MoDoMoDo: Multi-Domain Data Mixtures for Multimodal LLM Reinforcement Learning

Yiqing Liang[1]*, Jielin Qiu[2], Wenhao Ding[3], Zuxin Liu[2], James Tompkin[1], Mengdi Xu[4], Mengzhou Xia[5], Zhengzhong Tu[6], Laixi Shi[7], Jiacheng Zhu[8]*
[1]Brown University [2]Salesforce AI Research [3]NVIDIA Research
[4]Carnegie Mellon University [5]Princeton University [6]Texas A&M University
[7]California Institute of Technology [8]MIT CSAIL

Project Website: `https://modomodo-rl.github.io/`

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) has recently emerged as a powerful paradigm for post-training large language models (LLMs), achieving state-of-the-art performance on tasks with structured, verifiable answers. Applying RLVR to Multimodal LLMs (MLLMs) presents significant opportunities but is complicated by the broader, heterogeneous nature of vision-language tasks that demand nuanced visual, logical, and spatial capabilities. As such, training MLLMs using RLVR on multiple datasets could be beneficial but creates challenges with conflicting objectives from interaction among diverse datasets, highlighting the need for optimal dataset mixture strategies to improve generalization and reasoning. We introduce a systematic post-training framework for Multimodal LLM RLVR, featuring a rigorous data mixture problem formulation and benchmark implementation. Specifically, (1) We developed a multimodal RLVR framework for multi-dataset post-training by curating a dataset that contains different verifiable vision-language problems and enabling multi-domain online RL learning with different verifiable rewards; (2) We proposed a data mixture strategy that learns to predict the RL fine-tuning outcome from the data mixture distribution, and consequently optimizes the best mixture. Comprehensive experiments showcase that multi-domain RLVR training, when combined with mixture prediction strategies, can significantly boost MLLM general reasoning capacities. Our best mixture improves the post-trained model's accuracy on out-of-distribution benchmarks by an average of 5.24% compared to the same model post-trained with uniform data mixture, and by a total of 20.74% compared to the pre-finetuning baseline.

1 Introduction

Multimodal large language models (MLLMs)[46] are significantly expanding the frontiers of artificial intelligence. They integrate diverse data types like images and audio with text to achieve a more comprehensive understanding and enable richer, human-natural AI interactions, unlocking advanced capabilities such as cross-modal reasoning and spatial intelligence [50, 1, 31, 42]. Effective post-training of multimodal LLMs remains challenging as they must integrate diverse data modalities, whose data is orders of magnitude scarcer than text-only LLMs [59] and present fundamentally different problem features. They translate into a broad spectrum of distinct capabilities—spatial

[1]* corresponding: yiqing_liang@brown.edu, zjc@mit.edu

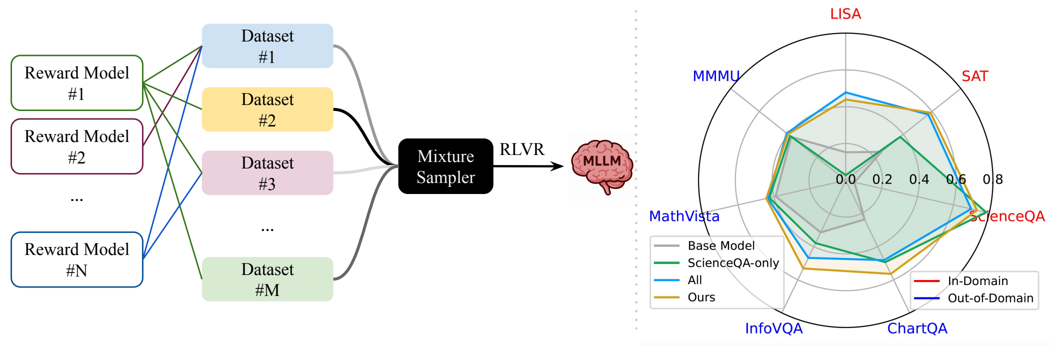


Figure 1: **MoDoMoDo** is a framework that combines *Multi-Domain Data Mixtures* with *Multimodal LLM Reinforcement Learning*, enabling generalizable performance gain across diverse VL tasks. Models trained with our estimated optimal mixtures can outperform those trained with naive mixtures on in-domain and out-of-domain benchmarks.

reasoning, fine-grained recognition, chart interpretation, and more. In response, researchers have developed and adapted various fine-tuning techniques for MLLMs, including instruction following [32], Reinforcement Learning from Human Feedback (RLHF) [47, 59], Direct Preference Optimization (DPO)[51, 61], and online reinforcement learning (RL) approaches [15, 56, 7].

