstream performance of different open-source models and their average token loss on task validation datasets. To avoid biases introduced by different tokenizers across models, we normalize the loss calculation at the character level.

Figure 2 illustrates the relationship between validation loss and downstream performance in MMLU and BBH. We observe that models can exhibit varying downstream performance across different tasks with the same validation loss. We revisit the computation of validation loss. Traditional

scaling laws calculate validation loss by averaging it at the dataset level, treating all loss values with equal importance. However, different loss values may contribute unequally to measuring a model’s task specific capability. This indicates that a more fine-grained approach may be necessary for modeling loss effectively. In the next section, we introduce the concept of the **Capability Salience Vector** (CSV). This method assigns different importance weights to token-level loss values in the validation set, aiming to establish a more predictable relationship between validation loss and downstream performance.

### 3.2 Modeling Downstream Scaling Law

Inspired by the work of Arora and Goyal (2023), we consider that predicting different token fragments requires different meta-capabilities. These meta-capabilities ultimately determine the performance of downstream tasks. Traditional scaling laws treat all tokens equally when modeling validation loss. While the expectation of validation loss provides a rough estimate of model capability, it does not accurately capture the relationship between loss and downstream performance. To solve this problem, we propose the Capability Salience Vector. By applying a simple linear weighting to loss calculation, we can better model the relationship between loss and downstream performance. Given a validation set  $S = \{X_s\}_{s=1}^{|S|}$ , we first compute the validation loss of each sample  $X_s = (x_1, x_2, \dots, x_{|X_s|})$ . Next, we use the **Capability Salience Vector**, denoted as  $W = \{w_{s,i}\}$ , to apply linear weighting to each token’s loss. This allows us to compute the capability score  $C_m$ , which represents the model’s ability based on the validation text.

$$C_m = -\frac{1}{N_c} \sum_{s=1}^{|S|} \sum_{i=1}^{|X_s|} w_{s,i} \log p(x_i | x_{<i}). \quad (1)$$

where  $N_c$  is the number of validation text characters. Then we use the sigmoidal function to model the functional relationship between capability score  $C_m$  and the model’s downstream task performance (e.g., accuracy) :

$$A_{t,m} = \gamma + \frac{1 - \gamma}{1 + \exp(-\alpha(C_m - \beta))}. \quad (2)$$

where  $\alpha$  and  $\beta$  is the parameters to be fitted,  $\gamma$  represents the expected performance of the task under random guessing.



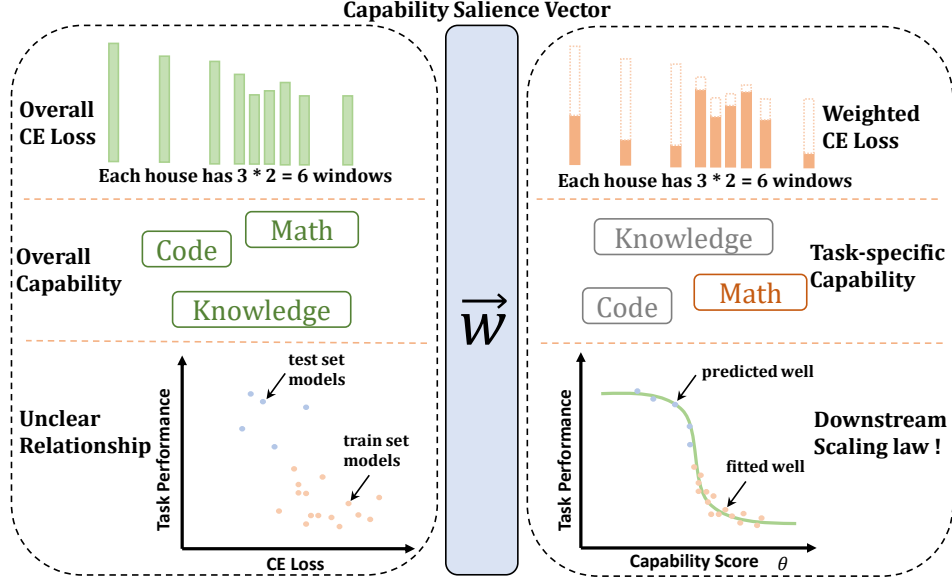


Figure 3: Capability saliency vector establishes the relationship between cross-entropy loss and downstream task performance. It quantifies a model’s capabilities as a weighted sum of token losses, with each token contributing differently to the overall performance.

### 3.3 Modeling Capability Saliency Vector

Since the weights of the Capability Saliency Vector and the parameters  $\alpha, \beta$  are unknown, we use three steps to get the weights of capability Saliency vector.

