Corrupted but Not Broken: Understanding and Mitigating the Negative Impacts of Corrupted Data in Visual Instruction Tuning

Yunhao Gou[1,2*], Hansi Yang[2*], Zhili Liu[2,3], Kai Chen[2], Yihan Zeng[3], Lanqing Hong[3], Zhenguo Li [3], Qun Liu[3], Bo Han[4], James T. Kwok[2], Yu Zhang[1†]

[1]Southern University of Science and Technology
[2]The Hong Kong University of Science and Technology
[3]Huawei Noah's Ark Lab [4]Hong Kong Baptist University

Abstract

Visual Instruction Tuning (VIT) aims to enhance Multimodal Large Language Models (MLLMs), yet its effectiveness is often compromised by corrupted datasets with issues such as hallucinated content, incorrect responses, and poor OCR quality. Previous approaches to address these challenges have focused on refining datasets through high-quality data collection or rule-based filtering that can be costly or limited in scope. In this paper, we conduct a systematic investigation into the impact of corrupted data on MLLMs and discover that, although corrupted data degrade model performance, such adverse effects are largely reversible, and MLLMs are **corrupted but not broken**. Specifically, we find that disabling a small subset of parameters can almost fully restore performance. Moreover, corrupted MLLMs inherently possess the capability to differentiate between clean and corrupted samples, facilitating dataset cleaning without external intervention. Building on these insights, we introduce a corruption-robust training paradigm that significantly surpasses existing strategies for mitigating the effects of corrupted data.

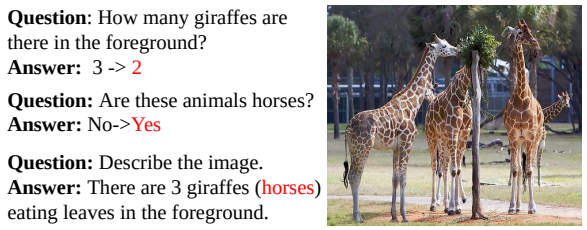


Figure 1: Examples of corrupted samples in VIT.

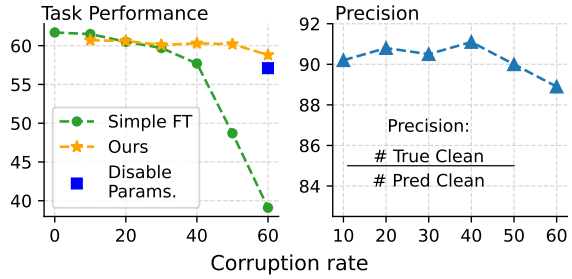


Figure 2: **Left**: Average task performance of MLLMs with various corruption ratios. Though simple fine-tuning suffers from a performance drop, disabling corruption-related parameters (1.4%) can largely restore the performance. Our method is robust to various corruption rates. **Right**: MLLM's (fine-tuned with corrupted samples) precisions of classifying clean and corrupted samples. Details are in Appendix A.

1 Introduction

Visual Instruction Tuning (VIT) (Liu et al., 2024b) has been actively explored to enhance the visual processing capabilities of Multimodal Large Language Models (MLLMs), extending beyond basic vision-language tasks to more complex domains like geometric problem-solving (Gao et al., 2023), chart interpretation (Li et al., 2024a), and self-driving (Chen et al., 2025a,b). To support these advancements, large-scale VIT datasets are either crawled from the Internet or synthesized using generative AI models. However, those datasets often contain corrupted data such as incorrect responses (Dubey et al., 2024) and hallucinated content (Wu et al., 2024a) (illustrated in Figure 1). Intuitively, such corruption in data should degrade the performance of MLLMs or cause abnormal behaviors. As such, effort has been devoted to mitigating the negative effects of data corruption in VIT data. For example, LLaMA-3 (Dubey et al., 2024) conducts post-training with verified samples. Molmo (Deitke et al., 2024) employs human annotators to curate 1M high-quality image captions. DeepSeek-VL2 (Wu et al., 2024b) refines responses using meta-information (*e.g.*, location and camera settings of an image when it is taken or image captions and OCR results from external CV tools) . However, collecting high-quality data or meta-information can be expensive. To reduce such costs, Eagle-2 (Li et al., 2025) and InternVL2.5 (Chen et al., 2024c) adopt heuristic rule-

*Equal contribution
†Corresponding author (yu.zhang.ust@gmail.com)

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 25934–25960
November 4-9, 2025

based filtering. Such heuristic rules are derived from simple intuition without a systematic analysis of negative effects of data corruption in VIT, limiting applicability to specific types of corruptions and preventing generalization to other scenarios.

Motivated by the limitations of existing works on the data corruption issue in VIT, in this paper, we propose to analyze the negative effects of data corruption in VIT from two perspectives:

1. *How does data corruption (negatively) impact the performance and behavior of MLLMs?*
2. *Do MLLMs possess any hidden capacities to overcome such negative impacts? If they do, how can we utilize them?*

Through experiments with meticulously designed corrupted VIT datasets (details in Appendix B.1-B.2), we discover the following intriguing findings:

The negative effect of data corruption is reversible. The left part of Figure 2 shows the model performance under various corruption levels. While corrupted fine-tuning data significantly degrade the MLLM's performance, simply disabling 1.4% of its parameters can largely restore its performance. This suggests that the damage due to corrupted data is restricted to only a small proportion of parameters, indicating that MLLMs may retain the ability on evaluation tasks even when fine-tuned on corrupted data (Section 4.2).

Corrupted models have underlying capability to differentiate clean and corrupted samples. As shown in the right of Figure 2, MLLMs fine-tuned on corrupted data can still distinguish clean samples from corrupted ones. We provide explanations for this ability and further propose a corruption-robust training paradigm that significantly outperforms existing corruption mitigation strategies (Section 5).

In summary, our contributions are three-fold.

- We are the first to study the impact of corrupted data in VIT, revealing its detrimental yet reversible effect on the MLLM's performance after fine-tuning.
- We further demonstrate that MLLM fine-tuned with corrupted VIT data possess an underlying ability to identify clean samples in corrupted training data, and we can utilize such ability to guide further fine-tuning to reverse the negative impacts of corrupted data.
- Empirical results across different tasks demonstrate that the underlying capabilities of the MLLM enable it to be more robust in the presence of corrupted data compared to existing approaches.

2 Related Work

2.1 Data Enhancement For MLLMs

Data corruption in VIT—such as repetitive (Chen et al., 2024c; Li et al., 2025), hallucinated responses, poor OCR quality (Wu et al., 2024b), and incorrect answers (Dubey et al., 2024)—degrades the model performance. To improve dataset quality, one can use a costly clean **oracle model** to regenerate (Chen et al., 2023c, 2024a) or filter (Fu et al., 2024b; Xiong et al., 2024) the clean samples. **Heuristic** methods detect corruption via patterns like repetition and image resolution (Chen et al., 2024c; Li et al., 2025) but fail to address hallucinations and incorrect responses. In addition, the lack of a comprehensive understanding of how corruption affects MLLMs[1] limits the development of more effective mitigation strategies. Our work fills this gap by analyzing the impact of corrupted samples and proposing a robust solution.

2.2 Learning with Noisy Labels (LNL)

Our study is related to learning with noisy labels (LNL) in machine learning, which aims to mitigate the effect of mis-labeled data when training a classification model. It can be generally categorized into three main approaches (Han et al., 2020): designing special loss functions that are robust to possibly wrong supervision (Ghosh et al., 2017; Zhang and Sabuncu, 2018; Menon et al., 2019), correcting wrong supervision with model prediction (Tanaka et al., 2018; Yi and Wu, 2019; Zhang et al., 2020), and sample selection, which identifies noisy samples from the training data and then makes them less influential in the training process (Jiang et al., 2018; Han et al., 2018; Wei et al., 2020). Among these approaches, sample selection based on the memorization effect (Arpit et al., 2017; Zhang et al., 2017), which considers samples with small loss values as clean samples (Han et al., 2018; Jiang et al., 2018; Yao et al., 2020; Yang et al., 2024), usually achieves the best performance. However, these approaches cannot effectively leverage the instruction following abilities of MLLMs, which prevents

[1]We provide extended related work on MLLM in Appendix G

them to effectively handle corrupted VIT data.

3 Preliminaries

Notations. Given an image x_v and the corresponding instruction x_q, an MLLM (parameterized by $\boldsymbol{\theta}$) predicts a response $\hat{x}_y$. For simplicity of notations, let $x_c \equiv (x_v, x_q)$. The fine-tuning process optimizes the model on a dataset $\mathcal{D} = \{(x_c, x_y)\}$ by minimizing the loss $\ell(x_y|x_c;\boldsymbol{\theta}) = -\log p(x_y|x_c;\boldsymbol{\theta})$. For convenience, we sometimes use x to denote (x_c, x_y) and simplify the loss $\ell(x_y|x_c;\boldsymbol{\theta})$ as $\ell(x;\boldsymbol{\theta})$.

Corruption. In this paper, we mainly focus on image-text alignment corruption, *i.e.*, given an image and the instruction, the response can be incorrect. This covers common corruptions in VIT such as incorrect and hallucinated responses. On the other hand, text-only corruptions, such as grammar errors and repetitions, are not the focus of this paper.

To construct image-text alignment corruptions, we replace the correct responses with incorrect ones generated from GPT-4o [2] Specifically, let $z \in \{0, 1\}$ be the correctness of a sample x_c. A dataset with corruptions is $\tilde{\mathcal{D}} = \{(x_c, \tilde{x}_y)\}$, with

$$\tilde{x}_y = \begin{cases} x_y & \text{if } z = 1 \text{ (clean)} \\ g(x_c, x_y) & \text{if } z = 0 \text{ (corrupted)} \end{cases},$$

where g denotes GPT-4o. The prompt used and examples of corrupted data are shown in Appendix B.1. We define the corruption ratio cr as the proportion of corrupted data (*i.e.*, those with $z = 0$) in $\tilde{\mathcal{D}}$.

Experimental Setup. We follow the setup of LLaVA-1.5 (Liu et al., 2023). Specifically, we begin by pre-training the vision projectors, which connect the CLIP visual encoder ViT-L/14 (Radford et al., 2021) to the LLM (*e.g.*, LLaMA-3.1-8B (Dubey et al., 2024) and Qwen-2.5 0.5B/3B/7B models (Team, 2024)), using approximately 600K image-text caption pairs. Then, we perform supervised fine-tuning (SFT) with an 100K instruction-tuning dataset, which is uniformly sampled from LLaVA-665K (Liu et al., 2023) with 665K text-only and vision-language instances. Note that for all experiments throughout this paper, we only inject corruption into the instruction-tuning dataset with images, excluding pre-training dataset and text-only instruction-tuning datasets.

[2] https://genai.ust.hk/

Following LLaVA-1.5, we evaluate the performance on 11 standard evaluation datasets. Based on the response formatting prompts, all the training and evaluation datasets are divided into the following groups: (i) VQA (visual question answering); (ii) MC-VQA (multiple-choice VQA); and (iii) Conversation. Note that these groups only differ in response formatting, *i.e.*, they share the same underlying vision-language content. More details on the datasets and performance metrics are in Appendices B.2 and B.3, respectively.

4 Effects of Data Corruption in VIT

To analyze how data corruption may impact the MLLM's performance on downstream tasks, we first consider the simplest way of introducing data corruption which uniformly draws clean samples from all datasets and replaces them with the corrupted ones. Denote the ratio of corrupted samples in the whole dataset as cr (corruption ratio), which is varied from 0% to 60%. To see whether the corrupted samples contribute negatively, we further construct a reference dataset with them removed. Figure 3 shows the performance of MLLM fine-tuned under different corruption ratios on various benchmarks (results on more datasets are in Appendix C). For most tasks, corrupting the data results in worse performance than simply removing them. This degradation worsens with increasing cr, indicating the negative effect of corrupted data.

While such negative effect well matches our intuition, our further investigation will reveal that such negative effect is *restricted* and largely *reversible*. In Section 4.1, we demonstrate that data corruption only on specific tasks does not yield negative effects on other tasks. In Section 4.2, we further demonstrate that such effect can be largely reverted by removing affected parameters that only take a small proportion of the whole MLLM and that MLLMs fine-tuned with corrupted data retain capabilities on the evaluated tasks.