Recently, reinforcement learning with verifiable rewards (RLVR)[55, 44] has emerged as a powerful post-training paradigm for text-based language models, delivering state-of-the-art performance in domains with programmatically checkable answers, such as mathematics and code. By replacing noisy human preferences with rule-based signals, RLVR yields stable optimization and strong generalization[55, 15, 44]. However, using RLVR to enable MLLMs presents significant challenges due to the relative scarcity of accessible verifiable datasets compared to the huge demand for data driven by inherently richer and heterogeneous multimodal tasks. Each verifiable reward tackles a slice of that capability spectrum, leaving large gaps in multimodal reasoning. Existing effective attempts that transfer vision tasks into verifiable reward signals for RL fine-tuning often only focus on one specific multimodal task domain (e.g., VQA [7], object detection [56], math [49, 26], and video [14]). Such single-domain focus is inadequate for achieving the wide range of capabilities required by MLLMs.

To address this, it is beneficial to incorporate multiple datasets from diverse multimodal task domains during post-training, rather than relying on a single dataset, to ensure broad coverage and generalization across a wide range of capabilities. This is particularly important for multimodal LLMs, given the vast number of distinct tasks arising from multimodal compositions. However, using multiple training datasets introduces challenges, including potential conflicting objectives resulting from interactions among diverse datasets, as well as corresponding unstable behaviors during training. These challenges underscore the need to optimize dataset mixture strategies. Notably, data mixture has been extensively investigated and has demonstrated its effectiveness in various scenarios, such as supervised deep learning, pre-training LLMs, and supervised fine-tuning. Here we seek to answer:

*(**Q**) How to mix diverse datasets in RLVR to achieve the wide range of multimodal capabilities?*

To enable efficient multi-dataset multimodal RLVR for multimodal LLM, our main contributions are:

- We develop a **novel multimodal RLVR framework** for **multi-dataset post-training**, designed to optimize the strategies for mixing these datasets to achieve desired capabilities. This includes the creation of five image-text datasets with different verifiable rewards, as well as the RLVR framework for training on these datasets (Sec. 3.2).
- We propose a **multi-domain data mixture optimization strategy**, which learns a surrogate function to predict the outcome of RL finetuning given the mixture distribution (Sec. 2.3). We show that a multivariate quadratic function can capture the counterfactual interplay among different datasets (Fig. 6).
- We comprehensively **evaluate the performance on various evaluation sets**, including MMMU, MathVista, InfoVQA, and ChartQA. The predicted optimal data mixture distribution drives the finetuned model to have strong and generalizable reasoning capability, compared with any other standalone or naive mixture recipes (Sec. 3.3).

2 Multimodal RLVR with Data Mixtures

In this section, we first present the problem formulation of RLVR with data mixtures, framed as a bi-level optimization problem. Subsequently, we propose an approach to optimize the data mixture strategy using a two-step procedure, incorporating a prior model for the objective of the inner optimization problem.

2.1 Reinforcement Learning with Verifiable Reward (RLVR) with Data Mixtures

Multiple multimodal datasets. Suppose there are m domains $\{D_i := (S_i, r_i)\}_{i=1}^m$, where $S_i = \{x_{i,1}, ..., x_{i,n}\} \sim P_{s,i}$ are prompt samples draw from distribution $P_{s,i}$. Given a prompt $x_{i,n} \sim S_i$ and an LLM policy model $\pi_\theta(y|x)$, the reward model $r_i : \mathcal{X} \times \mathcal{Y} \mapsto \{0, 1\}$ labels each prompt and generation pairs $(x_{i,n}, y_{i,n})$ with binary score as per verifiable criteria.

Multimodal RLHF. Reinforcement Learning from Human Feedback (RLHF) [77, 47] is a powerful and scalable strategy to align LLMs with human values, and has become a major technique for LLM post-training. Typically, pipelines includes: (1) *Multimodal Supervised Fine-Tuning (SFT)*, where a MLLM is fine-tuned on an instruction following dataset using token-level supervision [32]. (2) *Multimodal Preference Modeling*, which seeks to maximize the difference between the reward for the chosen response and that for the rejected response[61]. (3) *Reinforcement Learning*. A policy model $\pi :\mapsto \Delta(\mathcal{A})$, usually initialized through multimodal SFT checkpoint [48], is trained to generate appropriate responses by maximizing the reward signal given by a reward model $r : \mathcal{X} \times \mathcal{A} \mapsto [0, 1]$ during the post-training stage.

