Supplementary Materials

1. Detailed Description of Textual Feature-Based Baseline Methods

In this section, we supplement the implementation details of the textual feature space baseline methods (B1/B2) in Sec. 4.1 .

1.1. POS (Part of Speech) Tagging Analysis

The POS (Part of Speech) tagging is a process to mark up the words in text format for a particular part of a speech based on its definition and context. The “part of speech” is a typical low-level textual feature, which could potentially reflect the characteristic of the representative textual subset. We conduct the POS tagging analysis on COCO annotations. We use the NLTK’s (Nature Language Toolkit) tokenizer and POS tagging tool to analyze the POS distribution. We present the POS that occupies more than 1% in the legend. The “NN” represents “singular noun”, “DT” represents “determiner” (e.g., “my”), “IN” represents “Preposition and conjunction” (e.g., “at/in”), “JJ” represents “Adjective”, “.” represents the punctuation, “VBG/VBZ” represents different forms of verbs, “NNS” represents “Proper and plural noun”, and “CC” represents “conjunction of coordinating” (e.g., “that/which”). As demonstrated in Fig. 1, the POS distributions of the entire set and subsets of 100 items with high/low standalone KD are **quite similar**. Therefore, **it is not feasible to select representative subsets according to this characteristic**.

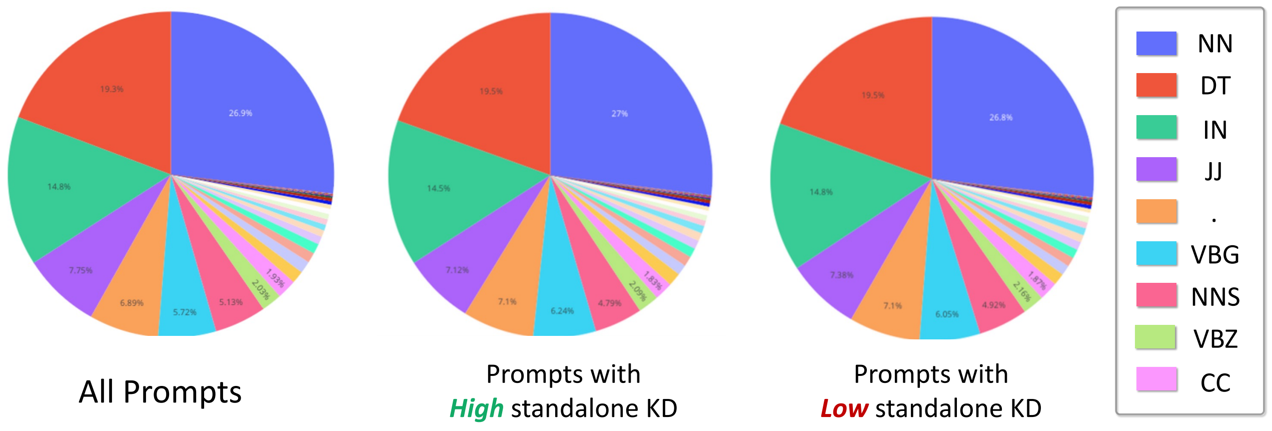


Figure 1. **Comparison of pos tagging analysis for the entire COCO annotations, 100 item subsets with high/low standalone KD.**

1.2. Semantic Class Diversity

To effectively illustrate the inability to select a representative subset through semantic class diversity, we conduct comprehensive experiments on this matter. We partition CLIP/BERT features into 10 clusters using the K-means algorithm, ensuring that prompts in each cluster encompass similar semantic information categories (e.g., humans, cats, etc). Subsequently, we selected $N'/10$ prompts from each cluster based on the Euclidean distance to the cluster centroid (near or far), or through random selection. As depicted in Tab. 1 and Tab. 2, all the methods yields results similar to random sampling. Therefore, **ensuring semantic class diversity does not lead to an improvement in performance.**

Methods	$N' = 10$		$N' = 100$	
	Train	Test	Train	Test
RS	0.355	0.346	0.706	0.692
Near Centroid	0.455	0.451	0.474	0.570
Far Centroid	0.551	0.539	0.617	0.713
RS-each cluster	0.341	0.313	0.617	0.682

Table 1. **Comparisons of the Kendall's Tau of CLIP-Score for subsets acquired by clustering CLIP features.**

Methods	$N' = 10$		$N' = 100$	
	Train	Test	Train	Test
RS	0.355	0.346	0.706	0.692
Near Centroid	0.057	0.335	0.490	0.540
Far Centroid	0.285	-0.116	0.659	0.545
RS-each cluster	0.228	0.251	0.686	0.651

Table 2. **Comparisons of the Kendall's Tau of CLIP-Score for subsets acquired by clustering Bert features.**

Algorithm 1: Greedy Search Algorithm

```
Input: Whole textual dataset $\mathcal{P} = \{p_i | i = 1, ..., N\}$, The corresponding features of dataset $\mathcal{F} = \{f_i | i = 1, ..., N\}$, The mean $\mu_{all}$ and covariance $\Sigma_{all}$ of multivariate Gaussian distribution of $\mathcal{F}$;
Ouput: Subsets $\hat{\mathcal{P}}$ of size $N'$;
Random Initialize $\hat{\mathcal{P}} \leftarrow p_k, \hat{\mathcal{F}} \leftarrow f_k (1 \le k \le N)$;
for $i \leftarrow 2$ to $N'$ do
    $index = INF, D_{min} = INF$;
    for $f_j$ in $\mathcal{F} - \hat{\mathcal{F}}$ do
        $\mathcal{F}_j = \hat{\mathcal{F}} \cup f_j$;
        Calculate $\mu_j$ and $\Sigma_j$ of $\mathcal{F}_j$;
        // KL distance between $\mathcal{F}_j$ and $\mathcal{F}$
        $D_j = KL(\mu_j, \Sigma_j, \mu_{all}, \Sigma_{all})$
        if $D_{min} > D_j$ then
            $index = j, D_{min} = D_j$;
    $\hat{\mathcal{P}} = \hat{\mathcal{P}} \cup p_{index}, \hat{\mathcal{F}} = \hat{\mathcal{F}} \cup f_{index}$;
return $\hat{\mathcal{P}}$
```

1.3. Statistical Property Similarity

As shown in Algo. 1, we design a greedy search algorithm that selects the prompt that minimizes the KL divergence between the CLIP feature distribution of the subset and the entire set in each iteration. Since the algorithm requires computing the KL divergence, the feature distribution is assumed to be a multivariate Gaussian distribution. To better fit the feature distribution, we expand the Gaussian distribution into the Gaussian mixture distribution. It can be depicted in Fig. 2, when the Gaussian Mixture distribution contains five Gaussian functions, it effectively fits the features. Then, we run the algorithm in each Gaussian function, identifying $N'/5$ prompts in each function (a total of N' prompts). We show the results of different items in Tab. 3. **It was evident that matching the statistical property similarity does not improve the ranking outcome**.

Methods	$N' = 100$		$N' = 200$		$N' = 500$		$N' = 1000$	
	Train	Test	Train	Test	Train	Test	Train	Test
RS	0.706	0.692	0.784	0.763	0.857	0.829	0.896	0.867
Gaussian	0.714	0.696	0.764	0.791	0.873	0.847	0.913	0.872
Gaussian Mixture	0.565	0.420	0.809	0.770	0.799	0.802	0.881	0.880

Table 3. **Comparisons of the Kendall's Tau of CLIP-Score for subsets acquired by statistical property similarity.**

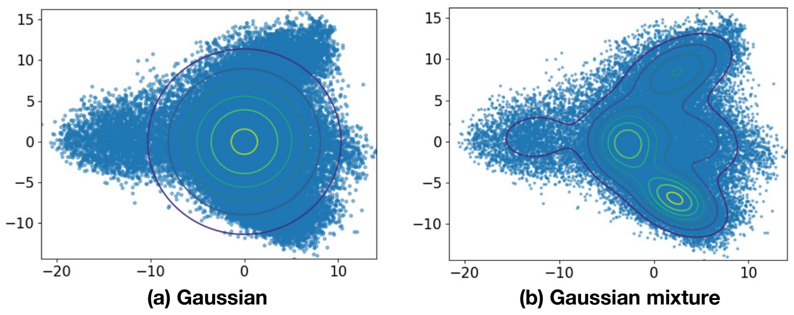


Figure 2. **T-SNE Feature Visualization of Gaussian and Gaussian Mixture (#component = 5) distributions fitted to all features.**

2. FlahEval's Capability of Estimating Model Scores

In some scenarios, users may require the exact metric scores of models rather than the relative model rankings. FlashEval also supports fitting model scores using a small subset of N' **by searching for the mean square error**. As shown in Fig. 3 and Fig. 4, we demonstrate a comparison of our method against random sampling for model scores prediction. It is evident that our method can accurately estimate model scores even with smaller N' and it generalizes well to testing models across three sub-tasks.