4.1 Negative Effects of Data Corruption is Restricted to Tasks with Corrupted Data

Besides uniformly selecting clean samples from all datasets and replacing them with corrupted ones, we further consider another type of corruption that only selectively corrupts clean samples for specific tasks: (i) `no_vqa`: corruption is injected into all datasets except the VQA datasets; (ii) `no_mc-vqa`, in which the corruption is injected into all datasets

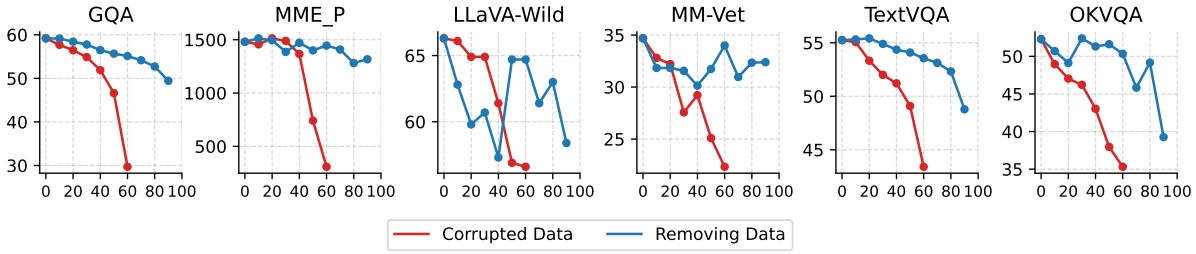


Figure 3: **Performance (y-axis) of LLaVA-1.5 (LLaMA-3.1-8B) under different corruption ratios (x-axis).**

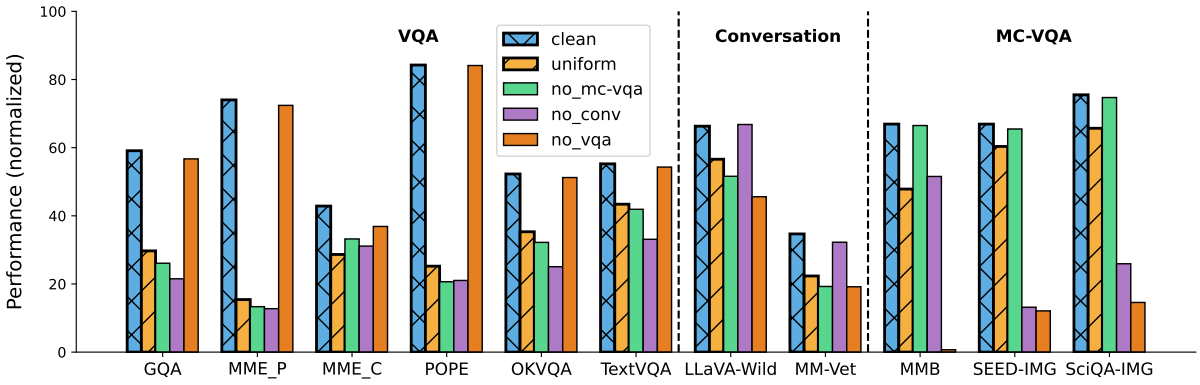


Figure 4: **Effects of corruption on LLaVA-1.5 (LLaMA-3.1-8B).** The evaluation datasets are shown in 3 groups: VQA, Conversation and MC-VQA. The corruption ratio here is 60%.

except the multiple-choice VQA datasets; and (iii) `no_conv`, in which the corruption is injected into all datasets except the conversation datasets.

Figure 4 compares the performance of MLLMs fine-tuned on datasets with different types of corruption. While Figure 3 demonstrates that data corruption leads to performance degradation, *such effect is limited to tasks that contain corrupted training data and does not extend to other tasks without corruption.* For example, on the VQA task, the MLLMs fine-tuned on `no_vqa` perform comparably to those fine-tuned on `clean` and considerably better than those fine-tuned on `uniform` across different datasets. This observation remains valid for models fine-tuned on the `no_mc-vqa` and `no_conv` datasets, when evaluated on the multiple-choice VQA and conversation tasks, respectively. In other words, even if the majority of training data are corrupted ($cr = 60\%$), the model can still maintain its performance on specific tasks as long as the corresponding training data are not corrupted.

Implications. As detailed in Appendix B.2, the VQA, MC-VQA and Conversations tasks in this section differ only in their response format prompts rather than the underlying vision-language (or multi-modal) knowledge. Therefore, the restricted

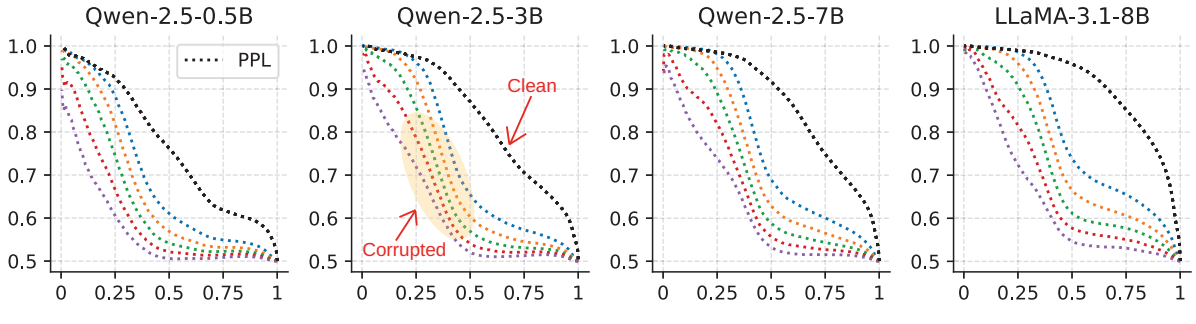
Model	Avg. Score	% Disabled	(p, q)
`Clean`	61.7	-	-
`Clean (40K)`	59.3	-	-
	39.1	0	-
`Corrupted`	51.4	0.84	(12, 10)
($cr = 60\%$, `100K`)	55.2	1.16	(17, 15)
	57.1	1.39	(22, 20)

Table 1: **Performance of LLaVA-1.5 (LLaMA-3.1-8B) with corruption-related weights disabled**.

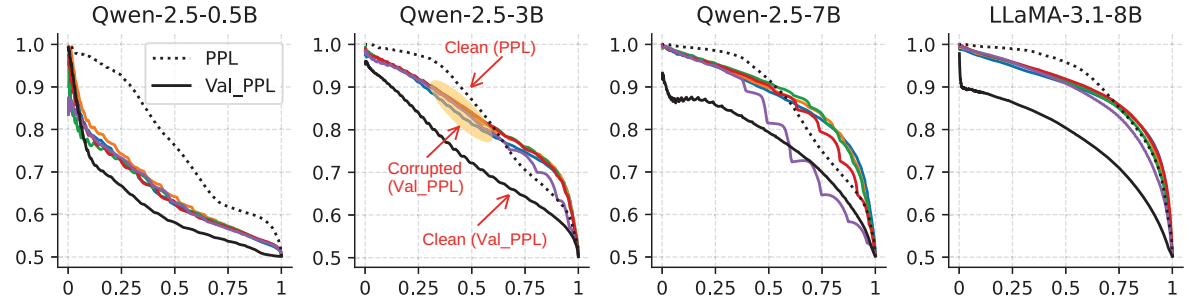
effect of corrupted VIT data revealed in this section indicates that even though the MLLM is corrupted, it still retains recoverable capabilities on the evaluated tasks. We shall discuss detailed recovery strategies in the next section.

4.2 Negative Effects of Data Corruption is Reversible

To recover the performance of the MLLM fine-tuned on corrupted data, we remove model parameters affected by the corrupted data. To achieve this goal, we first try to identify weights in the MLLM that are particularly responsible for generating the corrupted responses. Specifically, following Wei et al. (2024), we select weights with top-q% influence scores on the **corrupted** dataset but remove



(a) **PPL**



(b) **PPL and Val_PPL**

Figure 5: **Precision-recall curves of MLLM's predictions on the correctness of 100K samples** ($cr = 50\%$)**.** x-axis: recall; y-axis: precision. Solid and dotted line denote predictions based on `Val_PPL` and `PPL`, respectively. Color represents the corruption ratio of the training dataset: ●0%, ●10%, ●20% ●30%, ●40%, ●50%.

those that overlap with weights with top-$p\%$ influence on a **clean** dataset. This ensures that only weights contributing specifically to corrupted samples are considered. We use the SNIP score (Lee et al., 2018) to compute the influence, and consider an MLLM fine-tuned on 100K VIT data samples with corruption ratio $cr = 60\%$. With different choices of (p, q), we remove weights that are only related to the corrupted data and report the performance after removing these weights. For comparison, we include results from models fine-tuned with 100K and 40K clean VIT data samples (denoted `Clean` and `Clean(40K)`, respectively). More details on the identification and removal of corruption-related weights are in Appendix D.

Table 1 shows the performance on the evaluation tasks after removing weights related to the corrupted data. By removing fewer than **1.4%** of the parameters, the corrupted model can restore its performance from 39.1 to 57.1, which is already close to that (*i.e.*, 59.3) of the model fine-tuned with 40K clean data (all clean data under a cr of 60%).

Implications. Notably, removing weights relevant to the corrupted data aims to "delete" effects of corrupted data rather than "override" with new knowledge. This confirms that the corrupted MLLM does retain capabilities on the evaluated tasks.

5 Underlying Capabilities of MLLMs Fine-tuned with Corrupted Data

Motivated by the restrictive and reversible nature of the negative impact of corrupted data as analyzed in Sections 4.1 and 4.2, we conjecture that the MLLMs fine-tuned with corrupted data may possess the underlying capacity to identify clean training samples and introduce a simple method called **self-validation** in Section 5.1.

By further fine-tuning the corrupted MLLMs with samples selected by self-validation, we demonstrate that the corrupted MLLM can recover from the negative impacts of corruption without any external annotations in Section 5.2.

5.1 Corrupted MLLM Can Detect Clean Samples from Corrupted Dataset

To identify clean and noisy samples from a corrupted VIT dataset, a straight-forward idea following existing works on learning with label noise is to consider the sample loss (Jiang et al., 2018; Han et al., 2018): a larger sample loss may indicate that the corresponding sample is likely to contain

wrong response. This can be implemented using the perplexity score (Marion et al., 2023), as it is proportional to the MLLM's training loss (details in Appendix E.1). We term this score PPL.

Figure 5a compares the recalls and precisions of using PPL from MLLM fine-tuned with varying corruption levels to identify clean and noisy samples (details in Appendix E.2). As can be seen, MLLMs fine-tuned with clean data (black dotted line) can accurately identify clean samples by only keeping the small-loss samples. For example, Qwen-2.5-3B achieves precision over 0.85 at a recall of 0.5. On the contrary, MLLMs fine-tuned with corrupted data (non-black dotted lines) achieve worse performance with increasing corruption levels. Specifically, all their precisions drop below 0.65 at a recall of 0.5. *In other words, existing loss-based sample selection approaches are not robust to data corruption and cannot be simply extended to VIT.*

5.1.1 Utilize Self-Validation to Activate Corrupted MLLM's Capability

Given that the negative effect of corrupted data is restricted to specific response formats (Section 4.1) and that the corrupted MLLMs still retain capabilities on the evaluated tasks (Section 4.2), we propose to instruct the corrupted MLLM with a different response format that is unseen during training. Specifically, using the following template, we directly prompt the MLLM to predict whether a sample is corrupted:

Template to Obtain Val_PPL

<image>Query: {instruction text}
Response: {response text}
Is the response correct? Answer yes or no:

The perplexity of the predicted word "No" is then used as the score. To distinguish from the perplexity scores of responses, we refer the perplexity of word "No" in self-validation as Val_PPL. Contrary to PPL, samples with smaller Val_PPL scores are considered more likely to be corrupted.

Self-Validation is robust against corrupted data. Figure 5b compares the recalls and precisions of using the Val_PPL scores (solid curves) from MLLM fine-tuned with varying corruption levels to identify clean and corrupted samples. For reference, we also include the corresponding precision-recall curves of clean MLLMs previously shown in Figure 5a (black dotted curve). Remarkably, for MLLMs with 3B parameters and above, models fine-tuned with corrupted data (non-black solid curves) obtain similar precision-recall curves compared to that of the clean model using PPL. Furthermore, even at increasing corruption levels, these curves are consistent and close to each other (except for Qwen-2.5-7B at $cr = 50\%$). This demonstrates that the corrupted MLLMs can effectively distinguish clean and corrupted samples using Val_PPL.

However, note that the advantage of using Val_PPL is less significant for the small-sized LLM (Qwen-2.5-0.5B). This suggests that self-validation is an emergent ability which is not present in smaller models but only present in larger models (Wei et al., 2022; Berti et al., 2025).

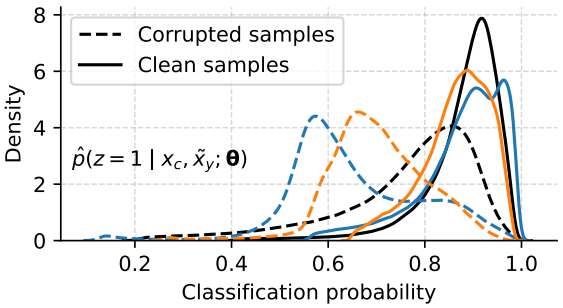


Figure 6: Distribution of classification probability $\hat{p}(z = 1|x_c, \tilde{x}_y; \boldsymbol{\theta})$. Color represents corruption ratios of datasets the model is trained on: ●0%,●10%, ●20%.

5.1.2 Why do Corrupted Data Improve the Performance of Self-Validation?

An intriguing observation from Figure 5b is that with self-validation, the corrupted model can identify corrupted samples even better than the clean model, as the solid black curves are always below the solid non-black curves. In this section, we provide an empirical analysis for this phenomenon.