#### Extract Weights of Capability Saliency Vector

For a given validation set, we first apply a loss mapping algorithm to align losses across different vocabulary spaces. Specifically, we average the token-level losses over their corresponding characters to obtain character-level losses, and then aggregate these character-level losses according to the tokenization result of the target tokenizer. Then we use language model with a scoring head to obtain capability Saliency Vector weights  $W = \{w_{s,i}\}$  of each token:

$$w_{s,i} = f_{\theta}(x_{s,i} | x_{s,<i}). \quad (3)$$

where  $\theta$  is the parameters of scoring head. The scoring head is implemented as a linear layer applied to the final hidden states of a the language model and we froze the other parameters of language model during optimization. This step can be seen as assigning a score to each token, reflecting its contribution to the overall capability representation.

**Fitting Downstream Scaling Law Function** After getting capability Saliency Vector, we collect

token losses on the given validation set and downstream task performance of different models to fitting downstream scaling law function. To do this, we fix  $\theta$  and estimate the parameters  $\alpha$  and  $\beta$  of the downstream scaling law function. We minimize the MSE loss between the predicted and observed downstream task performance by using the Levenberg-Marquardt algorithm.

$$\min_{\alpha, \beta} \sum_{\text{Models } m} [A_{t,m} - \hat{A}_{t,m}]^2. \quad (4)$$

**Optimize Capability Saliency Vector** In this stage, we use the downstream scaling law function to optimize Capability Saliency Vector parameters  $\theta$ . This is done by minimizing the MSE loss between predicted and observed downstream task performance using SGD:

$$\min_{\theta} \sum_{\text{Models } m} [A_{t,m} - \hat{A}_{t,m}]^2. \quad (5)$$

We run multiple iterations of the three-steps optimization process to select the parameters that achieve the best downstream performance prediction. These optimal parameters are saved to obtain the **Capability Saliency Vector** and fit downstream scaling law function. Finally, our algorithm workflow is illustrated in Appendix B.

## 4 Experiment

To validate the effectiveness of our approach, we divide the experiment into two parts. First, we evalu-

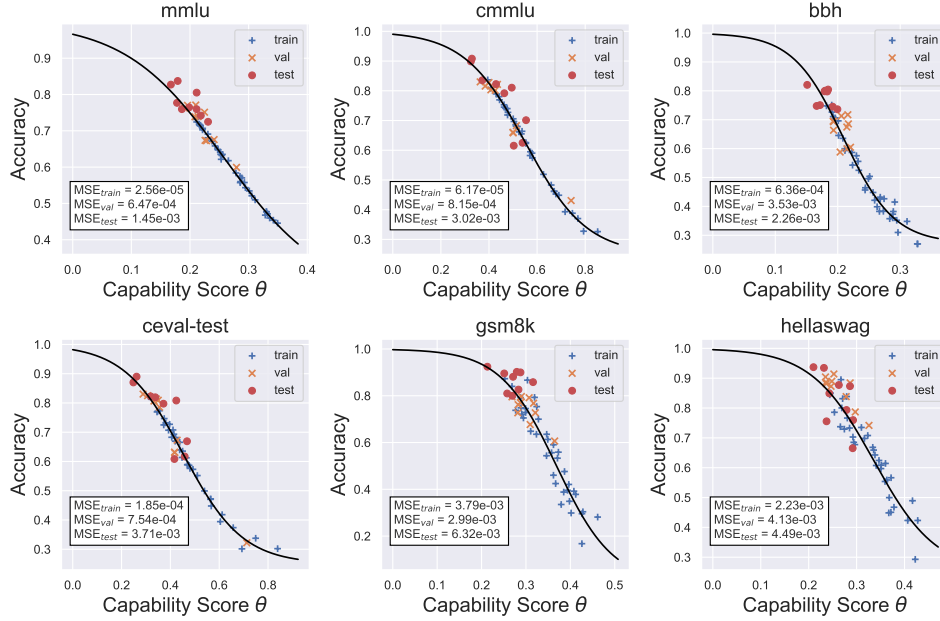


Figure 4: Downstream task prediction results of open-source models using capability score derived from Capability Saliency Vector.

ate the performance of the Capability Saliency Vector using open-source models. Next, we use checkpoints from our own models to assess whether the Capability Saliency Vector, optimized from open-source models, can accurately predict downstream task performance for models trained on different data distributions.

#### 4.1 Settings

**Dataset** To explore the scaling capabilities of models across various downstream tasks requiring different abilities, we conducted our experiments on six popular benchmarks that cover a wide range of skills, including knowledge, reasoning, commonsense, and mathematics. For knowledge capabilities, we choose MMLU (Hendrycks et al., 2020). For reasoning capabilities, we choose BBH (Suzgun et al., 2022). For commonsense capabilities, we choose Hellaswag (Zellers et al., 2019). For mathematics capabilities, we choose Gsm8k (Cobbe et al., 2021). For Chinese language capabilities, we selected the CMMLU (Li et al., 2023) and Ceval (Huang et al., 2024) benchmarks.