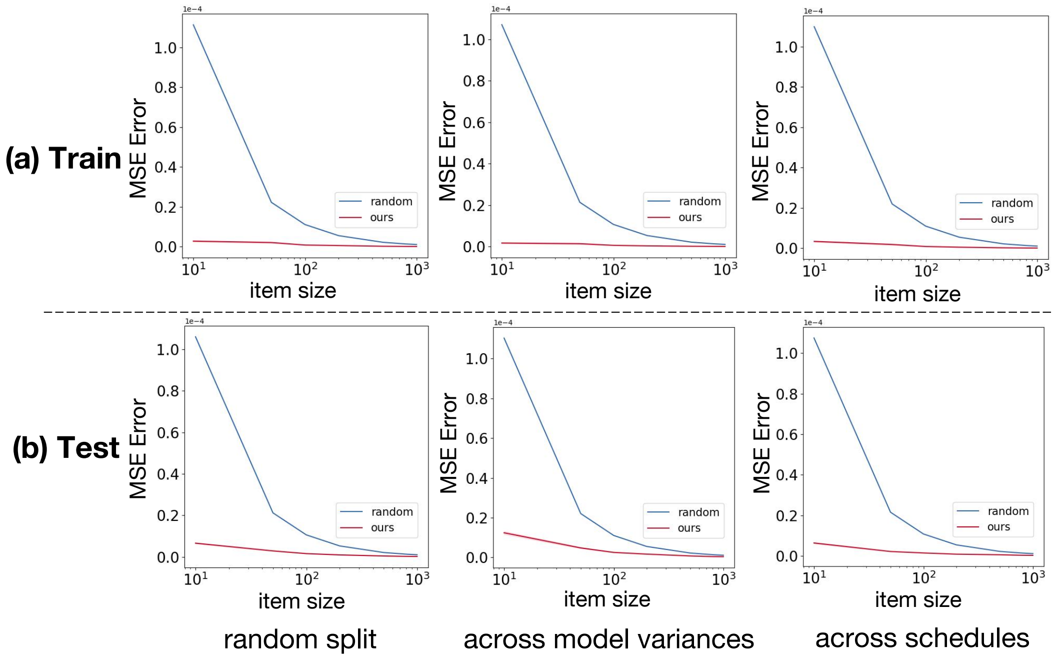


Figure 3. **Comparisons with random sampling of the estimation quality for CLIP-Score on COCO dataset.** Small MSE Error is better.

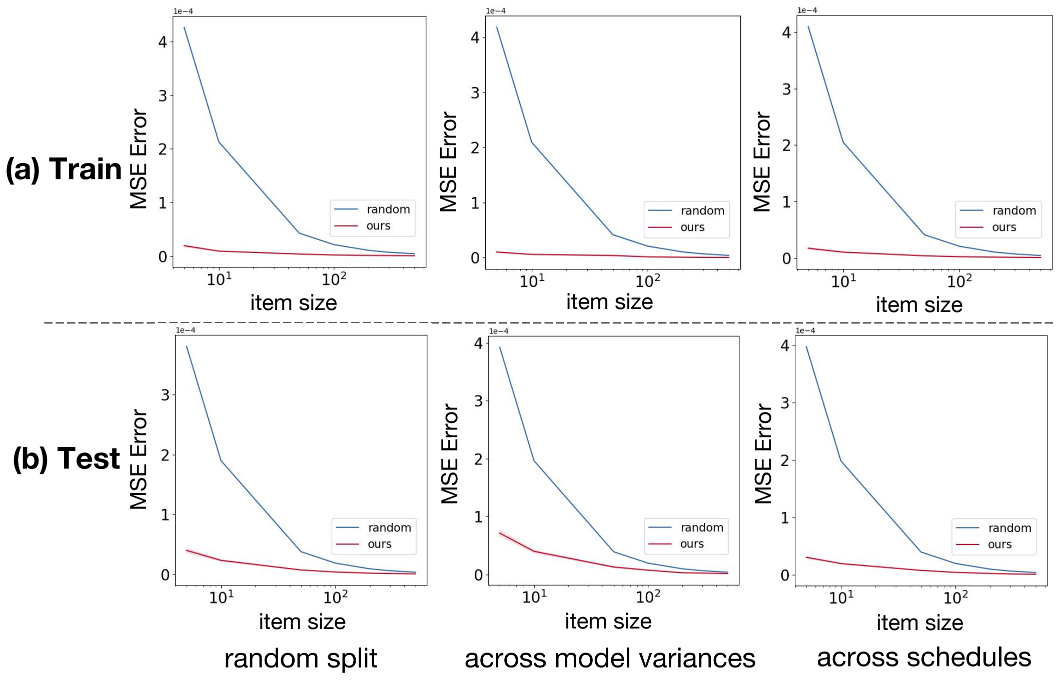


Figure 4. **Comparisons with random sampling of the estimation quality for CLIP-Score on DiffusionDB dataset.** Small MSE Error is better.

3. Analysis of Alternative Approaches to Acquire Representative Subsets

In the main paper, we recognize that "some subsets with high KD has large estimation error", which reflects such sets lack precise estimation of the performance. Therefore, an intuitive question arises "could we use the estimation error as the metric for recognizing representative subsets?". We could measure the "estimation error" by the MSE (Mean Squared Error) loss between the ground-truth performance (averaged on the entire set) and the subset's performance $S_j^{'} = \sum_{i\in\mathcal{I}} \frac{1}{N'}\mathcal{S}_{ij}$ for each model setting. It could be described as follows (N_m is the number of model settings):

$$\Omega_{EA} = \frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{\mathcal{S}}_j - S_j^{'})^2. \tag{1}$$

In this section, we investigate this assumption. We discuss some intuitive alternative approaches to acquire representative subsets based on estimation error ("prompts with least estimation error","sets with least estimation error") and demonstrate their failure. Furthermore, We analyze the underlying reason for their failure and why the FlashEval search method is necessary.

As shown in Fig. 5, we visualize the scores of all prompts in COCO dataset across two randomly selected models. It is noticeable that scores obtained by prompts for a certain model generally follows a Gaussian distribution. The mean of the gaussian distribution is the "ground-truth" model performance evaluation across the entire set. Therefore, if there exists prompts/subsets that could correctly approximate the mean of the gaussian distribution for all models, using the prompt/subsets for evaluation will have both high KD values and low estimation error.

Choose prompts with lowest estimation errors. To validate this existence of such prompts, we separately select 50 prompts with high KD values and 50 prompts with low estimation errors, examining their performance in the other criterion. From Tab. 4, we observe that prompts with high KD values have large estimation errors, while prompts with low estimation errors also exhibit small KD values. We conclude that **single prompt could not simultaneously satisfy high KD and estimation error, which means that "prompt that approximates the mean for all models" does not exist**. Therefore, simply choosing prompts with least estimation error to construct subsets is not feasible.

Choose subsets with lowest estimation errors. Similarly, we design experiments to validate the existence of the subsets that could precisely estimate the performance of each model. We generate $10{,}000{,}000$ random samplings at various N' values, and select a set with the highest KD value and a set with the lowest estimation error. From Fig. 6, it is evident that the set we discovered with the lowest error reaches only a KD value of roughly around 0.7 in training models when $N' = 10$. However, the subset with the highest KD value which b3-set discovered can achieve KD values above 0.8. Therefore, simply choosing subsets with least estimation error to construct subsets is also not feasible.

The underlying reason for their failure: As discussed above, in Fig. 6, we can see that when N' is small, the subset chosen based on estimation error performs poorly, indicating a lack of consistency between KD values and estimation errors. However, as N increases, the subsets acquired by "error-set" achieves relatively high KD. To further validate the above findings, we present the KD values and estimation errors with respect to N' in Fig. 7. We discover that the MSE error decreases sharply with $N' < 500$. Such phenomenon reveals the potential reason for "high KD and low estimation error could not be simultaneously satisfied for a smaller N'". When the N' is small, the estimation error is still relatively large (larger than 0.002), in which case, relatively lower estimation error does not guarantee higher KD. However, as N' increases, the estimation error is sufficiently small to ensure high correlation between lower estimation error and higher KD.

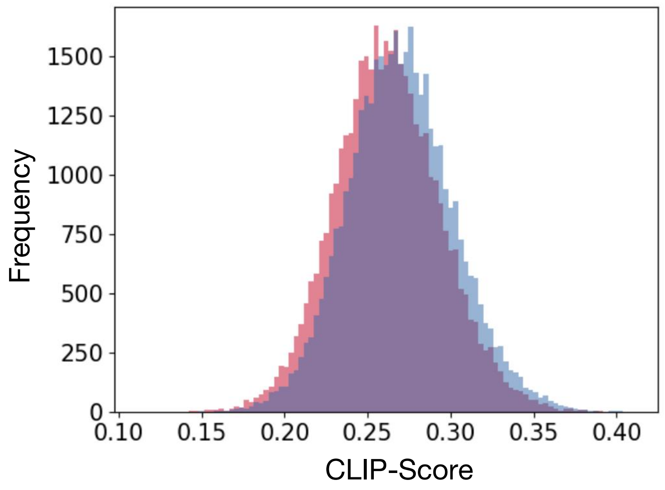


Figure 5. **Visualization of CLIP-Score of all prompts for two models.** The x-axis representing CLIP-Score values and the y-axis indicating the count of prompts achieving that value.