Probabilistic modeling of self-validation. Recall that the Val_PPL template asks the MLLM to predict correctness of a sample. Essentially, the MLLM estimates $p(z = 1|x_c, \tilde{x}_y)$, the (ground-truth) probability that response $\tilde{x}_y$ is correct, with its output probability $\hat{p}(z = 1|x_c, \tilde{x}_y; \boldsymbol{\theta})$. Figure 6 shows the distributions of $\hat{p}$ on the clean samples and corrupted samples from a corrupted VIT dataset that is also used to finetune the MLLM. We can see that corruption in the fine-tuning data has little effect on the output probability distribution for clean samples (solid curves). This re-

sults in robustness of Val_PPL at various corruption ratios and can be explained by the restrictive nature of corrupted data on MLLMs as the corrupted dataset does not contain self-validation instructions. In contrast, significant distribution shift is observed on that of the corrupted samples (dashed curves), which is the cause of improved performance of Val_PPL when fine-tuned with corrupted data. However, this cannot be explained by the conclusions drawn so far.

Improved understanding of corrupted data leads to distribution shift of $\hat{p}(z = 1|x_c, \tilde{x}_y; \boldsymbol{\theta})$ on the corrupted samples (see Appendix E.3 for a detailed analysis). Although there are no explicit annotations indicating which samples are corrupted, the MLLM is still, to some extent, aware of the corrupted data during fine-tuning. This awareness stems from its prior pre-training on image-text pairs, which enables the model to learn correct image-text alignment. However, the PPL score cannot benefit from the improved understanding of corrupted data because it shares the same response format as the corrupted data and is thereby affected.

5.2 Corruption-Robust Fine-tuning Strategies

To tackle the adverse effects of corrupted data during fine-tuning, we propose a corruption-robust fine-tuning strategy that integrates self-validation and further fine-tuning.

Method. Specifically, for a training dataset that might potentially contain corrupted samples, we first conduct simple fine-tuning for the MLLM on this data. Then, the MLLM conducts self-validation to select clean samples by thresholding the Val_PPL score. Finally, we further fine-tune the MLLM with the selected samples.

Baselines and Implementation Details. We conduct experiments on the 11 benchmark datasets introduced in Section 3 with the corruption ratio $cr = 50\%$, by using LLaVA-1.5 built on LLama-3.1-8B and Qwen-2.5 series models. When thresholding, we choose samples with the highest 30% Val_PPL scores for further fine-tuning (*i.e.*, 30K) by default. In addition to using samples selected from self-validation with the Val_PPL score, we also consider using the following scores to select samples: (i) the PPL score proposed in Section 5.1; (ii) EL2N and GradNorm, which measure the ℓ_2-norm of the output error vector and the gradient, respectively (Paul et al., 2021); and (iii) Entropy (Coleman et al., 2019), which reflects uncertainty in the output probabilities. Detailed formulae of these scores are in Appendix F.1.

To better benchmark the performance of using self-validation for performance recovery, we also introduce the following baselines from traditional LNL (learning with noisy labels) methods:

1. Noise-robust loss functions. By default, the MLLM is trained with the Cross-Entropy (CE) loss, denoted None (CE). We consider two noise-robust loss functions from Menon et al. (2019): (i) Generalized Cross-Entropy (GCE) loss (Zhang and Sabuncu, 2018), that combines the CE loss and mean absolute error (MAE), which is more robust to label noise (Ghosh et al., 2017), and (ii) Phuber Cross-Entropy loss (Menon et al., 2019) (Phuber CE), which incorporates gradient clipping into CE. Their detailed formulae are in Appendix F.2.

2. Online sample selection methods, which focus on selecting clean samples during training. MentorNet (Jiang et al., 2018) identifies small-loss samples as clean. Co-teaching (Han et al., 2018) trains two networks and exchanges small-loss samples between them to avoid error accumulation. JoCoR (Wei et al., 2020) enforces agreement between networks to prevent biased selection. Their details are in Appendix F.3.

Results. Table 2 compares the performance of various corruption-robust strategies. By using the Val_PPL score to select clean VIT data samples for further fine-tuning, we can significantly restore the performance of a corrupted model, improving it from 49.13 to 60.2 on average (the clean model achieves 61.65). We also outperform all existing baseline methods on average and achieves the best results on 8 out of 11 evaluation tasks, which further justify the high quality of VIT data samples selected by the Val_PPL score.

Analysis. To further assess the effectiveness of self-validation, we conduct further fine-tuning on LLaVA-1.5 fine-tuned with 20% and 50% corruption data, which cover low and high corruption in practical scenarios,[3] using "clean data" from various sources. (i) GT: Clean data with responses directly from the clean dataset. This can be regarded as the "best" clean data possible); (ii)

[3] Results for other corruption levels are in Figure 19 of the appendix.

Methods	**Avg.**	GQA	MME_P	MME_C	POPE	LLaVA Wild	MM-Vet	MMB	SEED IMG	SciQA IMG	Text VQA	OKVQA
Clean	61.65	59.18	1480.36	342.86	84.25	66.30	34.68	66.92	66.91	75.51	55.25	52.28
None (CE)	49.13	41.87	668.17	253.21	62.90	57.30	23.47	63.83	63.26	74.02	50.01	38.67
Noise-robust loss functions												
GCE	50.95	40.67	751.27	240.00	69.37	62.00	23.85	65.81	64.84	**74.81**	50.05	41.51
Phuber CE	47.28	37.24	595.69	258.21	46.77	59.90	26.93	61.51	63.49	73.82	48.24	40.16
Sample selection												
MentorNet	46.21	40.07	746.12	261.43	69.67	60.20	27.84	46.13	49.39	62.82	47.57	34.69
Co-teaching	47.97	39.95	583.35	253.93	66.38	57.60	29.63	55.58	60.91	72.14	48.83	35.68
JoCoR	47.00	39.23	571.96	245.36	59.09	58.40	27.80	55.33	60.28	72.19	48.69	36.76
Further fine-tuning												
EL2N	47.07	49.34	1357.43	269.64	**83.97**	58.20	27.34	12.97	43.49	51.96	50.62	38.27
GradNorm	55.77	50.06	1342.40	**318.93**	76.86	66.50	27.25	59.79	64.01	73.43	49.07	39.48
Entropy	48.60	43.76	1118.39	261.79	73.53	60.30	27.57	42.10	52.68	63.01	48.12	34.94
PPL	54.69	47.54	1222.56	279.64	82.65	60.50	25.69	64.86	62.83	73.38	49.04	39.02
Val_PPL	**60.17**	**56.65**	**1510.48**	297.50	82.18	**69.30**	**31.51**	**67.18**	**65.48**	74.62	**53.48**	**48.76**

Table 2: **Comparisons of different corruption-robust strategies at a corruption ratio of 50% on LLaVA-1.5 (LLaMA-3.1-8B)**, where **Avg.** refers to the average results on 11 benchmarks (normalized to 0-100). Best results are in **bold**, and the second best are underlined. Results for Qwen-2.5 series are in Tables 6, 7, and 8 in the appendix.

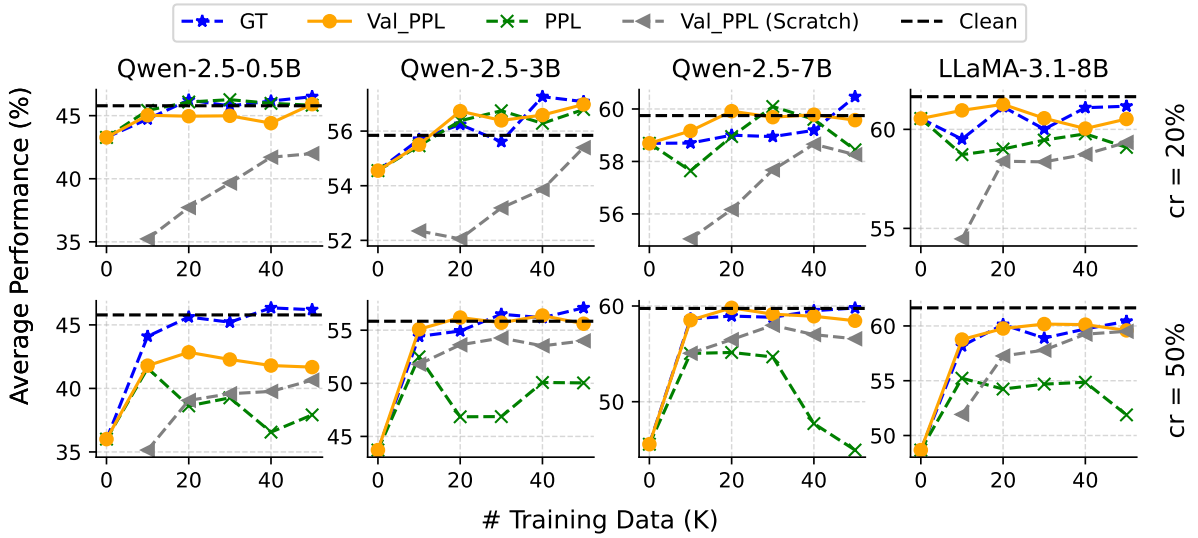


Figure 7: **Average task performance of models further fine-tuned (or fine-tuned from scratch) on data from different sources**. The corruption ratios are 20% (low) and 50% (high), results for other corruption levels are in Figure 19 in the appendix.

PPL (we choose it rather than GradNorm or Entropy because it is widely adopted in LNL and we studied it in Section 5.1); and (iii) the proposed Val_PPL. In addition, we also experiment with (iv) Val_PPL(Scratch), which fine-tunes the model from scratch using the samples selected by Val_PPL rather than after it is fine-tuned on corrupted data.

Figure 7 shows the average task performance of these models further fine-tuned (or fine-tuned from scratch) using "clean data" from various sources with different sizes. As can be seen, with GT, most MLLMs can largely restore the performance using 20K samples, which indicates that given clean data, further fine-tuning can be a fast and cost-effective approach for performance recovery on corrupted MLLMs. Notably, both at low (20%) and high (50%) corruption ratios, Val_PPL performs as well as GT for the 3B and larger MLLMs, rapidly approaching the performance of the clean model. However, for Qwen-2.5-0.5B, only GT can restore the model performance. This is consistent with our observation in Figure 5b that self-validation is an emergent (Wei et al., 2022; Berti et al., 2025) ability for larger LLMs. On the contrary, though PPL can achieve results comparable to Val_PPL at low

corruption, it is less effective at high corruption, demonstrating the robustness of self-validation at various corruption ratios. Further, we find that `Val_PPL(Scratch)` is consistently outperformed by its further fine-tuning counterparts, demonstrating the sample-efficiency of further finetuning.

6 Experiments with Other Sources of Corrupted Data

To validate the generalization of our findings, we also experiment with other sources of corrupted data in addition to GPT-4o. Specifically, we consider the following two strategies.

Gemini-2.5-Flash. We use another proprietary model Gemini-2.5-Flash (Comanici et al., 2025) to generate the corrupted data in the same way as GPT-4o. We follow previous experiment setting in Sec. 3 and use Qwen2.5-3B for experiments. We report the performance when the model is fine-tuned on clean and corrupted data ($cr = 50\%$) and further fine-tuned with selected (PPL, Val_PPL) clean samples in Table 10. Similarly, the results for disabling corruption-related and selecting clean samples are provided in Tables 11 and 12. As can be seen, the observations are consistent with that when using GPT-4o as corruption source.

Explicitly training a weak MLLM. Instead of deliberately prompting the MLLM to generate incorrect answers, we explicitly train a weak MLLM (using LLaMA-3.1-8B as the LLM) with accurate but very limited visual instruction tuning data (*e.g.*, 20K). We then run inference with this model on the 100K training data. All generated data from this model is regarded as corrupted data, which are then mixed with the clean data for experiments. This design aims to simulate a practical case where the practitioners are trying to construct synthetic data from existing MLLMs, which might make mistakes in the generated data.

We report the performance when the model (Qwen2.5-7B as the LLM) is fine-tuned on clean, corrupted data ($cr = 50$) and further fine-tuned with selected (PPL, Val_PPL) clean samples in Table 13. As shown, the corrupted data generated by a weak model negatively affects the performance of the model but can be mitigated via further fine-tuning with clean samples selected by Val_PPL. This indicates that the findings in this paper is generalizable to various sources of corrupted data.

7 Experiments with Qwen2-VL

We also validate our findings with recent Qwen2-VL (Wang et al., 2024). Note that although the more recent Qwen2.5-VL serie (Bai et al., 2025) is released, only the Qwen2-VL serie releases the weights before SFT. As we mainly investigate the effects of corrupted data in SFT, we choose Qwen2-VL-2B (the weights before SFT) for experiments.

We report the performance when the model is fine-tuned on clean, corrupted data ($cr = 50$) and further fine-tuned with selected (PPL, Val_PPL) clean samples in Table 14. Similarly, the results for selecting clean samples are provided in Table 15. As can be seen, the observations are consistent with that when using LLaVA-1.5 as the MLLM architecture.

8 Conclusion

In summary, we show that the negative impact of corrupted data on MLLMs is largely reversible and that these models can inherently identify corrupted samples. Leveraging these insights, we introduce a corruption-robust training method that outperforms existing strategies, improving the robustness of Visual Instruction Tuning.