**Evaluation Models** Our experiments were conducted under two different evaluation setups:

- **Open-source Model Evaluation:** We validate the effectiveness of our method on over 50 open-source models across different series, including LLAMA2 (Touvron et al., 2023),

LLAMA3 (Dubey et al., 2024), Gemma2 (Team et al., 2024), Qwen1.5 (Bai et al., 2023), Qwen2 (Yang et al., 2024), Yi, Yi1.5 (Young et al., 2024), InternLM2, and InternLM2.5 (Cai et al., 2024). In this setup, we use smaller models from the same series as the training set, while larger models are treated as the validation and test sets. The specific splits are detailed in Appendix A.

- **Closed-source Model Evaluation:** We train two different series of models under varying data distributions and aim to predict their downstream capabilities using the Capability Saliency Vector. The train data source is the same as InternLM2.5. For Capability Saliency Vector optimization, we use all the open-source models along with early checkpoints (the first 50k steps) from both series. One series of our own model is used as the validation set, while the other serves as the test set.

**Evaluation Details** We use the OpenCompass (Contributors, 2023b) tool to evaluate the downstream scores for the aforementioned models. OpenCompass is a python package that supports various large language models and datasets for evaluation and benchmarking. The detailed evaluation results for the open-source models are provided

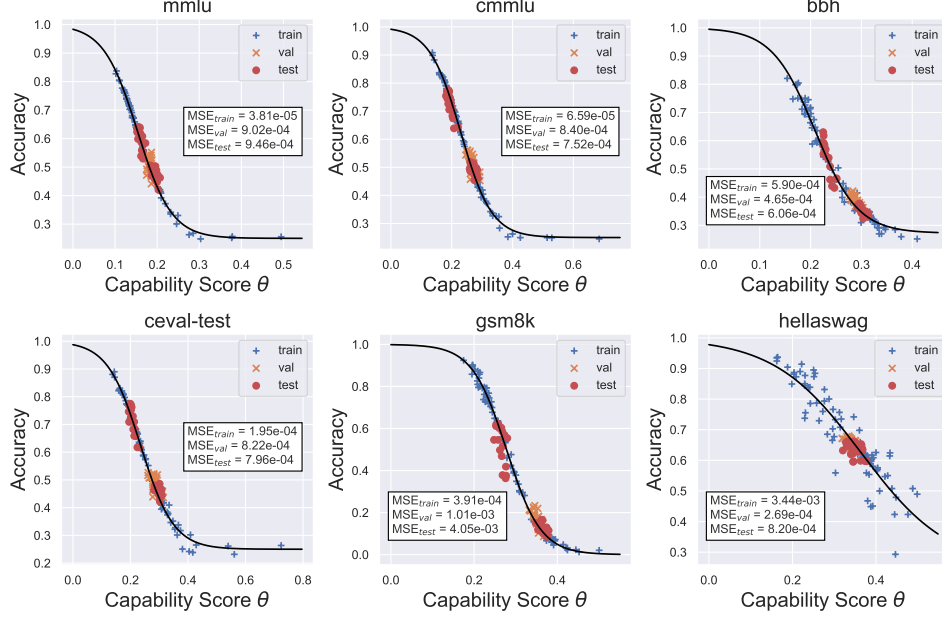


Figure 5: Downstream task prediction results of closed-source models using capability score derived from Capability Saliency Vector.

in Appendix C. To obtain the token cross-entropy loss on specific validation sets, we utilize LMDe-ploy(Contributors, 2023a) framework. To ensure the validation loss better reflects the different capabilities of the models, we randomly sample 100 examples from the six validation sets and mixed them. Additionally, we include 50 examples with chain-of-thought related to BBH. We treated CMMLU and CEval as reflecting the same capability and only sampled from one of these datasets. In total, we gathered 550 examples for optimizing the Capability Saliency Vector. Our method can use only 550 examples as validation set for modeling downstream scaling law.

### Baseline

- All token loss : This method simply calculate the average of all token loss in validation set. Then We use the metric to model downstream scaling law by fitting a sigmoidal function.
- Label token loss : Following the same setup as Llama3 (Dubey et al., 2024), we compute the negative log-likelihood of the correct answers on each task’s test set and calculate its expectation. We then use this metric to fit a sigmoidal function between fitting performance and downstream task performance.

**Training Details** We use InternLM2.5 1.8B (Cai et al., 2024) as the scoring model, replacing its lan-

guage head with a linear layer followed by a non-linear activation function. In practice, we found that increasing model complexity has little impact on the final results. To optimize efficiently, we freeze the language model parameters and only train the linear layer. We train with a learning rate of 1e-3, and most tasks converged to the best performance within 300 steps(training for 1-3 hours on a single A800.), demonstrating the practical efficiency of our method.

Table 1: MSE of Different loss to predict downstream performance.