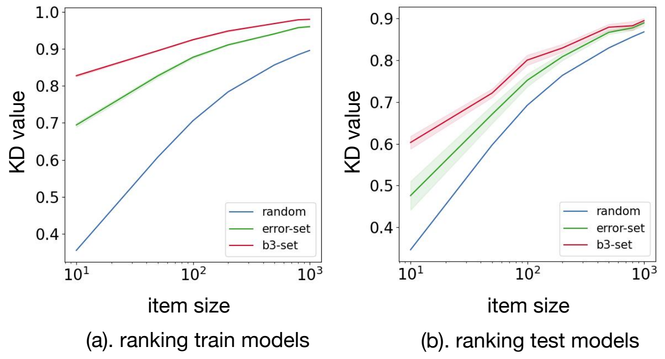
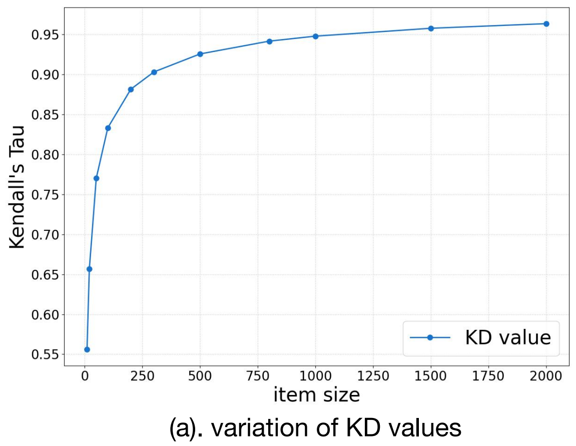


Figure 6. **Comparisons of set-based baselines using Kendall's Tau for CLIP-Score on COCO dataset.** The shaded area are standard errors.



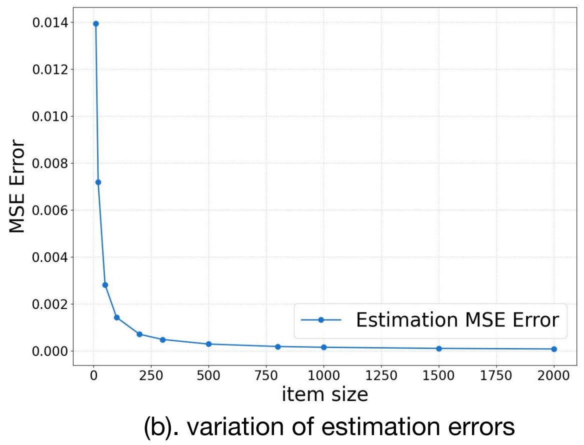


Figure 7. **The variation of average KD values and estimation errors with increasing N'.** The blue dots represent the average values obtained from $100{,}000$ random samplings at the current N'.

Methods	Kendall's Tau↑	Estimation Error↓
All prompts mean	0.269/0.201	0.142/0.138
prompts with low errors	0.391/0.245	0.052/0.062
prompts with high KD values	0.678/0.463	0.117/0.117

Table 4. **The average performance of the top 50 prompts in one criterion on the other criterion for training/testing models.** The "All prompts mean" represents the averaged value for all independent prompts.

4. Qualitative Results of FlashEval Acquired Subsets

In this section, we present and analyze the FlashEval searched textual subset, to demonstrate the searched subset effectively distinguishes different model settings. We choose the $N' = 10$ item subset for the COCO annotations. Tab. 5 and Tab. 6 respectively outline the prompts included in the representative set for CLIP and ImageReward. Additionally, we show the images generated by prompts across different models in Fig. 8 and Fig. 9. The upper row displays the generated images under different schedules of the model "dreamlike", sorted by the ground-truth metric values (the right has higher metric values). As could be seen, the image visual quality for this prompt ("A plate with rice topped with scallops and a side of broccoli.") aligns with the ground-truth performance. It verifies that the searched representative subsets contain prompts correctly reflecting the model performance. Similarly, the lower row demonstrates that when using the same schedule, the prompt could also correctly rank different models. Aside from the CLIP-Score, we also present the generated images picked by human-preference-based metric ImageReward in Fig. 9, similar alignment could be witnessed.

item size	prompts in the representative set
$N' = 10$	"A plate with rice topped with scallops and a side of broccoli." "A decorative Asian lantern sculpture in the garden with flower ornaments." "A man is riding his skateboard down the road." "A close-up of a plate of food that has been eaten." "A small pizza in the middle of a table." "A white plate topped with a hot dog, french fries and condiments." "A black and white cat sitting on top of a pile of clothes." A giraffe and a zebra are on the grass near trees & cars. "A busy bus station with ramp going downstairs." "Heavily loaded green truck with passengers on back in roadway in urban area."

Table 5. **Prompts in the representative set of CLIP-Score when $N' = 10$ on COCO dataset.**

item size	prompts in the representative set
$N' = 10$	"Two giraffes standing around on the grassy plains." "there is someone holding a remote in there hand " "A man adjusts numbers at a tennis match." "A skateboarder catches major air during this stunt." "A little boy in a yellow shirt feeding a giraffe. " "An old building with two rusty, very old pick up trucks parked in front." "A little girl sitting on top of a bed next to a lamp." "An old rusty fire hydrant standing on a cracked sidewalk." "A man driving a car across an airport runway." "The german shepherd is guarding the front of the barred door,"

Table 6. **Prompts in the representative set of ImageReward when $N' = 10$ on COCO dataset.**

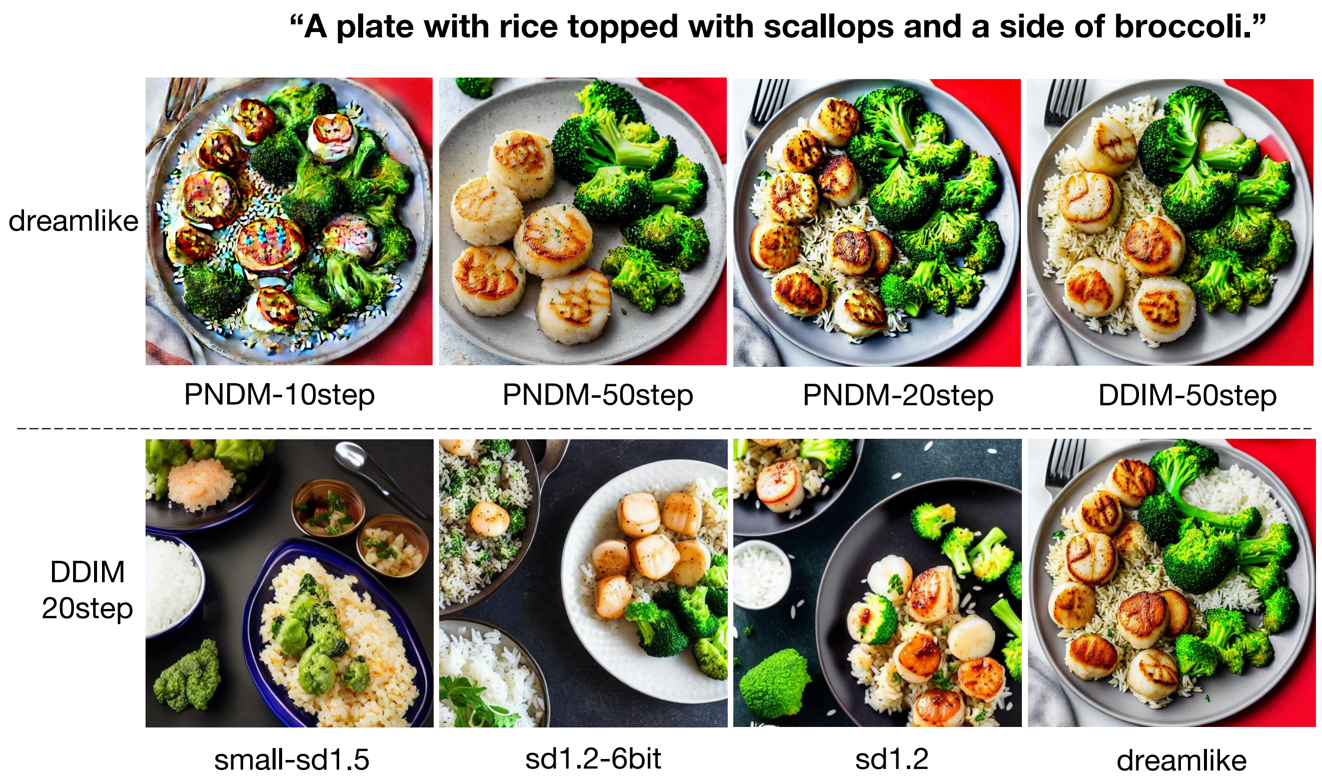


Figure 8. **Visualization of images generated by the prompt from the representative set of CLIP-Score across various models.** The first row represents variations among different schedules in the same model, while the second row depicts variances among different model architectures or parameters under the same schedule.