Acknowledgements

This work is supported by NSFC grant under grant no. 62136005. This work has been made possible by a Research Impact Fund project (RIF R6003-21) and General Research Fund projects (GRF 16203224 and 16202523, and CRF C7004-22G-1) funded by the Research Grants Council (RGC) of the Hong Kong Government.

Limitations

One limitation of this paper is that we did not study MLLMs with larger LLMs (*e.g.*, 70B and 400B) due to limited computational resources.

Ethic Statement

There is no ethical problem in our study.

References

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *ICML*, pages 233–242.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. Preprint arXiv:2308.12966.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2.5-VL. Preprint arXiv:2502.13923.

Leonardo Berti, Flavio Giorgi, and Gjergji Kasneci. 2025. Emergent abilities in large language models: A survey. Preprint arXiv:2503.05788.

Guiming Hardy Chen, Shunian Chen, Ruifei Zhang, Junying Chen, Xiangbo Wu, Zhiyi Zhang, Zhihong Chen, Jianquan Li, Xiang Wan, and Benyou Wang. 2024a. ALLaVA: Harnessing gpt4v-synthesized data for a lite vision-language model. Preprint arXiv:2402.11684.

Kai Chen, Ruiyuan Gao, Lanqing Hong, Hang Xu, Xu Jia, Holger Caesar, Dengxin Dai, Bingbing Liu, Dzmitry Tsishkou, Songcen Xu, and 1 others. 2025a. Eccv 2024 w-coda: 1st workshop on multimodal perception and comprehension of corner cases in autonomous driving. *arXiv preprint arXiv:2507.01735*.

Kai Chen, Yunhao Gou, Runhui Huang, Zhili Liu, Daxin Tan, Jing Xu, Chunwei Wang, Yi Zhu, Yihan Zeng, Kuo Yang, and 1 others. 2024b. EMOVA: Empowering language models to see, hear and speak with vivid emotions. Preprint arXiv:2409.18042.

Kai Chen, Lanqing Hong, Hang Xu, Zhenguo Li, and Dit-Yan Yeung. 2021. MultiSiam: Self-supervised multi-instance siamese representation learning for autonomous driving. In *ICCV*.

Kai Chen, Yanze Li, Wenhua Zhang, Yanxin Liu, Pengxiang Li, Ruiyuan Gao, Lanqing Hong, Meng Tian, Xinhai Zhao, Zhenguo Li, and 1 others. 2025b. Automated evaluation of large vision-language models on self-driving corner cases. In *WACV*.

Kai Chen, Zhili Liu, Lanqing Hong, Hang Xu, Zhenguo Li, and Dit-Yan Yeung. 2023a. Mixed autoencoder for self-supervised visual representation learning. In *CVPR*.

Kai Chen, Chunwei Wang, Kuo Yang, Jianhua Han, Lanqing Hong, Fei Mi, Hang Xu, Zhengying Liu, Wenyong Huang, Zhenguo Li, Dit-Yan Yeung, Lifeng Shang, Xin Jiang, and Qun Liu. 2023b. Gaining wisdom from setbacks: Aligning large language models via mistake analysis. Preprint arXiv:2310.10477.

Lin Chen, Jisong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. 2023c. ShareGPT4V: Improving large multimodal models with better captions. *arXiv preprint arXiv:2311.12793*.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Linxin Gu, Xuehui Wang, Qingyun Li, Yimin Ren, Zixuan Chen, Jiapeng Luo, Jiahao Wang, Tan Jiang, Bo Wang, and 23 others. 2024c. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. Technical report, arXiv preprint arXiv:2412.05271.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2019. Selection via proxy: Efficient data selection for deep learning. Preprint arXiv:1906.11829.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. Preprint arXiv:2507.06261.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, and 1 others. 2024. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. Preprint arXiv:2409.17146.

Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Songyang Zhang, Haodong Duan, Wenwei Zhang, Yining Li, and 1 others. 2024. InternLM-XComposer2-4KHD: A pioneering large vision-language model handling resolutions from 336 pixels to 4K HD. Preprint arXiv:2404.06512.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The Llama 3 herd of models. Preprint arXiv:2407.21783.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, and Rongrong Ji. 2024a. MME: A comprehensive evaluation benchmark for multimodal large language models. Preprint arXiv:2306.13394.

Deqing Fu, Tong Xiao, Rui Wang, Wang Zhu, Pengchuan Zhang, Guan Pang, Robin Jia, and Lawrence Chen. 2024b. TLDR: Token-level detective reward model for large vision language models. Preprint arXiv:2410.04734.

Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing Hong, Jianhua Han, Hang Xu, Zhenguo Li, and 1 others. 2023. G-LLaVA: Solving geometric problem with multi-modal large language model. Preprint arXiv:2312.11370.

Aritra Ghosh, Himanshu Kumar, and P Shanti Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *AAAI*.

Yunhao Gou, Kai Chen, Zhili Liu, Lanqing Hong, Xin Jin, Zhenguo Li, James T Kwok, and Yu Zhang. 2025. Perceptual decoupling for scalable multi-modal reasoning via reward-optimized captioning. *arXiv preprint arXiv:2506.04559*.

Yunhao Gou, Kai Chen, Zhili Liu, Lanqing Hong, Hang Xu, Zhenguo Li, Dit-Yan Yeung, James T Kwok, and Yu Zhang. 2024. Eyes Closed, Safety On: Protecting multimodal llms via image-to-text transformation. *arXiv preprint arXiv:2403.09572*.

Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T Kwok, and Yu Zhang. 2023. Mixture of cluster-conditional LoRA experts for vision-language instruction tuning. *arXiv preprint arXiv:2312.12379*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *CVPR*.

Bo Han, Quanming Yao, Tongliang Liu, Gang Niu, Ivor W Tsang, James T. Kwok, and Masashi Sugiyama. 2020. A survey of label-noise representation learning: Past, present and future. *arXiv preprint arXiv:2011.04406*.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NIPS*, pages 8527–8537.

Runhui Huang, Xinpeng Ding, Chunwei Wang, Jianhua Han, Yulong Liu, Hengshuang Zhao, Hang Xu, Lu Hou, Wei Zhang, and Xiaodan Liang. 2024. HiRes-LLaVA: Restoring fragmentation input in high-resolution large vision-language models. *arXiv preprint arXiv:2407.08706*.

Drew A Hudson and Christopher D Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*.

Lu Jiang, Z. Zhou, T. Leung, J. Li, and Fei-Fei Li. 2018. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, pages 2309–2318.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. ReferItGame: Referring to objects in photographs of natural scenes. In *EMNLP*, pages 787–798.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, and 1 others. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. SNIP: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. 2023a. SEED-Bench: Benchmarking multimodal LLMs with generative comprehension. *arXiv preprint arXiv:2307.16125*.

Lei Li, Yuqi Wang, Runxin Xu, Peiyi Wang, Xiachong Feng, Lingpeng Kong, and Qi Liu. 2024a. Multimodal arxiv: A dataset for improving scientific comprehension of large vision-language models. *arXiv preprint arXiv:2403.00231*.

Yanwei Li, Yuechen Zhang, Chengyao Wang, Zhisheng Zhong, Yixin Chen, Ruihang Chu, Shaoteng Liu, and Jiaya Jia. 2024b. Mini-Gemini: Mining the potential of multi-modality vision language models. *arXiv preprint arXiv:2403.18814*.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2023b. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*.

Zhiqi Li, Guo Chen, Shilong Liu, Shihao Wang, Vibashan VS, Yishen Ji, Shiyi Lan, Hao Zhang, Yilin Zhao, Subhashree Radhakrishnan, and 1 others. 2025. Eagle 2: Building post-training data strategies from scratch for frontier vision-language models. *arXiv preprint arXiv:2501.14818*.

Ziyi Lin, Chris Liu, Renrui Zhang, Peng Gao, Longtian Qiu, Han Xiao, Han Qiu, Chen Lin, Wenqi Shao, Keqin Chen, Jiaming Hans, Siyuan Wang, Yichi Zhang, Xuming He, Hongsheng Li, and Yu Qiao. 2023. Sphinx: The joint mixing of weights, tasks, and visual embeddings for multi-modal large language models. Technical report, arXiv preprint arXiv:2311.07575.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024a. LLaVA-NeXT: Improved reasoning, ocr, and world knowledge. https://llava-vl.github.io/blog/2024-01-30-llava-next/.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024b. Visual instruction tuning. In *NeurIPS*.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. 2024c. Mmbench: Is your multi-modal model an all-around player? In *ECCV*, pages 216–233.

Zhili Liu, Yunhao Gou, Kai Chen, Lanqing Hong, Jiahui Gao, Fei Mi, Yu Zhang, Zhenguo Li, Xin Jiang, Qun Liu, and James Kwok. 2025. Mixture of insighTful experts (MoTE): The synergy of reasoning chains and

expert mixtures in self-alignment. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3022–3038.

Ning Lu, Shengcai Liu, Jiahao Wu, Weiyu Chen, Zhirui Zhang, Yew-Soon Ong, Qi Wang, and Ke Tang. 2025. Safe Delta: Consistently preserving safety when fine-tuning LLMs on diverse datasets. *arXiv preprint arXiv:2505.12038*.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*.

Gen Luo, Yiyi Zhou, Yuxin Zhang, Xiawu Zheng, Xiaoshuai Sun, and Rongrong Ji. 2024. Feast your eyes: Mixture-of-resolution adaptation for multimodal large language models. *arXiv preprint arXiv:2403.03003*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *CVPR*.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. OK-VQA: A visual question answering benchmark requiring external knowledge. In *CVPR*.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When less is more: Investigating data pruning for pretraining LLMs at scale. *arXiv preprint arXiv:2309.04564*.

Aditya Krishna Menon, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. 2019. Can gradient clipping mitigate label noise? In *ICLR*.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. 2019. OCR-VQA: Visual question answering by reading text in images. In *ICDAR*.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, and 1 others. 2023. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. In *NeurIPS*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *ICML*.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. 2022. A-OKVQA: A benchmark for visual question answering using world knowledge. In *ECCV*.

ShareGPT. 2023. https://sharegpt.com/.

Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. 2020. TexCaps: A dataset for image captioning with reading comprehension. In *ECCV*.

Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. 2019. Towards VQA models that can read. In *CVPR*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2018. Joint optimization framework for learning with noisy labels. In *CVPR*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, and 1 others. 2024. Cambrian-1: A fully open, vision-centric exploration of multimodal LLMs. *arXiv preprint arXiv:2406.16860*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. Technical report, arXiv preprint arXiv:2302.13971.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. 2024. Assessing the brittleness of safety alignment via pruning and low-rank modifications. *arXiv preprint arXiv:2402.05162*.

Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. 2020. Combating noisy labels by agreement: A joint training method with co-regularization. In *CVPR*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, and 1 others. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Junjie Wu, Tsz Ting Chung, Kai Chen, and Dit-Yan Yeung. 2024a. Unified triplet-level hallucination evaluation for large vision-language models. *arXiv preprint arXiv:2410.23114*.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, and 1 others. 2024b. DeepSeek-VL2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*.

Tianyi Xiong, Xiyao Wang, Dong Guo, Qinghao Ye, Haoqi Fan, Quanquan Gu, Heng Huang, and Chunyuan Li. 2024. LLaVA-Critic: Learning to evaluate multimodal models. *arXiv preprint arXiv:2410.02712*.

Hansi Yang, Quanming Yao, Bo Han, and James T. Kwok. 2024. Searching to exploit memorization effect in deep learning with noisy labels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):7833–7849.

Quanming Yao, Hansi Yang, Bo Han, Gang Niu, and James Kwok. 2020. Searching to exploit memorization effect in learning from corrupted labels. In *ICML*.

Kun Yi and Jianxin Wu. 2019. Probabilistic end-to-end noise correction for learning with noisy labels. In *CVPR*.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2024. MM-Vet: Evaluating large multimodal models for integrated capabilities. In *ICML*.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *ICLR*.

Zhilu Zhang and Mert Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*.

Zizhao Zhang, Han Zhang, Sercan O Arik, Honglak Lee, and Tomas Pfister. 2020. Distilling effective supervision from severe label noise. In *CVPR*.

Appendix

You will be provided with an image and several question-answer pairs.
For each answer, modify it to make it incorrect based on the image with the following constraint:
1. The answer should look plausible/correct without seeing the image.
2. One can only verify the correctness of the answers with the image.

##
Here are questions and the answers to be modified:

qid: 0
Question: {}
Answer: {}
qid: 1
Question: {}
Answer: {}
##
Please return the qid and the modified answers STRICTLY in the same order as they are given to you.
DO NOT include anything else.
DO NOT miss any QA pairs provided.

##
Here is an example format for your response.
qid: 0
[modified answer 0]
qid: 1
[modified answer 1]

Figure 8: **Prompts for generating corrupted data.**

A Details in Figure 2

The "Simple FT" results in Figure 2 are aggregated from Figure 16 with LLaMA-3.1-8B (experiment details in Sec. 4). Results of "Ours" are aggregated from Figures 7 and 19 (experiment details in Sec. 5.2). The results of "Disable Params." is taken from Table 1 with details in Sec. 4.2. The figure of precision under various corruption levels is obtained from Figure 5b using LLaMA-3.1-8B at a recall of 0.5 (see Sec. 5.1 for more details).