Task	Test Model Prediction MSE		
	CSV	All token	Label token
mmlu	1.45e-3	2.40e-2	3.31e-2
bbh	2.26e-3	5.81e-2	8.27e-2
gsm8k	6.32e-3	7.46e-2	9.88e-2
hellaswag	4.49e-3	3.15e-3	3.33e-2
ceval	3.71e-3	2.67e-2	3.57e-2
cmmlu	3.32e-3	2.69e-2	3.67e-2

## 4.2 Main Results

**Downstream Task Prediction for Open-source Models** Figure 4 shows the optimization results for different downstream tasks across various model families. It can be observed that by identifying a shared Capability Saliency Vector across

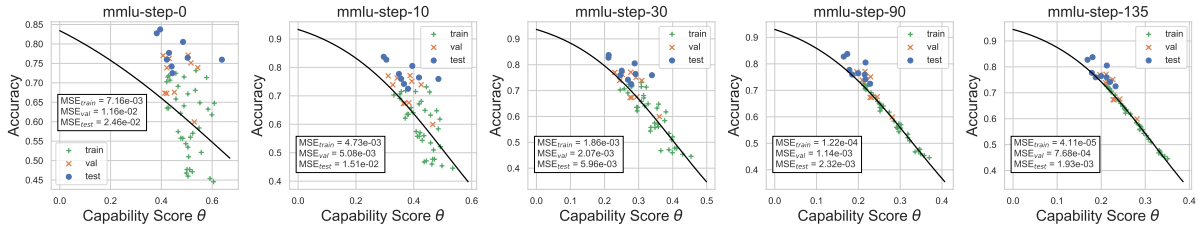


Figure 6: Relationship between capability scores and downstream task performance across different steps of the Capability Saliency Vector optimization process. As the optimization progresses, the capability scores of different model series gradually align, leading to improved predictive performance on the test set.

different model families on validation text, we are able to align the capabilities across different models for downstream tasks. Furthermore, the sigmoidal function fitted using the Capability Saliency Vectors can effectively predict the downstream task performance of models with larger parameters. This indicates that the downstream task capabilities of models also follow a scaling law, which makes their performance predictable. Table 1 and Figure 7 present the prediction MSE of different loss calculation methods. The capability saliency vector significantly improves the predictability of downstream task performance by assigning different importance weights to different token losses. In contrast, the label token loss does not perform well. We believe this is because the loss at the position of the correct answers in the downstream task test set is susceptible to biases in the pretrain data distribution. As a result, this metric fails to align across different model families. We further validated this on checkpoints from our own trained models.

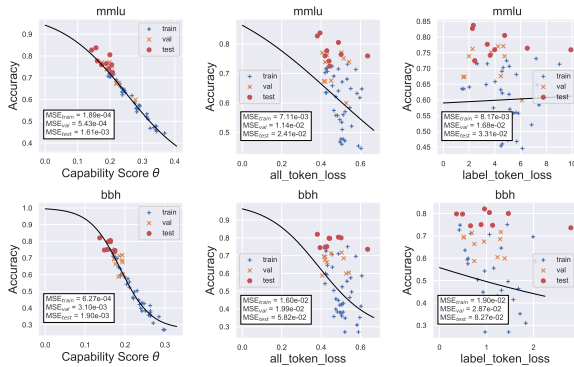


Figure 7: Visualization of prediction results of our method compared to baseline

**Downstream Task Prediction for Closed-source Models** The results in figure 5 demonstrate that by using the Capability Saliency Vector as a proxy,

we can predict downstream performance during the model training process, even when models are trained with different data distribution. During training, we were able to maintain a prediction accuracy for downstream task performance with a mean squared error of  $1e-4$ . This demonstrates that our method provides strong guidance for model training and effectively predicts downstream performance.

### 4.3 Convergence Trend of Capability Saliency Vector Optimization

Figure 6 depicts the optimization trend of model capabilities during the process of Capability Saliency Vector optimization. The figure shows that the model capabilities progressively converge toward a universal curve shared across different models. This convergence demonstrates that the LM cross-entropy loss reliably reflects model capabilities and that a functional relationship exists between LM cross-entropy loss and downstream task performance. Our algorithm successfully models the downstream scaling law across various model families, improving the predictability of downstream performance based on LM cross-entropy loss.

### 4.4 Analysis Studies

**Impact of Validation Set Data Distribution on Capability Saliency Vector Optimization** We found that as the diversity of the validation set distribution increases, the range of meta-capability combinations captured by the model from LM cross-entropy loss also expands. This leads to more accurate capability predictions for downstream scaling law. Therefore, when using loss to measure model capabilities, we recommend considering token losses from a more diverse set of data sources, rather than relying solely on the loss from a single data distribution. The result is shown in Figure 8



**Case Study** In this section, we visualize the Capability Saliency Vectors identified by the automatic optimization algorithm. Figure 9 shows the results of the Capability Saliency Vector optimized for the BBH downstream task. The intensity of the green color represents the importance weight assigned by the Capability Saliency Vector, with darker shades indicating larger weights. We observe that the optimized Capability Saliency Vector highlights the tokens which can measure the model’s reasoning capabilities. The model needs to summarize contextual information and apply reasoning to predict these tokens. These examples demonstrate that our method effectively extracts meta-capabilities that capture the model’s scaling behavior on specific tasks.