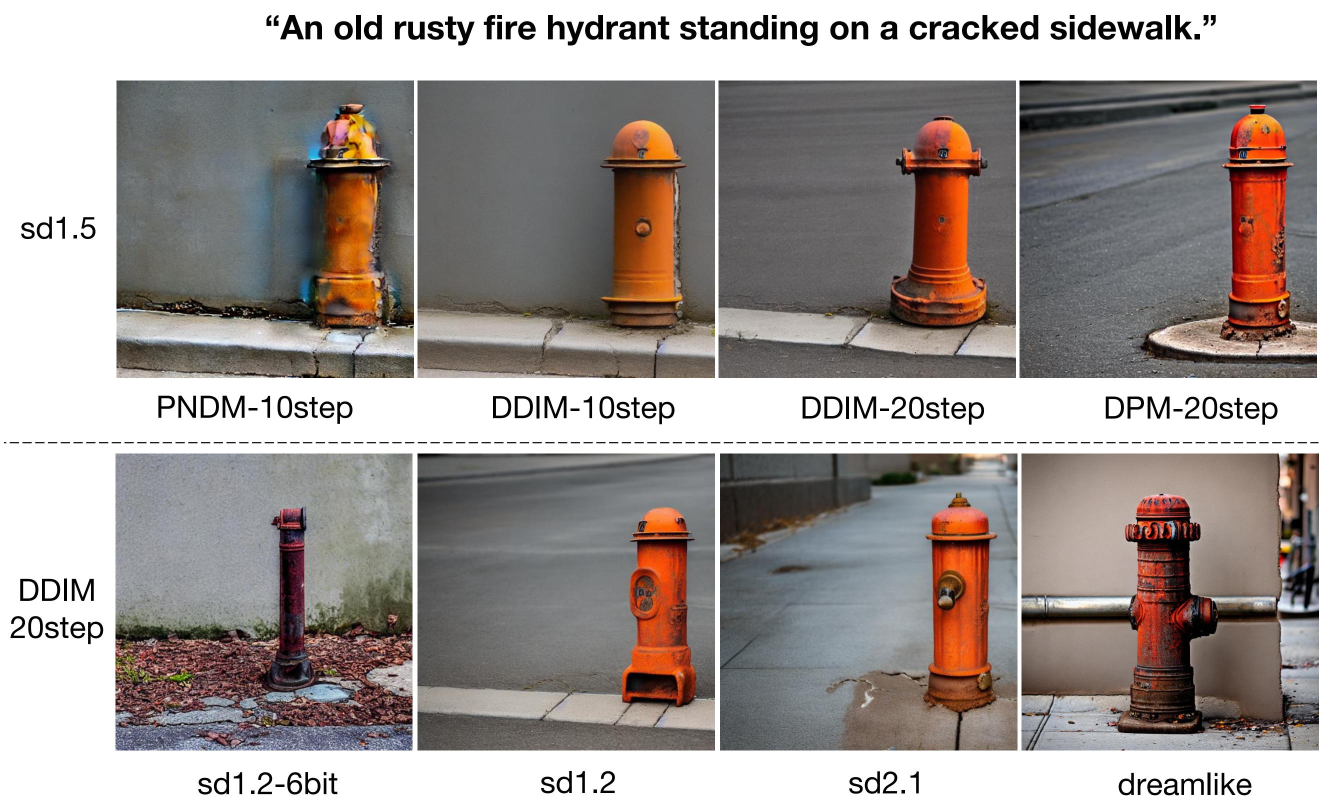


Figure 9. **Visualization of images generated by the prompt from the representative set of ImageReward across various models.** The first row represents variations among different schedules in the same model, while the second row depicts variances among different model architectures or parameters under the same schedule.

5. Verification of the Generelization Ability of FlashEval

As mentioned in Table.2 and Table.3 in the main paper, we design 3 distinct model settings train-test split to verify the generalization ability of FlashEval. They act as 3 diverse subtask to test the generalization ability across random settings, across models, and across schedules. In this section, we firstly provide a detailed description of the 3 train-test split. Then, we present the results on these subtasks on COCO annotations (Fig. 10, Fig. 11, Fig. 12, Fig. 13, and Fig. 14) and DiffusionDB datasets (Fig. 15, Fig. 16, Fig. 17, Fig. 18, and Fig. 19).

5.1. Detailed List of Models

In Tabs. 7 to 9, we present detailed lists of models encompassed by both the training and testing models for each subtask. We conduct the search on the training models and validate the ranking correlation on the testing models to verify the generalizaiton ability of FlashEval. The "random split" randomly splits the model setting zoo into two halves, the models of the training and testing splits have diverse model variants, solver, and timesteps. The "across model variants" uses only 4 models (SD-1.4, SD-1.5, SD-1.5-6bit,SD-2.1) as the training models, and the rest as the test models. The "across schedules" uses the "PNDM-10", "DDIM-10", "DDIM-20", "DDIM-50" as the train models.

5.2. Results under Different Model Settings Split on COCO annotations

We present the comparative results of all methods across various metrics in Figs. 10 to 14. It is evident that FlashEval consistently outperforms others in all scenarios, which demonstrates the generalization ability of FlashEval across diverse model or schedules.

5.3. Results under Different Model Settings Split on DiffusionDB

As mentioned in Sec. 5.1 of the main paper, due to the substantial size of the complete diffusionDB dataset, it is not feasible to iterate through all prompts (2,000,000) to get the ground-truth performance. Considering the cost, we select a relatively large amount of samples (5000) and use the averaged performance on them as the proxy "ground-truth" of DiffusionDB (treating the 5,000 prompts as the "whole set" to further condense it into smaller representative subset).

As could be witnessed from the Figs. 15 to 19, our acquired subsets achieves superior evaluation compared with the random sample baseline (adopted by recent literature Lee *et al.* [41]) quality for different metrics across different model splits.

training models			testing models		
model	solver	step	model	solver	step
stablediffusion1.2	PNDM	10	stablediffusion1.2	PNDM	20
stablediffusion1.2	DDIM	10	stablediffusion1.2	PNDM	50
stablediffusion1.2	DDIM	20	stablediffusion1.2-6bit	DDIM	10
stablediffusion1.2	DDIM	50	stablediffusion1.2-8bit	DDIM	10
stablediffusion1.2	DPM	10	stablediffusion1.2-8bit	DPM	10
stablediffusion1.2	DPM	20	stablediffusion1.2-8bit	DPM	20
stablediffusion1.2-6bit	DDIM	20	stablediffusion1.4	PNDM	20
stablediffusion1.2-6bit	DDIM	50	stablediffusion1.4	PNDM	50
stablediffusion1.2-6bit	DPM	10	stablediffusion1.4	DDIM	20
stablediffusion1.2-6bit	DPM	20	stablediffusion1.4	DPM	20
stablediffusion1.2-8bit	DDIM	20	stablediffusion1.4-6bit	DDIM	20
stablediffusion1.2-8bit	DDIM	50	stablediffusion1.4-6bit	DPM	20
stablediffusion1.4	PNDM	10	stablediffusion1.4-8bit	DDIM	10
stablediffusion1.4	DDIM	10	stablediffusion1.4-8bit	DDIM	20
stablediffusion1.4	DDIM	50	stablediffusion1.4-8bit	DDIM	50
stablediffusion1.4	DPM	10	stablediffusion1.4-8bit	DPM	20
stablediffusion1.4-6bit	DDIM	10	stablediffusion1.5	PNDM	20
stablediffusion1.4-6bit	DDIM	50	stablediffusion1.5	PNDM	50
stablediffusion1.4-6bit	DPM	10	stablediffusion1.5	DDIM	50
stablediffusion1.4-8bit	DPM	10	stablediffusion1.5	DPM	10
stablediffusion1.5	PNDM	10	stablediffusion1.5-6bit	DDIM	10
stablediffusion1.5	DDIM	10	stablediffusion1.5-6bit	DDIM	50
stablediffusion1.5	DDIM	20	stablediffusion1.5-6bit	DPM	20
stablediffusion1.5	DPM	20	stablediffusion1.5-8bit	DDIM	20
stablediffusion1.5-6bit	DDIM	20	stablediffusion1.5-8bit	DPM	10
stablediffusion1.5-6bit	DPM	10	stablediffusion1.5-8bit	DPM	20
stablediffusion1.5-8bit	DDIM	10	small-stablediffusion1.5	PNDM	20
stablediffusion1.5-8bit	DDIM	50	small-stablediffusion1.5	PNDM	50
small-stablediffusion1.5	PNDM	10	small-stablediffusion1.5	DPM	20
small-stablediffusion1.5	DDIM	10	stablediffusion2.1	PNDM	10
small-stablediffusion1.5	DDIM	20	stablediffusion2.1	PNDM	20
small-stablediffusion1.5	DDIM	50	stablediffusion2.1	DDIM	10
small-stablediffusion1.5	DPM	10	stablediffusion2.1	DDIM	50
stablediffusion2.1	PNDM	50	stablediffusion2.1	DPM	10
stablediffusion2.1	DDIM	20	stablediffusion2.1	DPM	20
dreamlike-photoreal	PNDM	20	dreamlike-photoreal	PNDM	10
dreamlike-photoreal	DDIM	10	dreamlike-photoreal	PNDM	50
dreamlike-photoreal	DDIM	20	dreamlike-photoreal	DPM	10
dreamlike-photoreal	DDIM	50	dreamlike-photoreal	DPM	20