B Dataset Details

B.1 Corrupted Datasets

The prompt used by GPT-4o (described in Sec 3) to generate corrupted samples is shown in Figure 8. We provide examples of corrupted samples for each of the datasets used in Figures 9-15. As can be seen, GPT produces corrupted samples in the following ways: inducing object hallucination in a conversation (Figure 9), changing the option letter in multiple choice VQA (Figure 10), replacing the correct answer in VQA with a plausible but incorrect one (Figures 11-14), and generating wrong captions (Figure 15).

B.2 Task Taxonomy

To better understand the effect of corruption on the fine-tuning process of MLLMs, we categorize the training and evaluation tasks into 3 classes according to their response formatting prompts. These prompts specify how the MLLM should format a response when queried with a question.

- VQA (visual question answering): The MLLM is prompted to answer shortly via *"Answer the question using a single word or phrase."*
- MC-VQA (multiple-choice VQA): The MLLM is prompted to answer only the option

Question: What are the main components of the meal on the white plate?
Answer: The main components of the meal on the white plate are a lobster sandwich and a bowl of clam chowder soup.
Answer (corrupted): The main components of the meal on the white plate are a turkey sandwich and a bowl of tomato soup.

Question: What is the setting of the meal? Is it placed on a specific type of surface?
Answer: The meal is displayed on a white plate that sits atop a wooden table or counter.
Answer (corrupted): The meal is displayed on a white plate that sits atop a marble countertop.

Question: What are some ingredients typically found in a lobster sandwich and clam chowder soup?
Answer: A lobster sandwich typically contains chunks of tender, cooked lobster meat, which may be combined with ingredients such as mayonnaise, lemon juice, celery, herbs, and spices, depending on the recipe. The lobster mixture is typically served cold or room temperature inside a lightly toasted bun or bread, often garnished with lettuce or other greens. \n\nClam chowder soup is a creamy, thick soup made primarily with clams, potatoes, onions, and a base made from milk or cream. Other common ingredients include celery, bacon or salt pork, chicken or fish stock, butter, flour, fresh herbs, and various seasonings. In some cases, especially in the New England style of clam chowder, the soup is topped with oyster crackers, which are small, round, unsalted crackers that can be crumbled into the soup or eaten on their own for added crunch and texture

Answer (corrupted): A turkey sandwich typically contains slices of cooked turkey, which may be combined with ingredients such as mayonnaise, mustard, lettuce, tomato, and cheese, depending on the recipe. The turkey mixture is typically served cold or room temperature inside a lightly toasted bun or bread, often garnished with pickles or other condiments.\n\nTomato soup is a creamy, smooth soup made primarily with tomatoes, onions, and a base made from milk or cream. Other common ingredients include garlic, basil, olive oil, chicken or vegetable stock, butter, flour, fresh herbs, and various seasonings. In some cases, the soup is topped with croutons, which are small, crunchy pieces of bread that can be crumbled into the soup or eaten on their own for added texture.

Figure 9: Example of corrupted sample in dataset (**LLaVA-158K**).

Question: What kind of citrus fruit is on top of the leaf on the right side of the white plate?
A. Lime B. lemon C. grapefruit D. orange
Answer with the option's letter from the given choices directly.
Answer: A. lime
Answer (corrupted): C. grapefruit

Question: Please provide the bounding box coordinate of the region this sentence describes: a bowl of creamy soup on a plate with a sandwich and crackers.
Answer: [0.51, 0.29, 0.88, 0.61]
Answer (corrupted): [0.15, 0.25, 0.45, 0.55]

Figure 10: Example of corrupted sample in datasets **A-OKVQA** and **RefCOCO**.

Question: Is the cupcake in the bottom part of the picture? Answer the question using a single word or phrase.
Answer: No
Answer (corrupted): Yes

Question: Is this lettuce or broccoli?
Answer: Broccoli
Answer (corrupted): Lettuce

Question: Is there either a brown table or nightstand?
Answer: No
Answer (corrupted): Yes

Figure 11: Example of corrupted sample in dataset **GQA**.

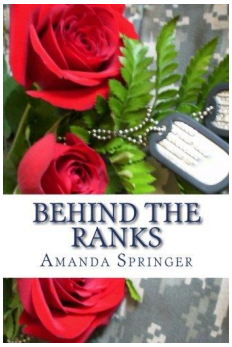
Question: What is written on the wall?Answer the question using a single word or phrase.
Answer: British tennis
Answer (corrupted): French tennis

Question: What color is the court?
Answer: Blue
Answer (corrupted): Green

Question: Is this an indoor match?
Answer: Yes
Answer (corrupted): No

Question: What game is being played by the men in the picture?
Answer: Tennis
Answer (corrupted): Badminton

Figure 12: Example of corrupted sample in dataset **VQAv2**.



Question: Who wrote this book? Answer the question using a single word or phrase.
Answer: Amanda Springer
Answer (corrupted): Michael Johnson

Question: What is the title of this book?
Answer: Behind The Ranks
Answer (corrupted): The Soldier's Tale

Question: Is this a child-care book?
Answer: Yes
Answer (corrupted): No

Figure 13: Example of corrupted sample in dataset **OCRVQA**.

Question: How tall is the average adult male animal in this picture?
Answer the question using a single word or phrase.
Answer: 15 feet
Answer (corrupted): 3 feet

Figure 14: Example of corrupted sample in dataset **OKVQA**.

Question: Provide a one-sentence caption for the provided image.
Reference OCR token: IN, TAKES, IPHOLE

Answer: A lot of candy bars including Almondjoy and Take5.
Answer (corrupted): A lot of candy bars including Snickers and KitKat.

Figure 15: Example of corrupted sample in dataset **TextCaps**.

Category	Training Datasets
VQA	VQAv2 (Goyal et al., 2017), GQA (Hudson and Manning, 2019), OKVQA (Marino et al., 2019), OCRVQA (Mishra et al., 2019)
MC-VQA	A-OKVQA (Schwenk et al., 2022)
Conversation	LLaVA-158K (Liu et al., 2024b), ShareGPT (ShareGPT, 2023)
Others (not categorized)	TextCaps (Sidorov et al., 2020), VG (Krishna et al., 2017), RefCOCO (Kazemzadeh et al., 2014; Mao et al., 2016)

Table 3: **Taxonomy of 10 Training Datasets.** Note that no corruption is injected into ShareGPT in all our experiments as it is a text-only dataset.

via *"Answer with the option's letter from the given choices directly."*

- Conversation: The MLLM receives no format prompt. It responds to the question in a verbose way like a conversation.

- Others (not categorized): The format prompt of this task falls into none of VQA, MC-VQA and Conversation. These format prompts only appear in the training dataset.

Based on the above categories, we list the training and evaluation datasets along with our taxonomy in Tables 3 and 4.

B.3 Evaluation Metrics

The ranges for the scores on MME_P and MME_C are $[0, 2000]$ and $[0, 800]$, respectively. We report both their original values and the normalized values (scaled to $[0, 100]$). All the remaining datasets have a metric range of $[0, 100]$. We also report the average performance by taking the mean scores (normalized) of the 11 tasks in some experiments.

C More on Effects of Corrupted Data

Due to limited space, Figure 3 only presents a subset of evaluation tasks used in this paper. Here we provide the results for all tasks in Figure 16.

Category	Evaluation Datasets
VQA	GQA (Hudson and Manning, 2019), MME (Fu et al., 2024a), POPE (Li et al., 2023b), OKVQA (Marino et al., 2019), TextVQA (Singh et al., 2019)
MC-VQA	MMB (Liu et al., 2024c), SEED-IMG (Li et al., 2023a), SciQA-IMG (Lu et al., 2022)
Conversation	LLaVA-Wild (Liu et al., 2024b), MM-Vet (Yu et al., 2024)

Table 4: **Taxonomy of the 11 Evaluation Datasets.** Note that MME is split into the perception set MME_P and the cognition set MME_C in our experiment.

D Details on Identifying Important and Corruption-related Weights

D.1 Identifying Important Weights

Following (Wei et al., 2024), we use the SNIP score (Lee et al., 2018) to quantify weight importance. For a linear layer with weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the importance score of weight W_{ij} is given by:

$$I(W_{ij}, x) = |W_{ij} \cdot \nabla_{W_{ij}} \ell(x; \boldsymbol{\theta})|,$$

which is the first-order Taylor approximation of the change in the loss when W_{ij} is set to zero. In matrix form, this is:

$$I(W, x) = |W \odot \nabla_W \ell(x; \boldsymbol{\theta})|,$$

where $\odot$ denotes element-wise multiplication. Given a dataset of interest D^*, we aggregate importance scores over all instances, as:

$$I(W) = \mathbb{E}_{x \sim D^*} |W \odot \nabla_W \ell(x; \boldsymbol{\theta})|.$$

Intuitively, $I(W)$ measures how critical each weight is for the model's predictions on D^*. A small $I(W)_{ij}$ indicates that setting W_{ij} to zero has minimal impact on the loss.

Note that in addition to the SNIP score (Lee et al., 2018), there are other methods (Sun et al., 2023; Lu et al., 2025) to identify task-specific weights. However, the ablations on various scores fall out of the focus of this paper. Therefore, we only use the SNIP score throughout this paper.

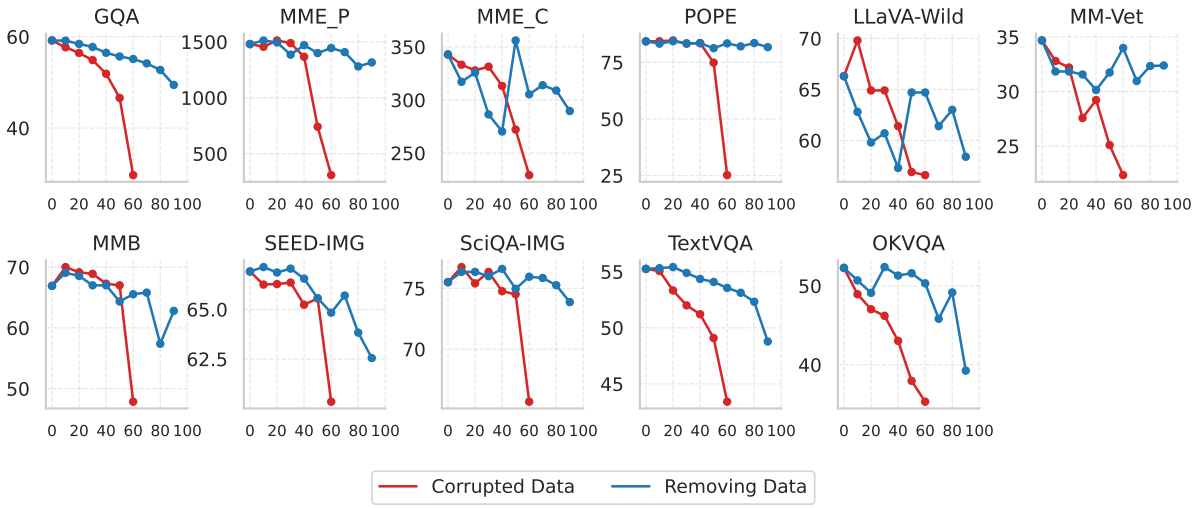


Figure 16: **Performance (y-axis) of LLaVA-1.5 (LLaMA-3.1-8B) under different corruption ratios (x-axis).**

D.2 Identifying Corruption-related Weights

A straightforward approach to identify corruption-related weights is to compute $I(W)$ using a corrupted dataset and select the highest-ranked weights. However, some of these weights may also contribute to correct predictions. To address this, we adopt the approach proposed in Wei et al. (2024) to identify weights that are **specific** to the corrupted data by leveraging set difference.

Specifically, we choose a small dataset consisting of 1K clean samples as D^* and compute I^c using a model trained only with clean data. Similarly, we choose 1K corrupted data as D^* and compute I^n using a model trained on a dataset with 60% corruption. For any pair of sparsity levels (p, q), we define the top-$p\%$ important weights $S^c(p)$ for **clean** samples as the weights whose $I^c_{i,j}$ scores rank within the top $p\%$ of the i-th row of I^c (Sun et al., 2023):

$$S^c(p) = \{(i,j) | I^c_{i,j} \text{ is among the top } p\% \text{ of } I^c_i\}.$$

Similarly, we define the top-$q\%$ important weights $S^n(q)$ for **corrupted** samples as :

$$S^n(q) = \{(i,j) | I^n_{i,j} \text{ is among the top } q\% \text{ of } I^n_i\}.$$

Finally, the isolated weights $S(p, q)$ are defined as the set difference between $S^n(q)$ and $S^c(p)$:

$$S(p,q) = S^n(q) - S^c(p).$$

This approach isolates weights specific to the corrupted samples while filtering out those that are also important for producing clean samples.

Choices of (p, q). We begin with small, identical values for (p, q) and gradually increase them until we observe a significant performance improvement compared to the original model. To prevent the model from collapsing due to the removal of critical weights essential for correct predictions, we set p to be slightly larger than q. This ensures that more of the weights responsible for generating correct, clean responses are preserved. We find that this approach leads to better overall performance, with fewer parameters disabled.