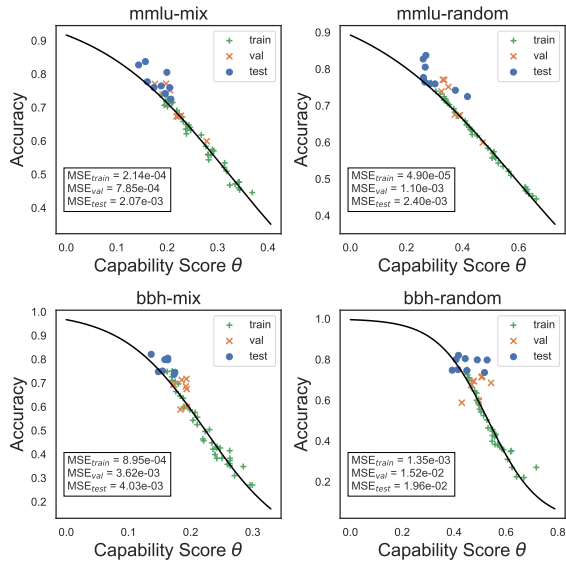


Figure 8: Impact of varying validation set distributions on Capability Saliency Vector optimization.

## 5 Conclusion

In this work, we first demonstrated that different tokens correspond to distinct meta-capabilities, highlighting the importance of accounting for token-level differences when exploring the scaling relationship between LM CE loss and downstream task performance. Furthermore, we proposed an automatic optimization method to identify the Capability Saliency Vector for a text segment. By assigning varying importance weights to the losses of different tokens, our approach enables researchers to model the scaling relationship between loss and downstream tasks. We believe that precisely modeling the scaling relationship between language CE

Let's trace the sequence of ball swaps to determine which ball Eve ends up with at the end of the game : 1 . \*\* C la ire and Dave swap balls \*\* : - Claire now has the yellow ball . - Dave now has the brown ball . 2 . \*\* E ve and Alice swap balls \*\* : - Eve now has the blue ball . - Alice now has the black ball . 3 . \*\* G er tr ude and Eve swap balls \*\* : - Ger tr ude now has the blue ball . - Eve now has the purple ball . 4 . \*\* C la ire and Alice swap balls \*\* : - Claire now has the black ball . - Alice now has the yellow ball . 5 . \*\* Dave and Eve swap balls \*\* : - Dave now has the purple ball . - Eve now has the brown ball . 6 . \*\* Alice and Ger tr ude swap balls \*\* : - Alice now has the blue ball . - Ger tr ude now has the yellow ball . 7 . \*\* F red and Bob swap balls \*\* : - Fred now has the pink ball . - Bob now has the red ball . After following all the sw aps , Eve ends up with the brown ball . Therefore , the correct answer is : (C ) brown ball .

Figure 9: Visualization of the importance weights assigned to different tokens by the Capability Saliency Vector after optimization on the BBH task. Tokens with higher importance weights are highlighted in green. This shows that the Capability Saliency Vector effectively identifies tokens that require reasoning by the model, capturing the model’s reasoning capabilities.

loss and downstream task performance is a meaningful and valuable area for future research.

## 6 Limitation

In this section, we will discuss some limitations of our method. First, while our approach can optimize any given text to identify its Capability Saliency Vector, the effectiveness varies across different texts. Therefore, selecting suitable optimization texts remains an open area for future research. Second, since the optimization needs to be performed simultaneously on all texts, the computation cost increases with the length of the text. Thus, improving the efficiency of the optimization algorithm or exploring simpler modeling approaches is a topic for future investigation. Finally, our experiments are limited to six popular objective evaluation benchmarks. Exploration of more challenging tasks or subjective evaluations is left for future work.

## Acknowledgements

The authors wish to thank the AC and anonymous reviewers for their constructive comments. This work was supported by the China Postdoctoral Science Foundation under Grant Number 2023M741851.

## References

Sanjeev Arora and Anirudh Goyal. 2023. A theory for emergence of complex skills in language models.