Table 7. **The detailed list of models in sub-task with "random split".**

training models			testing models		
model	solver	step	model	solver	step
stablediffusion1.4	PNDM	10	stablediffusion1.2	PNDM	10
stablediffusion1.4	PNDM	20	stablediffusion1.2	PNDM	20
stablediffusion1.4	PNDM	50	stablediffusion1.2	PNDM	50
stablediffusion1.4	DDIM	10	stablediffusion1.2	DDIM	10
stablediffusion1.4	DDIM	20	stablediffusion1.2	DDIM	20
stablediffusion1.4	DDIM	50	stablediffusion1.2	DDIM	50
stablediffusion1.4	DPM	10	stablediffusion1.2	DPM	10
stablediffusion1.4	DPM	20	stablediffusion1.2	DPM	20
stablediffusion1.5	PNDM	10	stablediffusion1.2-6bit	DDIM	10
stablediffusion1.5	PNDM	20	stablediffusion1.2-6bit	DDIM	20
stablediffusion1.5	PNDM	50	stablediffusion1.2-6bit	DDIM	50
stablediffusion1.5	DDIM	10	stablediffusion1.2-6bit	DPM	10
stablediffusion1.5	DDIM	20	stablediffusion1.2-6bit	DPM	20
stablediffusion1.5	DDIM	50	stablediffusion1.2-8bit	DDIM	10
stablediffusion1.5	DPM	10	stablediffusion1.2-8bit	DDIM	20
stablediffusion1.5	DPM	20	stablediffusion1.2-8bit	DDIM	50
stablediffusion1.5-6bit	DDIM	10	stablediffusion1.2-8bit	DPM	10
stablediffusion1.5-6bit	DDIM	20	stablediffusion1.2-8bit	DPM	20
stablediffusion1.5-6bit	DDIM	50	stablediffusion1.4-6bit	DDIM	10
stablediffusion1.5-6bit	DPM	10	stablediffusion1.4-6bit	DDIM	20
stablediffusion1.5-6bit	DPM	20	stablediffusion1.4-6bit	DDIM	50
stablediffusion1.5-8bit	DDIM	10	stablediffusion1.4-6bit	DPM	10
stablediffusion1.5-8bit	DDIM	20	stablediffusion1.4-6bit	DPM	20
stablediffusion1.5-8bit	DDIM	50	stablediffusion1.4-8bit	DDIM	10
stablediffusion1.5-8bit	DPM	10	stablediffusion1.4-8bit	DDIM	20
stablediffusion1.5-8bit	DPM	20	stablediffusion1.4-8bit	DDIM	50
stablediffusion2.1	PNDM	10	stablediffusion1.4-8bit	DPM	10
stablediffusion2.1	PNDM	20	stablediffusion1.4-8bit	DPM	20
stablediffusion2.1	PNDM	50	small-stablediffusion1.5	PNDM	10
stablediffusion2.1	DDIM	10	small-stablediffusion1.5	PNDM	20
stablediffusion2.1	DDIM	20	small-stablediffusion1.5	PNDM	50
stablediffusion2.1	DDIM	50	small-stablediffusion1.5	DDIM	10
stablediffusion2.1	DPM	10	small-stablediffusion1.5	DDIM	20
stablediffusion2.1	DPM	20	small-stablediffusion1.5	DDIM	50
			small-stablediffusion1.5	DPM	10
			small-stablediffusion1.5	DPM	20
			dreamlike-photoreal	PNDM	10
			dreamlike-photoreal	PNDM	20
			dreamlike-photoreal	PNDM	50
			dreamlike-photoreal	DDIM	10
			dreamlike-photoreal	DDIM	20
			dreamlike-photoreal	DDIM	50
			dreamlike-photoreal	DPM	10
			dreamlike-photoreal	DPM	20

Table 8. **The detailed list of models in sub-task with "across model variances".**

training models			testing models		
model	solver	step	model	solver	step
stablediffusion1.2	PNDM	10	stablediffusion1.2	PNDM	20
stablediffusion1.2	DDIM	10	stablediffusion1.2	PNDM	50
stablediffusion1.2	DDIM	20	stablediffusion1.2	DPM	10
stablediffusion1.2	DDIM	50	stablediffusion1.2	DPM	20
stablediffusion1.2-6bit	DDIM	10	stablediffusion1.2-6bit	DDIM	50
stablediffusion1.2-6bit	DDIM	20	stablediffusion1.2-6bit	DPM	20
stablediffusion1.2-6bit	DPM	10	stablediffusion1.2-8bit	DDIM	50
stablediffusion1.2-8bit	DDIM	10	stablediffusion1.2-8bit	DPM	20
stablediffusion1.2-8bit	DDIM	20	stablediffusion1.4	PNDM	20
stablediffusion1.2-8bit	DPM	10	stablediffusion1.4	PNDM	50
stablediffusion1.4	PNDM	10	stablediffusion1.4	DPM	10
stablediffusion1.4	DDIM	10	stablediffusion1.4	DPM	20
stablediffusion1.4	DDIM	20	stablediffusion1.4-6bit	DDIM	50
stablediffusion1.4	DDIM	50	stablediffusion1.4-6bit	DPM	20
stablediffusion1.4-6bit	DDIM	10	stablediffusion1.4-8bit	DDIM	50
stablediffusion1.4-6bit	DDIM	20	stablediffusion1.4-8bit	DPM	20
stablediffusion1.4-6bit	DPM	10	stablediffusion1.5	PNDM	20
stablediffusion1.4-8bit	DDIM	10	stablediffusion1.5	PNDM	50
stablediffusion1.4-8bit	DDIM	20	stablediffusion1.5	DPM	10
stablediffusion1.4-8bit	DPM	10	stablediffusion1.5	DPM	20
stablediffusion1.5	PNDM	10	stablediffusion1.5-6bit	DDIM	50
stablediffusion1.5	DDIM	10	stablediffusion1.5-6bit	DPM	20
stablediffusion1.5	DDIM	20	stablediffusion1.5-8bit	DDIM	50
stablediffusion1.5	DDIM	50	stablediffusion1.5-8bit	DPM	20
stablediffusion1.5-6bit	DDIM	10	small-stablediffusion1.5	PNDM	20
stablediffusion1.5-6bit	DDIM	20	small-stablediffusion1.5	PNDM	50
stablediffusion1.5-6bit	DPM	10	small-stablediffusion1.5	DPM	10
stablediffusion1.5-8bit	DDIM	10	small-stablediffusion1.5	DPM	20
stablediffusion1.5-8bit	DDIM	20	stablediffusion2.1	PNDM	20
stablediffusion1.5-8bit	DPM	10	stablediffusion2.1	PNDM	50
small-stablediffusion1.5	PNDM	10	stablediffusion2.1	DPM	10
small-stablediffusion1.5	DDIM	10	stablediffusion2.1	DPM	20
small-stablediffusion1.5	DDIM	20	dreamlike-photoreal	PNDM	20
small-stablediffusion1.5	DDIM	50	dreamlike-photoreal	PNDM	50
stablediffusion2.1	PNDM	10	dreamlike-photoreal	DPM	10
stablediffusion2.1	DDIM	10	dreamlike-photoreal	DPM	20
stablediffusion2.1	DDIM	20			
stablediffusion2.1	DDIM	50			
dreamlike-photoreal	PNDM	10			
dreamlike-photoreal	DDIM	10			
dreamlike-photoreal	DDIM	20			
dreamlike-photoreal	DDIM	50			

Table 9. **The detailed list of models in sub-task with "across schedules".**

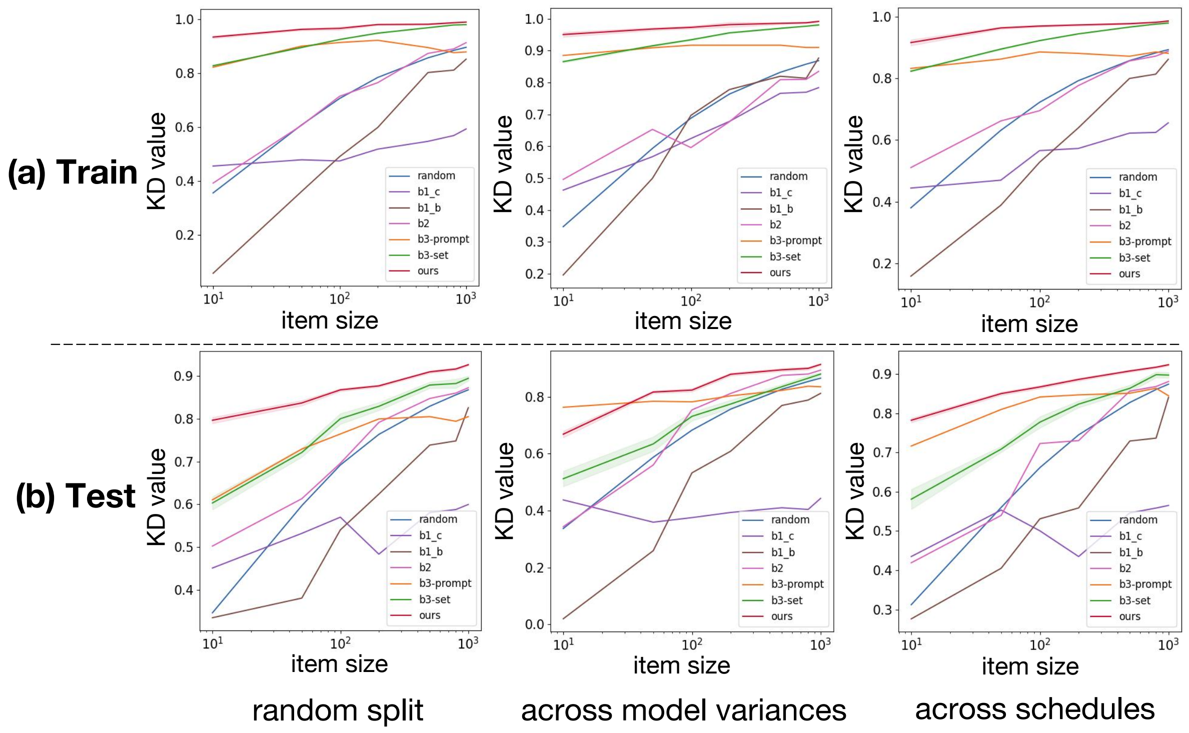


Figure 10. **Comparisons with all baselines of the Kendall's Tau for CLIP-Score on COCO dataset.**