E Details on Identifying Correct Samples using MLLM

E.1 Details on Scores

Perplexity For a sample x, its perplexity is computed as :

$$\text{PPL}(x) = \exp\left(-\frac{1}{n}\sum_{t=1}^{n}\log p(x_y^t | x_y^{<t}, x_c; \boldsymbol{\theta})\right), \quad (1)$$

where n is the length of the response. It quantifies how "surprised" or "uncertain" a model is when generating a response conditioned on an instruction. A lower PPL indicates higher confidence, while a higher PPL suggests that the response is less probable under the model's learned distribution. Intuitively, a higher PPL indicates the sample is more likely to be corrupted according to the models' knowledge.

The loss for this sample is $\log \text{PPL}(x)$.

E.2 Decision Rule and Evaluation

Giving a score (*e.g.*, `PPL` and `Val_PPL`) and the dataset (100K samples with $cr = 50\%$), the following steps are performed:

1. Compute the score for all samples in a dataset.
2. Choose threshold τ, starting from the first to the 100th percentiles of the score. For each threshold τ, we define: $\hat{z}_i = \mathbb{1}\left(\text{score}(x_i) < \tau\right)$, where $\mathbb{1}(\cdot)$ is the indicator function. Intuitively, a sample x_i is predicted as clean ($\hat{z}_i = 1$) if its score is lower than the threshold.
3. Compute the precision (P) and recall (R) scores on N samples for all thresholds as:

$$P = \frac{\sum_{i=1}^{N} \mathbb{1}(z_i = 1 \text{ and } \hat{z}_i = 1)}{\sum_{i=1}^{N} \mathbb{1}(\hat{z}_i = 1)},$$
$$R = \frac{\sum_{i=1}^{N} \mathbb{1}(z_i = 1 \text{ and } \hat{z}_i = 1)}{\sum_{i=1}^{N} \mathbb{1}(z_i = 1)}.$$

4. Draw the precision-recall curve.

E.3 Analysis on Improved Understanding of Corrupted Samples

To understand why improved understanding of corrupted samples lead to the distribution shift of $\hat{p}(z = 1|x_c, \tilde{x}_y; \boldsymbol{\theta})$ on corrupted samples, we start by rewriting $p(z = 1 \mid x_c, \tilde{x}_y)$ using the Bayes' rule as

$$p(z = 1 \mid x_c, \tilde{x}_y) = 1 - p(z = 0 \mid x_c, \tilde{x}_y). \quad (2)$$

By Bayes' theorem, we have

$$p(z = 0 \mid x_c, \tilde{x}_y) = \frac{p(\tilde{x}_y \mid z = 0, x_c)}{p(\tilde{x}_y \mid x_c)} \cdot p(z = 0 \mid x_c). \quad (3)$$

Since corruption is uniformly applied to all samples, the correctness label z is independent of x_c, and thus

$$p(z = 0 \mid x_c) = p(z = 0).$$

Substituting this into (2) and (3), we get

$$p(z = 1 \mid x_c, \tilde{x}_y) = 1 - c \cdot \frac{p(\tilde{x}_y \mid x_c, z = 0)}{p(\tilde{x}_y \mid x_c)},$$

where $c = p(z = 0)$. Therefore,

$$p(z = 1 \mid x_c, \tilde{x}_y) - 1 \propto -\frac{p(\tilde{x}_y \mid x_c, z = 0)}{p(\tilde{x}_y \mid x_c)}.$$

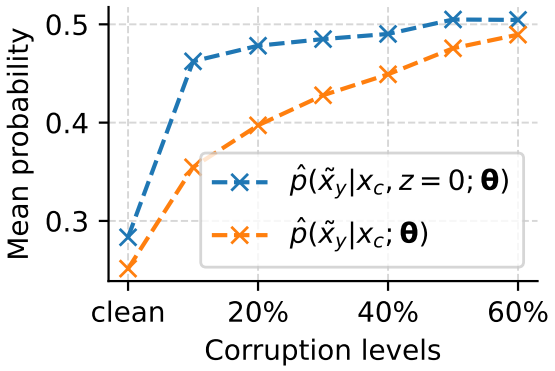


Figure 17: Mean of $\hat{p}(\tilde{x}_y|x_c, z = 0; \boldsymbol{\theta})$ and $\hat{p}(\tilde{x}_y|x_c; \boldsymbol{\theta})$ of corrupted samples.

This shows that $p(z = 1|x_c, \tilde{x}_y) - 1$ is proportional to the negative of the ratio between $p(\tilde{x}_y|x_c, z = 0)$ and $p(\tilde{x}_y|x_c)$. Note that $p(\tilde{x}_y|x_c, z = 0)$ can be regarded as the MLLM's understanding of corrupted samples, which we estimate through explicitly prompting the MLLM similar to `Val_PPL` as follows:

Template for Computing Conditional

<image>Give me an incorrect answer for the following question.
{instruction text}

Figure 17 shows the means of the estimated probabilities $\hat{p}(\tilde{x}_y|x_c, z = 0; \boldsymbol{\theta})$ and $\hat{p}(\tilde{x}_y|x_c; \boldsymbol{\theta})$ at varying corruption levels over the corrupted datasets for fine-tuning. We observe a sharp rise in $\hat{p}(\tilde{x}_y|x_c, z = 0; \boldsymbol{\theta})$ (blue) when transitioning from a clean model to one trained with slight corruption ($cr = 10\%$), followed by a slower growth as the corruption increases further. In contrast, $\hat{p}(\tilde{x}_y|x_c; \boldsymbol{\theta})$ (orange) exhibits a modest increase and remains consistently lower than the other probability.

This shows the MLLM's understanding of corrupted $\hat{p}(\tilde{x}_y|x_c, z = 0; \boldsymbol{\theta})$ gets significantly improved when only including 10% of corruptions.

F Implementation Details on Baselines

F.1 Scores Used for Sample Selection in Further Fine-tuning

Let $p_t(i) = p(x_y^t = i|x_y^{<t}, x_c; \boldsymbol{\theta})$ be the model's predicted probability for token i at timestep t.

Entropy The entropy is computed as

$$H(x;\boldsymbol{\theta}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{V} p_t(i)\log p_t(i),$$

which measures the model's uncertainty, averaged over tokens.

EL2N Denoting one-hot indicator vector for the true token x_y^t as

$$\mathbf{1}_{x_y^t} \in \mathbb{R}^V, \quad \text{where} \quad \mathbf{1}_{x_y^t,i} = \begin{cases} 1 & \text{if } x_y^t = i \\ 0 & \text{otherwise} \end{cases},$$

where V is the vocabulary size. The L2-norm of the output error (EL2N) is computed as:

$$\text{EL2N}(x;\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\sum_{i=1}^{V}\left(p_t(i) - \mathbf{1}_{x_y^t,i}\right)^2},$$

which quantifies the deviation of the predicted probability distribution from the ground truth, averaged over tokens.

GradNorm It is defined as the L2-norm of the gradient vector that is formed by stacking the flattened gradient of each trainable parameter in the model:

$$\text{GradNorm}(x;\boldsymbol{\theta}) = \|\nabla_{\boldsymbol{\theta}}\ell(x_y|x_c;\boldsymbol{\theta})\|_2.$$

When using these scores for sample selection in further fine-tuning, we experiment with both higher and lower values and found that selecting samples with lower values yields better results (Table 9).

F.2 Noise-robust Loss Functions

The mathematical formulations for the loss functions and their corresponding hyper-parameters are provided in Table 5. For GCE, a larger q ($q \in (0,1]$) reduces the sensitivity of the loss to over-confident yet incorrect predictions. In the case of the Phuber CE, hyper-parameter τ ($\tau > 1$) determines the threshold for gradient clipping when the model produces over-confident but incorrect predictions, thereby enhancing its robustness to potential data noise. Overall, these loss functions aim to mitigate the adverse impact of small predicted probabilities for the target classes (see the loss visualizations in Figure 18), which often arise due to corrupted data. To optimize performance, we conduct a hyper-parameter search for q and τ in Figure 20.

F.3 Sample Selection Methods

For MentorNet, Co-teaching and JoCoR (Algorithms 1, 2 and 3), we assume access to an estimated noise level α for the dataset and drop certain amount of data (which is linearly warmed up from 0 to α in T_k steps) according to the defined criterion in the algorithm. Recall the definition of loss in Sec. 3, we let $\mathcal{L}(\mathcal{B};\boldsymbol{\theta})$ be the batch-wise formulation $\sum_{x\in\mathcal{B}}\frac{1}{|\mathcal{B}|}\ell(x;\boldsymbol{\theta})$. We search for the best $\frac{T_k}{T}$ in Figure 20.

MentorNet (Jiang et al., 2018) While various implementations of MentorNet exist, we follow prior works on learning with label noise (Han et al., 2018; Wei et al., 2020) and adopt a simple selection criterion based on the loss computed by the model itself. Specifically, we retain only samples with small loss for model training.

Co-teaching (Han et al., 2018) Originally, Co-teaching utilizes two randomly initialized networks to generate diverse predictions and mitigate error accumulation. However, in the era of MLLM, pre-trained LLMs are required for initialization. To introduce diversity in model predictions, we train them using two independently shuffled data-loaders (with different random seeds). This strategy is also employed in JoCoR (Wei et al., 2020), which similarly relies on two distinct networks.

JoCoR (Wei et al., 2020) It is originally designed for image classification tasks, where the consistency loss minimizes the divergence between two models' class prediction y for image x:

$$\ell_{con}^{\text{cls}}(x;\boldsymbol{\theta}_f,\boldsymbol{\theta}_g) = D_{KL}(p_f\|p_g) + D_{KL}(p_g\|p_f),$$

$$p_f = p(y|x;\boldsymbol{\theta}_f), \quad p_g = p(y|x;\boldsymbol{\theta}_g).$$

For autoregressive sequence generation, the model predicts a sequence token by token, requiring consistency at each step:

$$\ell_{con}^{\text{seq}}(x;\boldsymbol{\theta}_f) = \sum_{t=1}^{T} D_{KL}(p_f^t\|p_g^t) + D_{KL}(p_g^t\|p_f^t),$$

where

$$p_f^t = p(x_y^t|x_y^{<t},x_c;\boldsymbol{\theta}_f),$$
$$p_g^t = p(x_y^t|x_y^{<t},x_c;\boldsymbol{\theta}_g).$$

This ensures divergence minimization at each decoding step rather than a single output. Therefore, we use ℓ_{con}^{cls} as ℓ_{con} and $\mathcal{L}_{con}(\mathcal{B};\boldsymbol{\theta}_f,\boldsymbol{\theta}_g)$ its batch-wise formulation.

Loss	Definition	Hyper-parameters
CE	$-\log(p)$	-
GCE	$(1-p^q)/q$	$q \in (0, 1]$
Phuber CE	$\begin{cases} -\log(p) & p > \frac{1}{\tau} \\ 1 + \log(\tau) - \tau * p & p \leq \frac{1}{\tau} \end{cases}$	$\tau > 1$

Table 5: Mathematical definition for noise robust loss functions. p denotes the probability of predicting specific tokens.

Algorithm 1 MentorNet

Let $\boldsymbol{\theta}$ be the MLLM parameters, T the total number of training steps, $\mathcal{D}$ the corrupted dataset, α the estimated corruption level, η the learning rate, and T_k the warm-up steps;
Shuffle training data $\mathcal{D}$;
for $t = 0, \ldots, T-1$ **do**
  $R(t) = 1 - \min\left\{\frac{t}{T_k} \cdot \alpha, \alpha\right\}$;
  Draw a mini-batch $\mathcal{B}$ from the dataset $\mathcal{D}$;
  Select $R(t) \cdot |\mathcal{B}|$ small-loss samples $\hat{\mathcal{B}}$ from $\mathcal{B}$ based on $\ell(x_i; \boldsymbol{\theta})$;
  Update the parameters: $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\hat{\mathcal{B}}; \boldsymbol{\theta})$;
end for

Methods	**Avg.**	GQA	MME_P	MME_C	POPE	LLaVA Wild	MM-Vet	MMB	SEED IMG	SciQA IMG	Text VQA	OKVQA
Clean	45.78	47.88	1140.91	257.86	81.75	46.60	14.72	45.19	51.14	60.44	36.61	30.02
None (CE)	36.01	37.06	670.50	217.86	48.25	40.80	15.87	33.25	48.61	55.33	32.21	23.95
Noise-robust loss functions												
GCE	38.83	38.23	735.12	254.29	46.56	**44.80**	**19.95**	**40.98**	**50.17**	**60.59**	**36.32**	20.94
Phuber CE	37.61	36.42	776.03	211.43	55.75	44.20	16.10	33.59	49.10	59.44	33.99	19.93
Sample selection (online)												
MentorNet	39.18	37.54	882.57	**268.57**	68.96	36.00	13.44	36.60	48.02	55.18	32.49	25.10
Co-teaching	39.76	36.85	884.94	235.71	69.98	39.20	18.67	37.97	48.18	55.13	32.52	25.20
JoCoR	39.26	36.80	844.99	239.29	70.40	38.10	15.37	36.43	47.61	56.72	32.95	25.29
Further fine-tuning												
EL2N	36.88	37.19	646.61	205.36	61.13	35.00	16.01	36.86	48.35	54.73	33.96	24.40
GradNorm	40.32	40.90	863.06	233.57	72.08	43.80	16.42	36.68	49.32	57.86	32.80	21.33
Entropy	30.18	37.92	841.09	253.21	27.87	38.80	16.01	0.26	42.13	46.41	29.20	19.69
PPL	39.26	39.86	913.69	227.14	74.46	41.20	16.70	26.12	46.71	54.54	33.86	24.32
Val_PPL	**42.28**	**43.62**	**1076.64**	226.79	**77.19**	42.70	18.72	34.97	49.10	55.68	35.02	**25.86**

Table 6: **Comparisons of different corruption-robust strategies at a corruption ratio of 50% on LLaVA-1.5 (Qwen-2.5-0.5B).** Here, **Avg.** refers to the average performance on 11 benchmarks (normalized to 0-100). Best results are **Bold**, second best are underlined.