*arXiv preprint arXiv:2307.15936.*

- Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. 2024. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.
- Frank B Baker. 2001. *The basics of item response theory*. ERIC.
- Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. 2024. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*.
- 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*.
- LMDeploy Contributors. 2023a. Lmdeploy: A toolkit for compressing, deploying, and serving llm. <https://github.com/InternLM/lmdeploy>.
- OpenCompass Contributors. 2023b. Opencompass: A universal evaluation platform for foundation models. <https://github.com/open-compass/opencompass>.
- Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. 2024. Understanding emergent abilities of language models from the loss perspective. *arXiv preprint arXiv:2403.15796*.
- 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*.
- Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, et al. 2024. Language models scale reliably with over-training and on downstream tasks. *arXiv preprint arXiv:2403.08540*.
- 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*.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.
- Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. 2024. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36.
- Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. 2024. Scaling laws for downstream task performance of large language models. *arXiv preprint arXiv:2402.04177*.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.
- Alex Kipnis, Konstantinos Voudouris, Luca M Schulze Buschoff, and Eric Schulz. 2024. metabench –a sparse benchmark to measure general ability in large language models. *arXiv preprint arXiv:2407.12844*.
- Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023. Cmmlu: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.
- Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, et al. 2024. Rho-1: Not all tokens are what you need. *arXiv preprint arXiv:2404.07965*.
- Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. 2023. Same pre-training loss, better downstream: Implicit bias matters for language models. In *International Conference on Machine Learning*, pages 22188–22214. PMLR.
- Frederic M Lord and Melvin R Novick. 2008. *Statistical theories of mental test scores*. IAP.
- 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.
- David Owen. 2024. How predictable is language model benchmark performance? *arXiv preprint arXiv:2401.04757*.
- Lorenzo Pacchiardi, Lucy G Cheke, and José Hernández-Orallo. 2024. 100 instances is all you need: predicting the success of a new llm on unseen data by testing on a few instances. *arXiv preprint arXiv:2409.03563*.

- Yotam Perlitz, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. 2023. Efficient benchmarking (of language models). *arXiv preprint arXiv:2308.11696*.
- Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. 2024. tinybenchmarks: evaluating llms with fewer examples. *arXiv preprint arXiv:2402.14992*.
- Yangjun Ruan, Chris J Maddison, and Tatsunori Hashimoto. 2024. Observational scaling laws and the predictability of language model performance. *arXiv preprint arXiv:2405.10938*.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.
- Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.
- 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*.
- Mengzhou Xia, Mikel Artetxe, Chunting Zhou, Xi Victoria Lin, Ramakanth Pasunuru, Danqi Chen, Luke Zettlemoyer, and Ves Stoyanov. 2022. Training trajectories of language models across scales. *arXiv preprint arXiv:2212.09803*.
- 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*.
- Qinyuan Ye, Harvey Yiyun Fu, Xiang Ren, and Robin Jia. 2023. How predictable are large language model capabilities? a case study on big-bench. *arXiv preprint arXiv:2305.14947*.
- Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.
- 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*.
- Qiyuan Zhang, Fuyuan Lyu, Xue Liu, and Chen Ma. 2024. Collaborative performance prediction for large language models. *arXiv preprint arXiv:2407.01300*.

## **A Appendix**

### **A Open-source Model Evaluation Setting**

In this section, we present the specific model splits used for Capability Saliency Vector optimization in the evaluation of open-source models. Larger models within the same series were used for validation and testing, as detailed in Table 2. This split ensures that our method is evaluated across models of varying sizes while maintaining consistency within each series for training and validation.

### **B Algorithm Workflow**

Based on the derivation in Section 3.3, we summarise the algorithmic flow of Capability Saliency Vector modeling in Algorithm 1 and 2.

### **C Open-source Model Evaluation Results**

In this section, we present the evaluation results in Table 3 for various tasks on open-source models using OpenCompass.



Table 2: Model split in open-source model evaluation

train	validation	test
Llama-2-{7, 13}B	Llama-2-70B	Yi-1.5-34B
Llama-2-{7, 13}B-chat	Llama-2-70B-chat	Llama-3-70B
Qwen1.5{1.8, 4, 7, 14}B	Qwen1.5-{32, 72}B	Llama-3-70B-Instruct
Qwen1.5{1.8, 4, 7, 14}B-chat	Qwen1.5-{32, 72}B-Chat	Yi-1.5-34B-Chat
Llama3-8B	internlm2-chat-20B	gemma-2-27B
Llama3-8B-Instruct	internlm2-20B	gemma-2-27B-it
Qwen2{1.5, 7}B	internlm2-chat-20B-sft	Qwen2-72B
Qwen2{1.5, 7}B-Instruct	Yi-{34}B	Qwen2-72B-Instruct
Yi-{6}B	Yi-{34}B-chat	internlm2_5-20B
Yi-{6}B-chat		internlm2_5-20B-chat
Yi1.5-{6, 9}B		Yi-1.5-34B-Chat
Yi1.5-{6, 9}B-chat		
Gemma2-2-{2, 9}B		
internlm2-{1.8, 7}B		
internlm2-{1.8, 7}B-chat		
internlm2-{1.8, 7}B-chat-sft		
internlm2.5-{1.8, 7}B		
internlm2.5-{1.8, 7}B-chat		
internlm2.5-{1.8, 7}B-chat-sft		