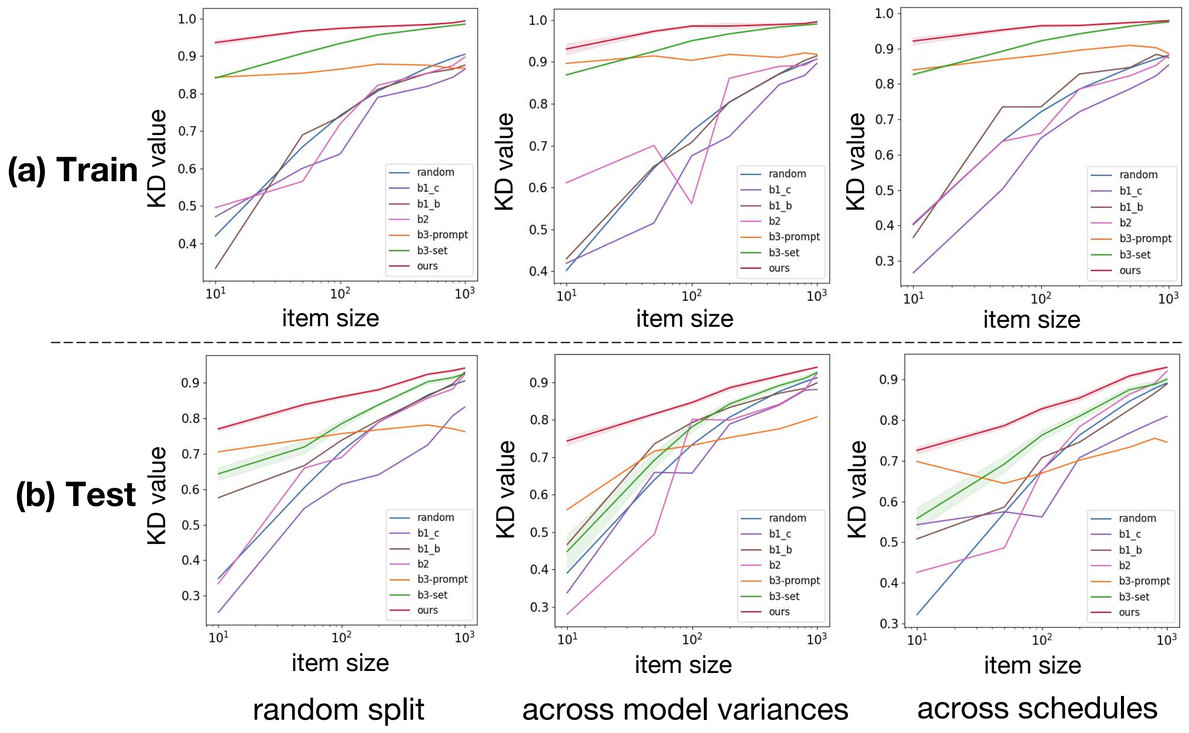


Figure 11. **Comparisons with all baselines of the Kendall's Tau for ImageReward on COCO dataset.**

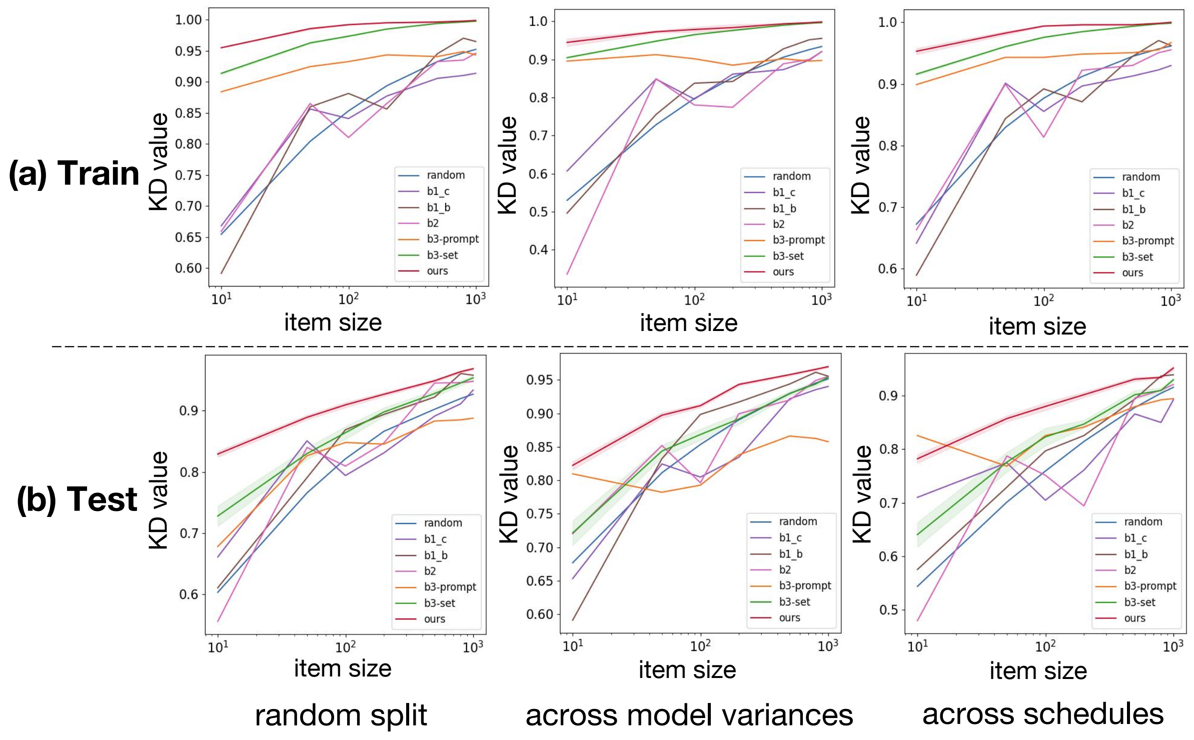


Figure 12. **Comparisons with all baselines of the Kendall's Tau for HPS on COCO dataset.**

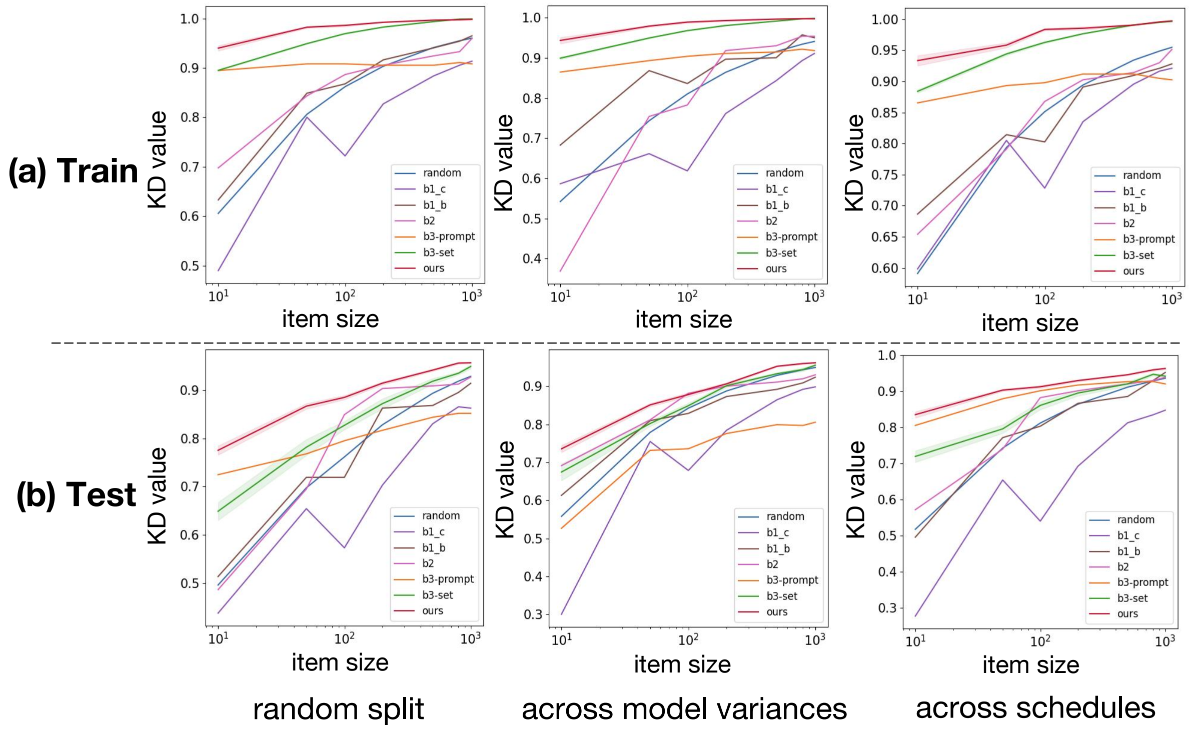


Figure 13. **Comparisons with all baselines of the Kendall's Tau for Aesthetic on COCO dataset.**