G Extended Related Work on MLLMs

MLLMs integrate the vision modality into LLMs (Touvron et al., 2023; Chen et al., 2023b; Liu et al., 2025), enabling the advanced understanding and reasoning over visual instructions (Liu et al., 2024b; Bai et al., 2023; Chen et al., 2024b; Gou et al., 2023, 2024, 2025). Recent VLLM works can be categorized into three directions, 1) *Vision encoders* (Oquab et al., 2023; Chen et al., 2021, 2023a) are enhanced and aggregated for robust representations (Lin et al., 2023; Li et al., 2024b; Tong et al., 2024). 2) *High-resolution* methods are proposed to overcome the fixed resolution of pre-trained vision encoders (*e.g.*, 336 × 336 for CLIP (Radford et al., 2021)), enabling LLMs to perceive fine-grained visual information (Liu et al., 2024a; Dong et al., 2024; Huang et al., 2024; Luo et al., 2024). 3) *High-quality instruction data* is essential for VLLMs to generate accurate and well-formed responses (Deitke et al., 2024; Chen et al.,

Algorithm 2 Co-teaching

Let $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_g$ be the parameters for two identical MLLMs, T the total number of training steps, $\mathcal{D}$ the corrupted dataset, α the estimated corruption level, η the learning rate, and T_k the warm-up steps;
for $t = 0, \ldots, T-1$ **do**
  Shuffle training data $\mathcal{D}$ twice with different seeds to get $\mathcal{D}_f$ and $\mathcal{D}_g$;
  $R(t) = 1 - \min\left\{\frac{t}{T_k} \cdot \alpha, \alpha\right\}$;
  Draw mini-batches $\mathcal{B}_f$ and $\mathcal{B}_g$ from datasets $\mathcal{D}_f$ and $\mathcal{D}_g$;
  Select $R(t) \cdot |\mathcal{B}_g|$ small-loss samples $\hat{\mathcal{B}}_g$ from $\mathcal{B}_g$ based on $\ell(x_i; \boldsymbol{\theta}_f)$;
  Select $R(t) \cdot |\mathcal{B}_f|$ small-loss samples $\hat{\mathcal{B}}_f$ from $\mathcal{B}_f$ based on $\ell(x_i; \boldsymbol{\theta}_g)$;
  Update the parameters: $\boldsymbol{\theta}_f = \boldsymbol{\theta}_f - \eta \nabla_{\boldsymbol{\theta}_f} \mathcal{L}(\hat{\mathcal{B}}_f; \boldsymbol{\theta}_f)$;
  Update the parameters: $\boldsymbol{\theta}_g = \boldsymbol{\theta}_g - \eta \nabla_{\boldsymbol{\theta}_g} \mathcal{L}(\hat{\mathcal{B}}_g; \boldsymbol{\theta}_f)$;
end for

Algorithm 3 JoCoR

Let $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_g$ be the parameters for two identical MLLMs, T the total number of training steps, $\mathcal{D}$ the corrupted dataset, α the estimated corruption level, η the learning rate, λ the weighting co-efficient, and T_k the warm-up steps;
for $t = 0, \ldots, T-1$ **do**
  Shuffle training data $\mathcal{D}$ twice with different seeds to get $\mathcal{D}_f$ and $\mathcal{D}_g$;
  $R(t) = 1 - \min\left\{\frac{t}{T_k} \cdot \alpha, \alpha\right\}$;
  Draw mini-batches $\mathcal{B}_f$ and $\mathcal{B}_g$ from datasets $\mathcal{D}_f$ and $\mathcal{D}_g$;
  Select $R(t) \cdot |\mathcal{B}_g|$ small-loss samples $\hat{\mathcal{B}}_g$ from $\mathcal{B}_g$ based on $(1-\lambda)\ell(x_i; \boldsymbol{\theta}_f) + \lambda \ell_{con}(x_i; \boldsymbol{\theta}_f, \boldsymbol{\theta}_g)$;
  Select $R(t) \cdot |\mathcal{B}_f|$ small-loss samples $\hat{\mathcal{B}}_f$ from $\mathcal{B}_f$ based on $(1-\lambda)\ell(x_i; \boldsymbol{\theta}_g) + \lambda \ell_{con}(x_i; \boldsymbol{\theta}_g, \boldsymbol{\theta}_f)$;
  Update the parameters: $\boldsymbol{\theta}_f = \boldsymbol{\theta}_f - \eta \nabla_{\boldsymbol{\theta}_f} \Big((1-\lambda)\mathcal{L}(\hat{\mathcal{B}}_f; \boldsymbol{\theta}_f) + \lambda \mathcal{L}_{con}(\hat{\mathcal{B}}_f; \boldsymbol{\theta}_g, \boldsymbol{\theta}_f)\Big)$;
  Update the parameters: $\boldsymbol{\theta}_g = \boldsymbol{\theta}_g - \eta \nabla_{\boldsymbol{\theta}_g} \Big((1-\lambda)\mathcal{L}(\hat{\mathcal{B}}_g; \boldsymbol{\theta}_g) + \lambda \mathcal{L}_{con}(\hat{\mathcal{B}}_g; \boldsymbol{\theta}_f, \boldsymbol{\theta}_g)\Big)$;
end for

Methods	**Avg.**	GQA	MME_P	MME_C	POPE	LLaVA Wild	MM-Vet	MMB	SEED IMG	SciQA IMG	Text VQA	OKVQA
Clean	55.84	53.78	1299.26	288.93	83.41	62.80	30.05	62.80	63.50	72.98	48.26	35.62
None (CE)	43.71	37.40	792.42	243.93	32.06	59.00	19.54	56.62	62.86	71.64	42.07	29.56
Noise-robust loss functions												
GCE	51.11	44.26	1203.31	249.64	76.49	58.30	28.26	56.62	56.55	68.22	45.40	36.79
Phuber CE	44.00	37.25	795.80	234.64	21.00	60.00	24.95	60.22	63.10	72.04	45.50	30.77
Sample selection (online)												
MentorNet	47.24	37.75	944.09	249.29	61.47	57.00	24.04	56.53	56.71	73.28	42.62	31.92
Co-teaching	47.70	37.32	805.62	252.50	58.10	60.40	27.71	59.79	60.60	**73.82**	42.28	32.85
JoCor	45.84	37.46	658.20	221.43	57.53	57.50	24.68	58.59	61.04	72.04	42.67	32.10
Further fine-tuning												
EL2N	51.06	42.49	1060.59	265.36	74.88	57.70	23.72	63.83	63.16	72.83	43.11	33.75
GradNorm	49.49	43.60	909.99	**303.93**	80.27	55.00	24.86	56.44	61.90	70.90	42.95	24.96
Entropy	48.81	42.11	996.08	266.43	78.76	55.50	20.37	54.30	59.33	70.05	42.98	30.45
PPL	46.87	40.17	1126.94	231.79	35.05	57.60	24.86	**65.64**	63.11	72.63	41.77	29.37
Val_PPL	**55.70**	**52.03**	**1254.83**	287.14	**82.66**	**60.60**	27.11	65.29	**63.51**	72.63	**48.21**	**42.07**

Table 7: **Comparisons of different corruption-robust strategies at a corruption ratio of 50% on LLaVA-1.5 (Qwen-2.5-3B).** Here, **Avg.** refers to the average performance on 11 benchmarks (normalized to 0-100). Best results are **Bold**, second best are underlined.

2024c; Li et al., 2025). This paper is related to the third directions. However, rather than constructing high-quality data, we study the effect of corrupted data on MLLMs and its mitigation.

Methods	**Avg.**	GQA	MME_P	MME_C	POPE	LLaVA Wild	MM-Vet	MMB	SEED IMG	SciQA IMG	Text VQA	OKVQA
Clean	59.75	56.94	1479.14	292.50	84.76	65.00	34.08	70.19	66.16	76.30	52.71	40.57
None (CE)	45.56	38.65	765.30	273.57	26.31	55.20	23.49	64.78	65.64	73.72	46.25	34.62
Noise-robust loss functions												
GCE	51.34	43.66	1129.36	291.43	68.83	57.70	27.80	57.73	57.15	72.24	49.38	37.34
Phuber CE	47.32	39.73	848.28	269.64	31.47	60.10	28.35	64.78	64.46	73.57	49.04	32.93
Sample selection (online)												
MentorNet	48.22	40.98	1055.65	280.36	52.27	54.30	26.88	55.33	57.10	74.02	45.81	35.92
Co-teaching	47.23	38.44	668.22	235.00	60.50	54.30	23.44	65.03	61.66	75.06	45.97	32.31
JoCor	46.73	38.56	633.31	259.64	51.35	53.10	23.12	63.06	61.34	74.67	47.13	37.63
Further fine-tuning												
EL2N	50.52	43.96	1046.46	261.43	52.98	57.30	26.42	64.95	64.81	74.62	47.11	38.62
GradNorm	54.69	47.92	1190.76	303.21	**84.32**	56.20	26.65	65.98	64.91	75.36	48.60	34.20
Entropy	49.71	43.82	1159.33	305.00	42.42	57.20	27.11	61.86	63.09	72.43	45.07	37.72
PPL	54.68	45.52	1305.64	296.79	77.34	53.50	27.48	**68.64**	64.91	75.56	47.68	38.45
Val_PPL	**59.15**	**52.02**	**1417.26**	**307.14**	83.19	**66.50**	**32.61**	68.38	**66.22**	**76.10**	**52.02**	**44.29**

Table 8: **Comparisons of different corruption-robust strategies at a corruption ratio of 50% on LLaVA-1.5 (Qwen-2.5-7B).** Here, **Avg.** refers to the average performance on 11 benchmarks (normalized to 0-100). Best results are **Bold**, second best are underlined.

Models	EL2N		GradNorm		Entropy	
	↓	↑	↓	↑	↓	↑
Qwen-2.5-0.5B	**36.88**	28.11	**47.07**	40.32	**30.18**	31.67
Qwen-2.5-3B	**51.06**	30.41	**49.49**	34.09	**48.81**	38.01
Qwen-2.5-7B	**50.52**	23.14	**54.69**	38.52	**49.71**	39.98
LLaMA-3.1-8B	**47.07**	23.32	**55.77**	39.81	**48.60**	30.73

Table 9: **Further fine-tuning on the bottom (↓) and top (↑) 30% subsets based on EL2N, GradNorm, and Entropy scores.** The reported results represent the average performance. All models are initially fine-tuned on data with a corruption ratio of $cr = 50\%$.