**Algorithm 1:** Loss Mapping Between Different Tokenizers

---

**Require:** Input text  $C = (c_1, c_2, \dots, c_n)$

**Require:** Source tokenizer  $T_s$  and target tokenizer  $T_t$

**Require:** Source token losses  $L_s = (\ell_1^s, \ell_2^s, \dots, \ell_m^s)$

- 1: Tokenize input with source tokenizer:  $T_s(C) = (\tau_1^s, \dots, \tau_m^s)$
- 2: Initialize character-level losses  $\ell_j^c = 0$  for each character  $c_j \in C$
- 3: **for** each token  $\tau_i^s$  in  $T_s(C)$  **do**
- 4:   Let  $S_i$  be the indices of characters spanned by  $\tau_i^s$
- 5:   **for** each  $j \in S_i$  **do**
- 6:      $\ell_j^c = \frac{\ell_i^s}{|\tau_i^s|}$
- 7:   **end for**
- 8: **end for**
- 9: Tokenize input with target tokenizer:  $T_t(C) = (\tau_1^t, \dots, \tau_k^t)$
- 10: Initialize target token losses  $L_t = (\ell_1^t, \dots, \ell_k^t)$
- 11: **for** each token  $\tau_j^t$  in  $T_t(C)$  **do**
- 12:   Let  $T_j$  be the indices of characters spanned by  $\tau_j^t$
- 13:    $\ell_j^t = \sum_{i \in T_j} \ell_i^c$
- 14: **end for**
- 15: **return** Target token losses  $L_t$

---

---

**Algorithm 2:** Optimization of Capability Saliency Vector and Downstream Scaling Law

---

**Require:** Validation set  $S = \{X_s\}_{s=1}^{|S|}$

**Require:** Language model with score head  $f_\theta$

**Require:** Model Set  $M = \{m\}$

**Require:** Observed downstream performance of different models  $\hat{A}_{t,m}$

- 1: Collect token losses on the validation set of different model  $\{L_m\}$  and mapping the losses from source tokenizer to target tokenizer Using algorithm 1
- 2: **for** each epoch **do**
- 3:   **Step 1: Extract Weights of Capability Saliency Vector**
- 4:   **for** each sample  $X_s = (x_1, x_2, \dots, x_{|X_s|}) \in S$  **do**
- 5:     **for** each token  $x_{s,i}$  in  $X_s$  **do**
- 6:       Compute  $w_{s,i} = f_\theta(x_{s,i} \mid x_{s,<i})$  using the language model scoring head.
- 7:     **end for**
- 8:   **end for**
- 9:   Obtain initial Capability Saliency Vector  $W = \{w_{s,i}\}$ .
- 10: **for** each model  $m$  in Model Set  $M$  **do**
- 11:   **Step 2: Fit Downstream Scaling Law Function**
- 12:   Using equation 1 and 2 to predict downstream task performance  $A_{t,m}$ .
- 13:   Fix  $\theta$  and estimate  $\alpha, \beta$  by minimizing the MSE loss:

$$\min_{\alpha, \beta} \sum_m [A_{t,m} - \hat{A}_{t,m}]^2$$

- 14:   Use the Levenberg-Marquardt algorithm to solve for  $\alpha, \beta$ .
- 15:   **Step 3: Optimize Capability Saliency Vector**
- 16:   Using equation 1 and 2 to predict downstream task performance  $A_{t,m}$ .
- 17:   Update  $\theta$  by minimizing:

$$\min_{\theta} \sum_m [A_{t,m} - \hat{A}_{t,m}]^2$$

- 18:   Use Stochastic Gradient Descent (SGD) to optimize  $\theta$ .
  - 19:   **end for**
  - 20:   Repeat Steps 1-3 until convergence.
  - 21: **end for**
  - 22: **return** Optimized Capability Saliency Vector  $W$  and Downstream Scaling Law parameters  $\alpha, \beta$ .
-

Table 3: Downstream Tasks Evaluation Results(1)