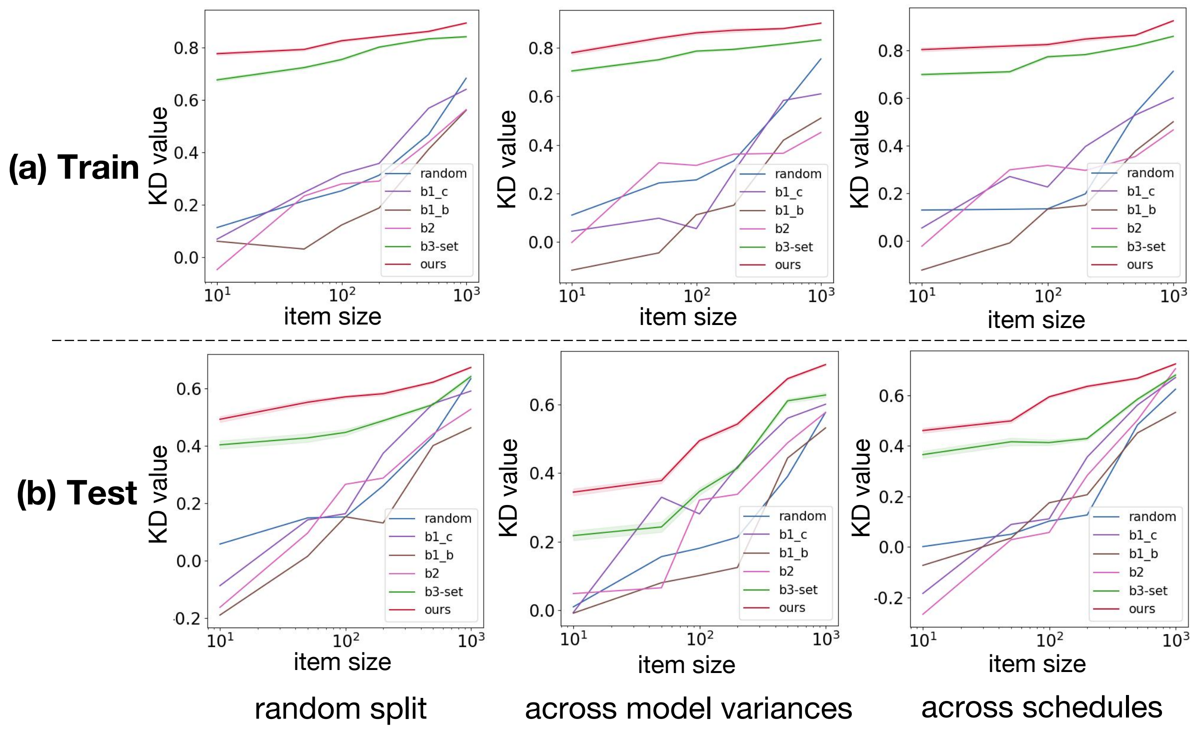


Figure 14. **Comparisons with all baselines of the Kendall's Tau for FID on COCO dataset.**

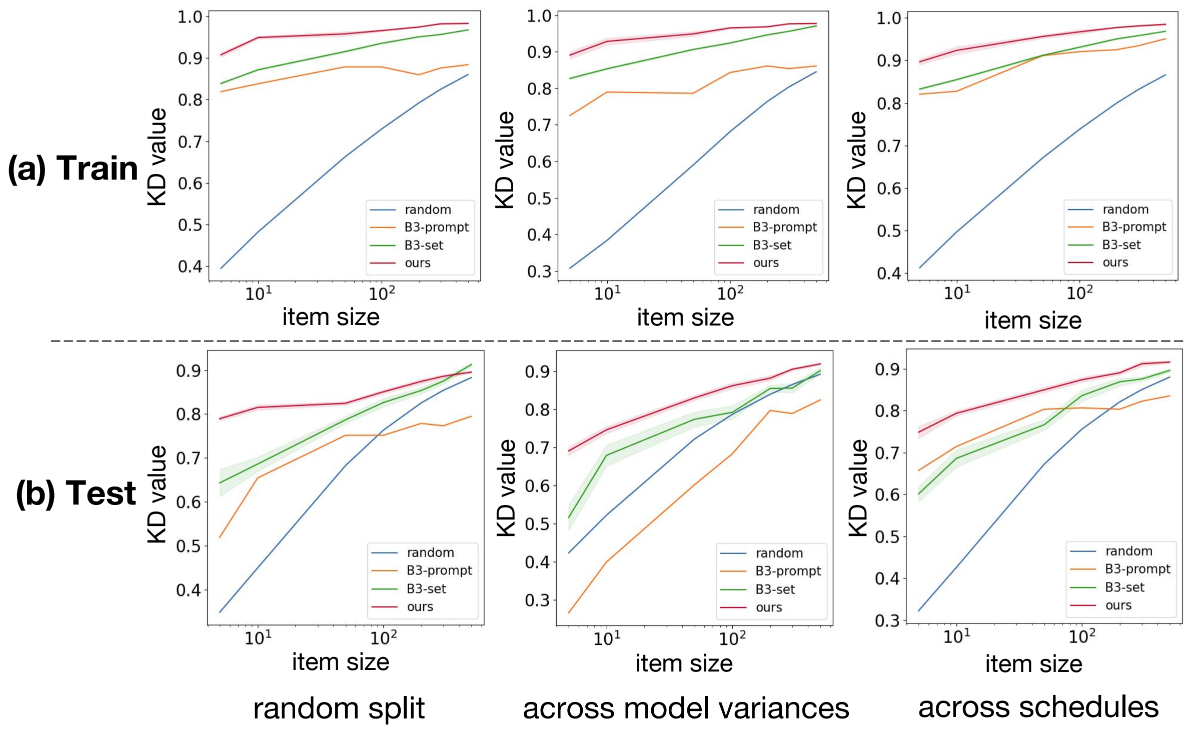


Figure 15. **Comparisons with metric space baselines of the Kendall's Tau for CLIP-Score on DiffusionDB dataset.**

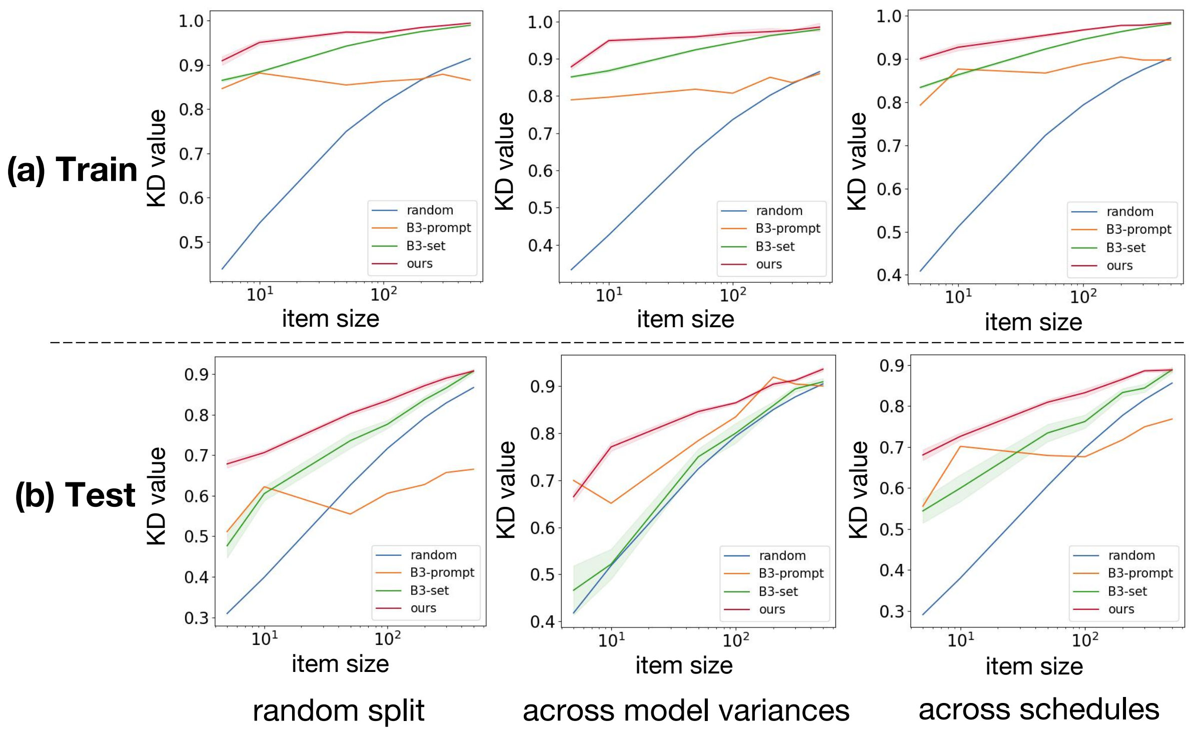


Figure 16. **Comparisons with metric space baselines of the Kendall's Tau for ImageReward on DiffusionDB dataset.**