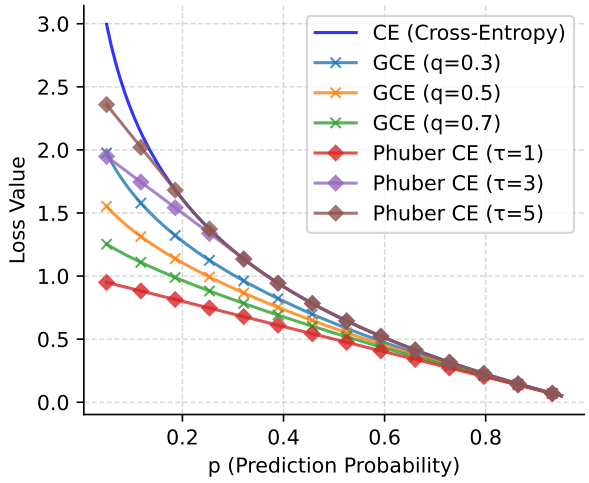


Figure 18: **Visualization of noise-robust loss functions.**

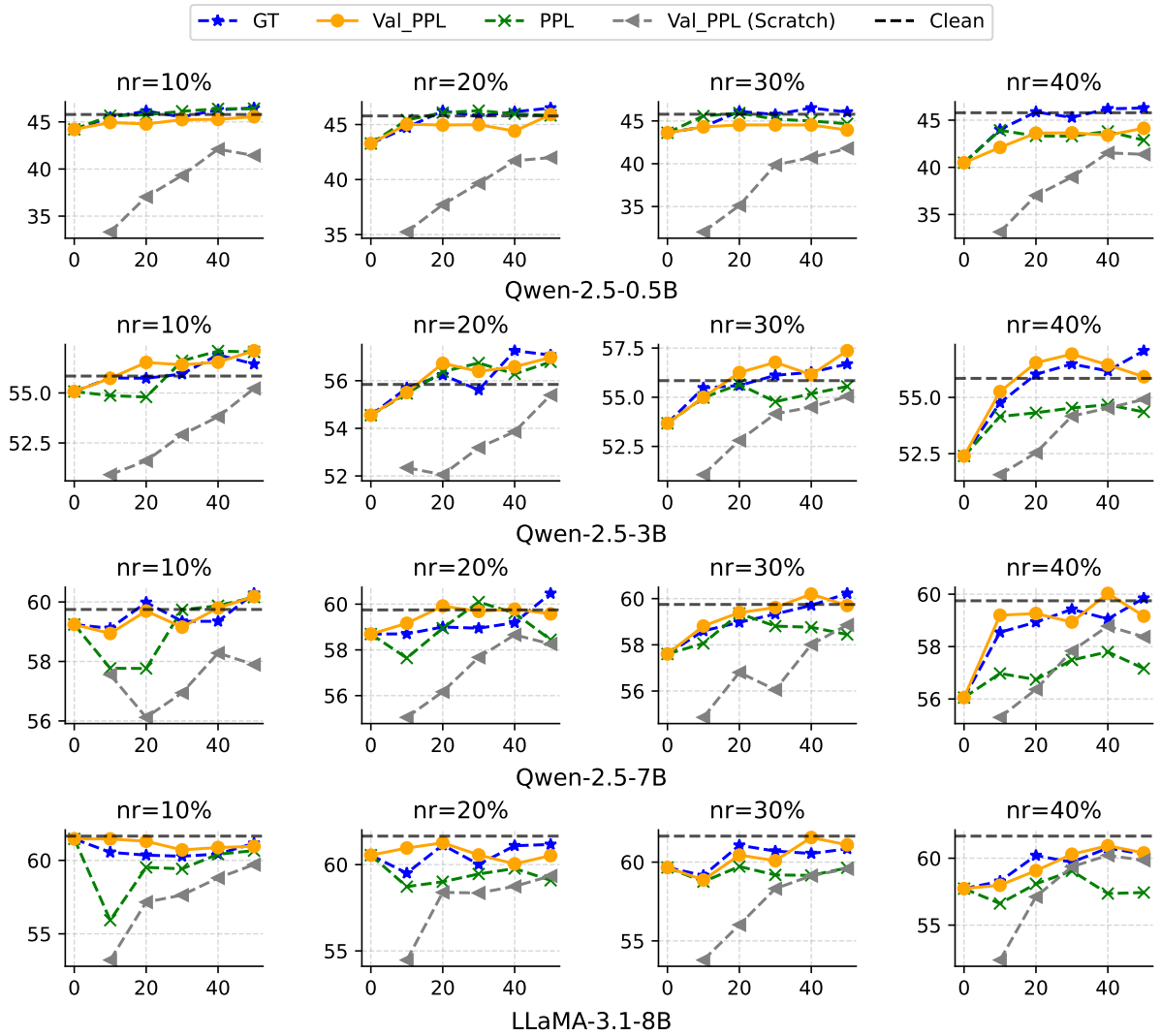


Figure 19: **Average model performance of models further fine-tuned (or fine-tuned from scratch) on data from data from different sources**. Further fine-tuned models are trained on dataset with various corruption levels (10%-40%) at first. Results with 20% and 50% corruption are in Figure 7.

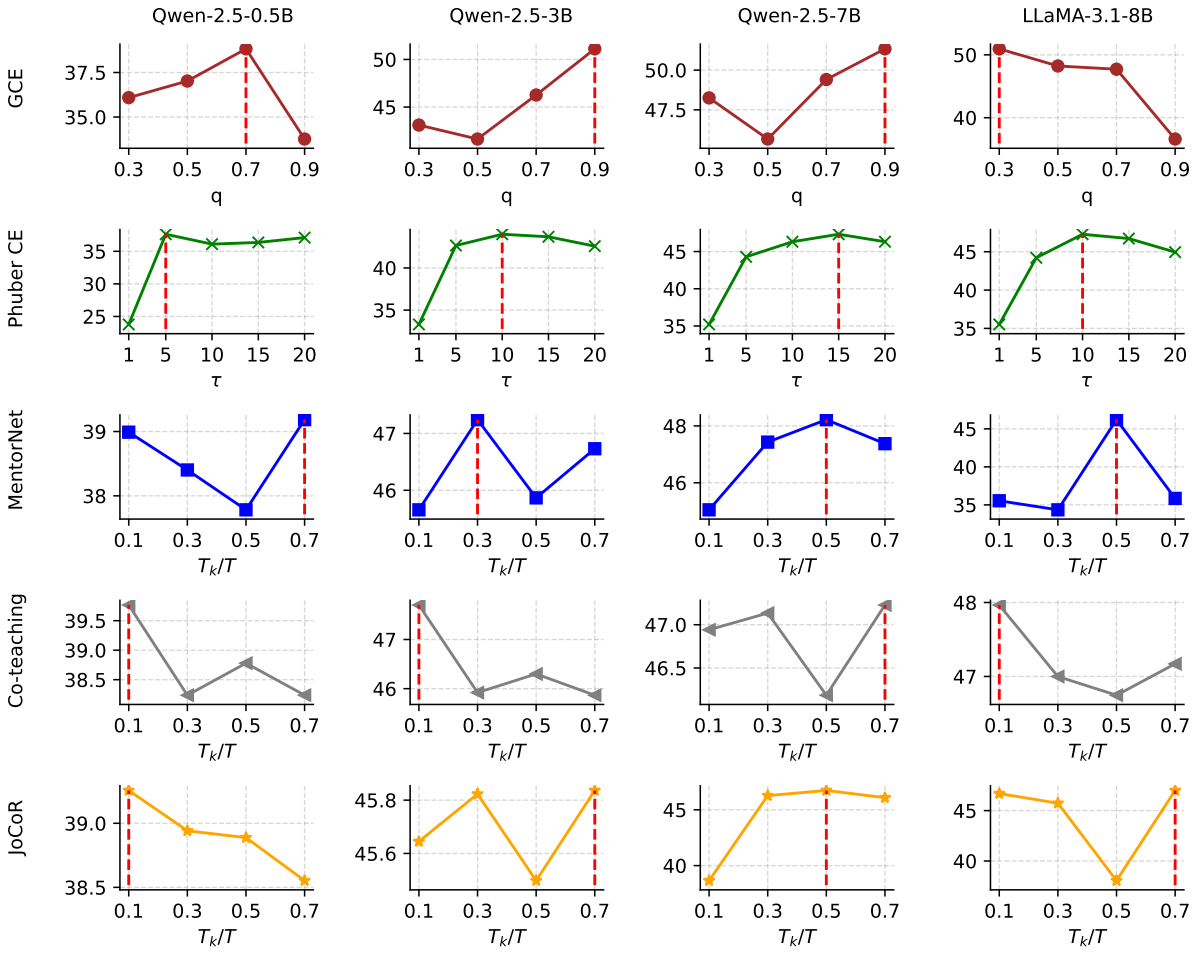


Figure 20: **Ablations on the choices of hyper-parameters for noise-robust loss functions and sample selection methods.** Average performance are reported. All experiments are conducted on datasets with $cr = 50\%$ and best results are indicated by red dashed lines and reported in Tables 2, 6, 7 and 8

Table 10: Performance of LLaVA-1.5 (Qwen2.5-3B) fine-tuned on clean and corrupted data and further fine-tuned with the selected clean data, respectively (corrupted data generated by Gemini-2.5-Flash with a corruption ratio (cr%) of 50%).

Methods	**Avg.**	GQA	MME_P	MME_C	POPE	LLaVA Wild	MM-Vet	MMB	SEED IMG	SciQA IMG	Text VQA	OKVQA
Simple fine-tuning												
Clean	55.8	53.8	1299.3	288.9	83.4	62.8	30.0	62.8	63.5	73.0	48.3	35.6
cr=50%	36.0	36.1	488.2	271.8	12.4	55.5	18.9	28.4	53.2	63.5	42.0	27.8
Further fine-tuning												
PPL (cr=50%)	47.3	46.0	994.7	288.9	61.2	55.4	22.8	47.9	59.6	70.3	44.6	26.8
Val_PPL (cr=50%)	55.0	51.9	1314.7	295.0	84.3	58.4	25.7	57.8	62.6	71.0	48.2	41.9

Table 11: Performance metrics for different models with LLaVA-1.5 (Qwen2.5-3B) across various evaluation tasks. Metrics include clean and corrupted data scenarios (cr=60%) with specific parameters (p, q) (corrupted data generated by Gemini-2.5-Flash).

Model	% Disabled	(p, q)	Avg.	GQA	MME_P	MME_C	POPE	LLaVA Wild	MM-Vet	MMB	SEED IMG	SciQA IMG	Text VQA	OKVQA
Clean	-	-	55.8	53.8	1299.3	288.9	83.4	62.8	30.0	62.8	63.5	73.0	48.3	35.6
cr=60%	0	-	31.5	30.9	290	230	20.5	59.7	19.4	9.6	40.4	57.6	36.4	29.1
cr=60%	1.42	(22, 20)	50.2	47.1	1184	262	79.6	57.6	23.4	46.6	62.9	72.2	44.0	26.5

Table 12: Precisions at different recall values (10%-90%) with LLaVA-1.5 (Qwen2.5-3B) (clean or fine-tuned with $cr = 50\%$ corrupted data) when using PPL and Val_PPL to select clean samples (corrupted data generated by Gemini-2.5-Flash).

Model	10%	20%	30%	40%	50%	60%	70%	80%	90%
Val_PPL (clean)	89.1	84.8	79.7	74.7	70.0	66.6	63.5	60.4	56.7
PPL (cr=50%)	88.7	80.6	70.2	58.8	55.0	54.3	54.1	54.3	54.4
Val_PPL (cr=50%)	96.4	94.0	90.3	86.8	81.7	79.3	76.2	69.8	62.1

Table 13: Performance of LLaVA-1.5 (Qwen2.5-7B) fine-tuned on clean and corrupted data and further fine-tuned with selected clean data, respectively (corrupted data generated by a weak MLLM with a corruption ratio (cr%) of 50%).

Methods	**Avg.**	GQA	MME_P	MME_C	POPE	LLaVA Wild	MM-Vet	MMB	SEED IMG	SciQA IMG	Text VQA	OKVQA
Simple fine-tuning												
Clean	59.8	56.9	1479.1	292.5	84.8	65.0	34.1	70.2	66.2	76.3	52.7	40.6
cr=50%	47.8	43.6	1124.7	268.2	72.2	33.4	21.5	51.1	61.0	74.3	44.4	35.0
Further fine-tuning												
PPL (cr=50%)	48.4	44.6	1141.5	283.6	79.9	30.0	21.7	58.5	61.6	74.9	43.8	25.1
Val_PPL (cr=50%)	57.2	52.9	1317.7	315.4	85.0	61.0	29.6	62.9	64.4	75.0	49.5	43.5

Table 14: Performance of Qwen2-VL-2B fine-tuned on clean and corrupted data and further fine-tuned with selected clean data, respectively (corrupted data generated by GPT-4o with a corruption ratio (cr%) of 50%).

Methods	**Avg.**	GQA	MME_P	MME_C	POPE	LLaVA Wild	MM-Vet	MMB	SEED IMG	SciQA IMG	Text VQA	OKVQA
Simple fine-tuning												
Clean	64.7	61.5	1459.0	375.0	86.2	66.6	46.0	59.9	73.1	75.6	74.5	48.6
cr=10%	64.6	60.5	1528.0	395.0	86.5	68.2	44.5	56.9	72.7	73.5	73.7	47.8
cr=20%	62.9	60.4	1455.0	332.0	84.7	69.7	42.7	51.9	72.8	76.6	72.4	46.7
cr=30%	60.1	58.6	1422.0	261.0	83.9	63.5	40.3	48.0	72.7	75.2	72.3	43.0
cr=40%	58.6	54.7	1454.0	262.0	74.9	67.7	42.8	42.3	72.2	74.0	68.4	42.2
cr=50%	51.1	46.1	1276.0	225.0	53.8	66.0	40.1	24.7	70.3	69.1	62.7	37.0
Further fine-tuning												
(PPL) cr=50%	54.0	48.6	1323.0	297.0	58.2	65.8	39.0	32.0	70.9	70.3	67.7	38.3
(Val_PPL) cr=50%	60.8	60.4	1387.9	348.0	86.9	64.3	43.9	40.5	71.7	66.2	73.6	47.9

Table 15: Precisions at different recall values (10%-90%) with Qwen2-VL (clean or fine-tuned with $cr = 50\%$ corrupted data) when using PPL and Val_PPL to select clean samples (corrupted data generated by GPT-4o).

Model	10%	20%	30%	40%	50%	60%	70%	80%	90%
Val_PPL (clean)	86.1	86.1	84.7	83.2	80.6	77.4	72.7	66.5	57.6
PPL (cr=50%)	90.1	80.8	74.3	66.0	61.1	57.7	55.2	55.0	54.8
Val_PPL (cr=50%)	89.6	89.3	90.9	88.6	86.9	85.9	80.6	79.4	66.2