model	bbh	ceval-test	cmmlu	gsm8k	hellaswag	mmlu
Llama-2-7b-hf	38.27	30.13	32.75	16.76	29.29	46.78
Llama-2-13b-hf	45.68	37.38	38.81	29.87	45.06	55.76
llama-2-70b-hf	64.78	49.53	53.05	63.53	55.91	69.87
Qwen1.5-1.8B	27.01	58.67	57.45	34.87	42.32	47.14
Qwen1.5-4B	34.81	66.55	66.38	47.61	55.89	57.03
Qwen1.5-7B	39.87	72.49	71.86	54.36	68.51	62.15
Qwen1.5-14B	50.38	76.93	76.95	63.53	83.86	69.1
Qwen1.5-32B	67.47	82.5	81.58	72.71	87.28	73.88
Qwen1.5-72B	58.81	83.03	83	79.53	90.41	77.02
Qwen2-7B	54.28	82.22	83.67	73.77	73.74	70.27
Qwen2-72B	74.52	89	90.82	89.54	93.46	83.73
Yi-6B	44.82	70.78	74.01	39.58	66.83	64.08
Yi-34B	66.37	80.93	82.79	67.7	83.83	76.26
Yi-1.5-6B	57.55	66.93	69.68	61.33	70.79	64.88
Yi-1.5-9B	71.09	72.71	74.11	74.53	76.64	71.52
Yi-1.5-34B	75.06	82.24	83.55	79.98	84.92	77.7
Qwen1.5-1.8B-Chat	27.03	55.19	48.3	29.57	42.32	45.39
Qwen1.5-4B-Chat	43.19	61.37	58.22	46.02	60.74	56.01
Qwen1.5-7B-Chat	35.19	68.18	67.98	55.88	69.8	61.78
Qwen1.5-14B-Chat	55.58	74.67	75.29	64.82	80.03	68.08
Qwen1.5-32B-Chat	68.55	80.66	80.17	79.15	88.42	75.12
Qwen1.5-72B-Chat	71.73	81.37	82.15	79.68	88.99	77.1
Qwen2-7B-Instruct	64.56	82.11	80.84	87.29	78.57	70.68
Qwen2-72B-Instruct	82.07	87.02	89.88	92.42	93.7	82.73
Yi-6B-Chat	30.98	70.69	72.74	42.46	64.27	63.22
Yi-34B-Chat	60.47	78.58	80.33	72.71	78.69	73.91
Yi-1.5-6B-Chat	58.68	68.05	68.59	75.36	75.66	64.76
Yi-1.5-9B-Chat	69.61	73.19	74.51	79.23	83.14	71.35
Yi-1.5-34B-Chat	73.63	80.83	81.05	85.82	87.33	75.95
internlm2-1_8b	36.03	44.79	45.27	30.4	44.86	45.99
internlm2-7b	63.56	63.54	66.17	69.98	89.52	65.84
internlm2-20b	71.29	67.28	68.28	76.8	91.41	67.58
internlm2-chat-1_8b	37.31	47.06	46.19	39.2	59.71	47.72
internlm2-chat-1_8b-sft	38.23	47.21	46.17	37.83	60.86	47.7
internlm2-chat-7b	59.96	58.85	62.66	72.1	84.44	63.51
internlm2-chat-7b-sft	59.35	58.99	62.57	70.51	85.01	63.51
internlm2-chat-20b	69.53	63.03	65.85	76.19	88.16	67.36
internlm2-chat-20b-sft	69.31	63.2	66.04	77.03	88.68	67.28
internlm2_5-1_8b-chat	42.58	61.25	62.3	53.3	49.97	50.98
internlm2_5-7b	72.64	77.21	78.95	74.83	73.49	71.5
internlm2_5-7b-chat	74.8	77.38	78.04	84	67.74	71.82
internlm2_5-20b-chat	79.77	79.75	79.18	88.1	66.53	72.49
internlm2_5-1_8b	42.65	63.97	65.5	38.82	56.64	53.44
internlm2_5-20b	79.83	81.94	82.25	82.64	79.35	74.2

Table 4: Downstream Tasks Evaluation Results(2)

model	bbh	ceval-test	cmmlu	gsm8k	hellaswag	mmlu
Meta-Llama-3-8B	59.69	48.83	50.95	54.28	50.86	66.43
Meta-Llama-3-70B	79.16	66.56	68.36	69.98	80.6	79.35
Llama-2-7b-chat-hf	41.51	30.2	32.62	28.13	48.94	44.56
Llama-2-13b-chat-hf	49.61	33.76	37.01	42.08	61.43	52.1
Meta-Llama-3-8B-Instruct	52.5	49.93	51.89	79.3	73.29	67.16
Meta-Llama-3-70B-Instruct	80.45	66.91	70.11	90.22	87.72	80.52
gemma-2-2b-it	46.44	41.82	44.86	55.65	62.43	58.39
gemma-2-9b-it	75.07	57.32	59.49	86.66	70.29	73.14
gemma-2-27b-it	80.05	61.71	62.5	89.99	75.96	76.45
gemma-2-27b	74.77	60.84	61.53	80.97	75.55	75.97
gemma-2-2b	42.12	39.44	39.31	33.51	65.91	54.35
gemma-2-9b	70.63	57.63	59.02	72.86	72.93	72.44
Llama-2-70b-chat-hf	59.94	32.21	43.07	60.65	74.18	59.93
Qwen2-1.5B-Instruct	38.31	68.27	68.12	63.53	55.38	55.73
Qwen2-1.5B	35.75	69.32	70.58	59.06	46.76	57.45