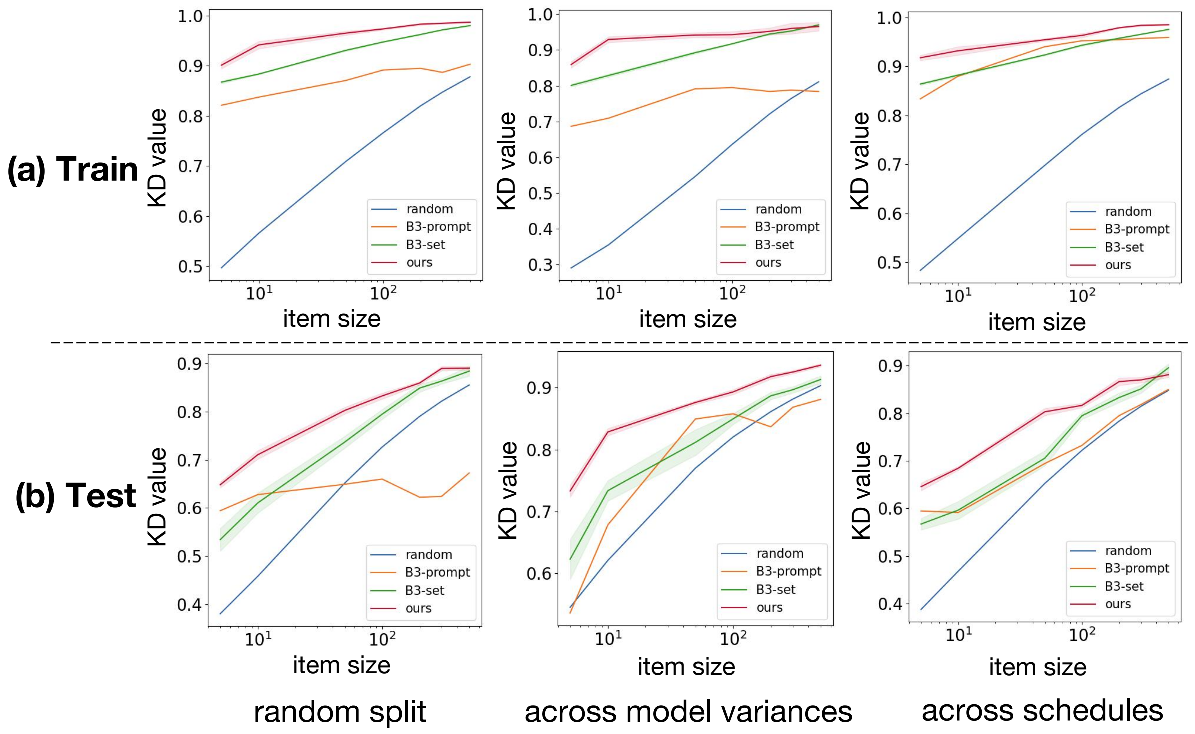


Figure 17. **Comparisons with metric space baselines of the Kendall's Tau for HPS on DiffusionDB dataset.**

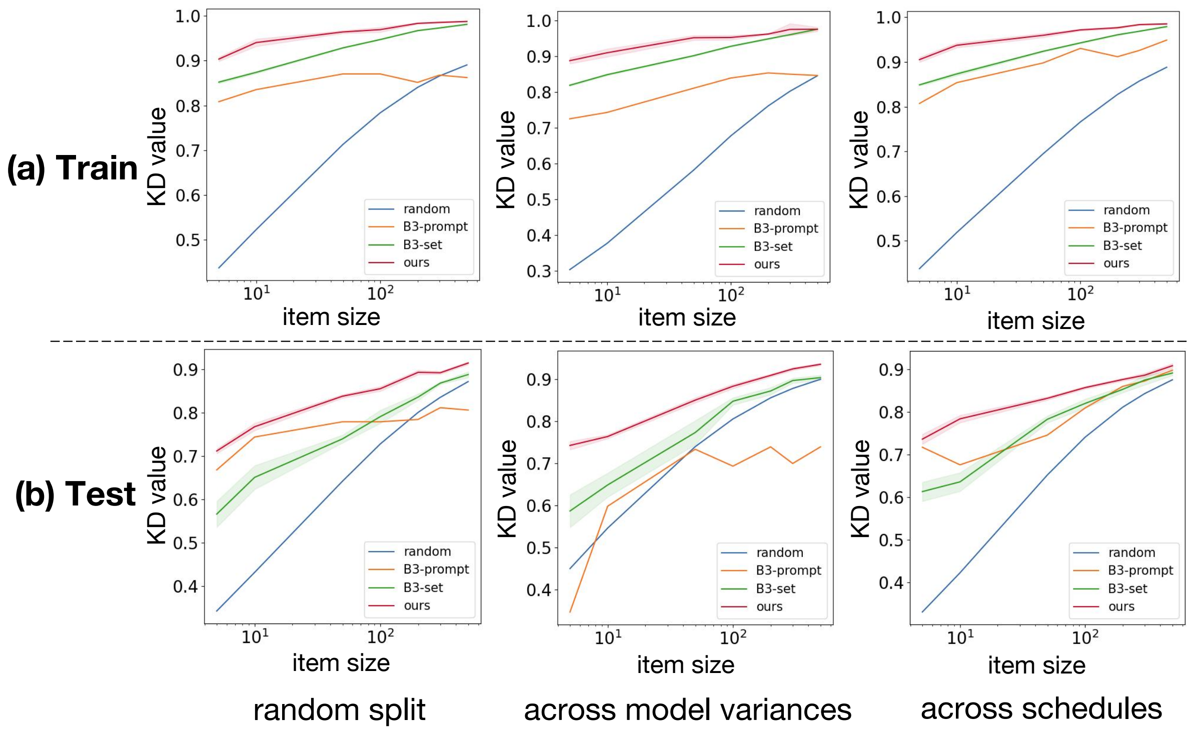


Figure 18. **Comparisons with metric space baselines of the Kendall's Tau for Aesthetic on DiffusionDB dataset.**

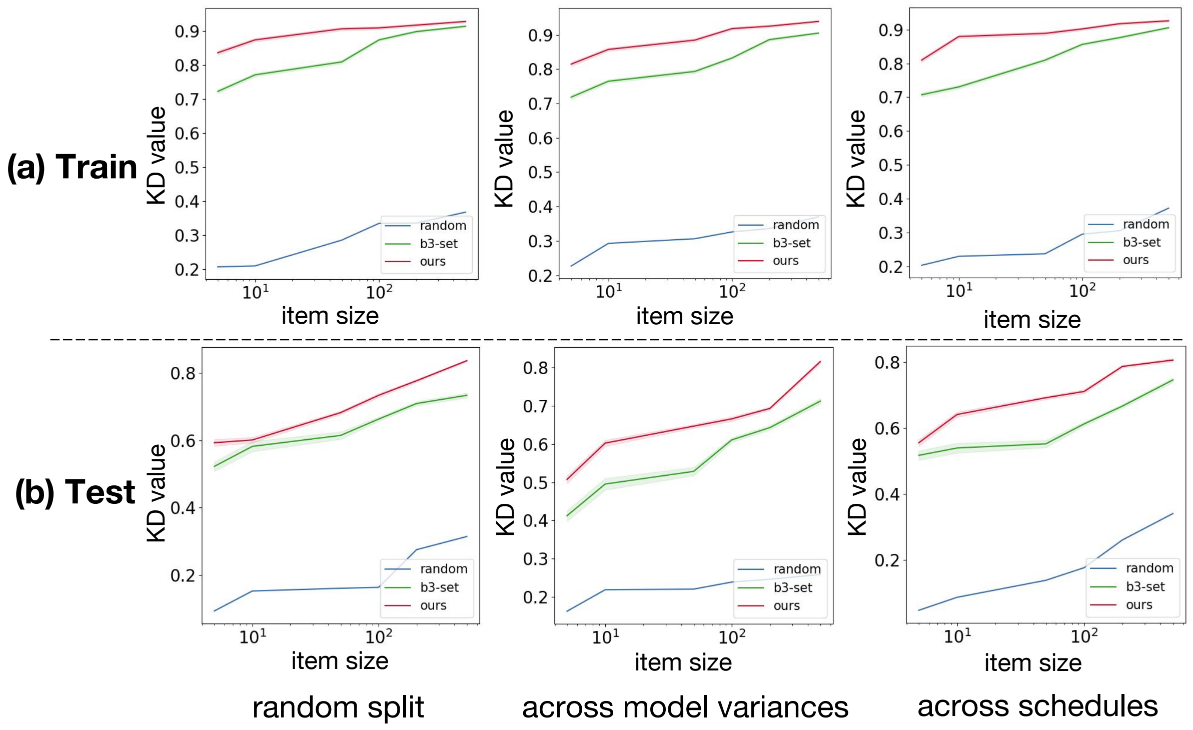


Figure 19. **Comparisons with metric space baselines of the Kendall's Tau for FID on DiffusionDB dataset.**