Reinforcement Learning with Verifiable Reward (RLVR) Group Relative Policy Optimization (GRPO) was proposed [16] as a group-based extension of the widely-used RL algorithm PPO [54, 60] to fine-tune the reference policy. It facilitates a variance-reduced and robust fine-tuning process, achieving excellent results in LLM reasoning. Assume the data distribution used in post-training is μ. Mathematically, GRPO maximizes the expected reward of the policy with respect to a group of G actions from the current policy, regularized to ensure that it does not deviate significantly from the reference policy π_{ref}. Let the policy network be parameterized by θ, and denote the policy at the current iteration as π_{old}. GRPO aims to optimize

$$\max_\theta \mathbb{E}_{x \in \mu, a_i \sim \pi_{\mathsf{old}}(x)} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \Bigg(\min \left(\frac{\pi_\theta(a_i|x)}{\pi_{\theta_{\mathsf{old}}}(a_i|x)} A_i, \text{ clip} \left(\frac{\pi_\theta(a_i|x)}{\pi_{\theta_{\mathsf{old}}}(a_i|x)}, 1-\epsilon, 1+\epsilon \right) A_i \right) \\ - \beta D_{\mathrm{KL}}(\pi_\theta(a_i|x) \| \pi_{\mathsf{ref}}(a_i|x)) \Bigg) \Bigg], \tag{1}$$

where the normalized reward feedback $A_i := \frac{r_i - \text{mean}\{r_1, \cdots, r_G\}}{\text{std}\{r_1, \cdots, r_G\}}$, and $D_{\mathrm{KL}}(\cdot)$ represents the Kullback–Leibler (KL) divergence.

Typically, there is one verifiable structured reward function r associated with each dataset sample, which is either (1) verifiable correctness, (2) verification via execution, or (3) verifiable constraints. It has been shown that verifiable rewards balance simplicity and bias and are thought to be less prone to reward hacking than reward models learned from preference data. However, the above verifiable rewards are carefully designed for certain types of datasets. In practice, we want to fine-tune LLM on a number of datasets to cover as large a domain as possible, to enable a generalizable performance across tasks.

2.2 Data Mixture Modeling for Multidomain RLVR

Suppose there are m domains $\{D_i := (S_i, r_i)\}_{i=1}^m$, where $S_i = \{x_{i,1}, ..., x_{i,n}\} \sim P_{s,i}$ are prompt samples draw from distribution $P_{s,i}$. Given a prompt $x_{i,n} \sim S_i$ and an LLM policy model $\pi_\theta(y|x)$, the reward model $r_i : \mathcal{X} \times \mathcal{Y} \mapsto \{0, 1\}$ labels each prompt and generation pairs $(x_{i,n}, y_{i,n})$ with binary score as per verifiable criteria.

Domain weights $w \in \Delta^m$ specifies a probability distribution over the m domains, and consequently a distribution over the training data: $P_w = \sum_{i=1}^{k} w_i \cdot P_{s,i}$. Here, without loss of generality we assume $P_{s,i}$ to be uniform distributions $unif(S) = \frac{1}{|S|}\sum_{x\in S}\delta_x$ where $\delta_x(x')$ is the Dirac function at location x'.

Consider a formal setting for *MLLM RL finetuning on multi-domain data mixture*, We define an RL algorithm $\mathcal{R}$ that takes training data subject to distribution P_w and output a fine-tuned model. Note this RL finetune process is not deterministic. Then, given a fixed input prompt x we can define:

$f_{\mathcal{R}}(x, P_w) :=$ the outcome of training a model on P_w using R and evaluated on input x,

where the outcome is a measure of the performance of the finetuned model on some task, e.g., pass rate at answering vision QA questions. Our objective is to find the optimal mixture distribution that maximizes the outcome on an unseen testing distribution P_{test}

$$\hat{w} := \arg\max_{w} \mathbb{E}_{x\sim P_{test}}\left[f_R(x, P_w)\right]. \tag{2}$$

Here, the complex yet stochastic function $f_R(x,\cdot)$ captures how the training data mixture distribution yields the final outcome, but its brute force estimation is implausibly expensive.

To solve this problem, we aim to find a simple *surrogate* datamodel [20] function $g(\cdot)$ which approximate $f_R(x,\cdot)$ efficiently. Specifically, we transform the challenge of estimating $f_R(x,\cdot)$ into a supervised learning problem, where the training examples are the distribution simplex w' and corresponding 'labels' are given by $f_R(x, w')$. This can be computed by fine-tuning a base model on data $P_{w'}$ using RL algorithm R. Then, our goal is to fit a parametric function g_θ that maps the training data mixture P_w to the outcome of fine-tuning a model using this distribution.

$$g_\theta : \Delta^m \mapsto \mathbb{R} \text{ where } \theta = \arg\min_\theta \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(g_\theta(w_i), f_R(x, w_i)), \tag{3}$$

and $\mathcal{L}(\cdot,\cdot)$ is a fixed loss function (e.g., RMSE). Achieving this goal would reduce the challenge of scrutinizing $f_R(x,\cdot)$—and more generally the map from training data distribution to model responses through RL algorithm R—to the hopefully easier optimization objective.

Afterwards, we seek a distribution of weights that achieves the desired performance by searching for the mixture that maximizes the expected criteria:

$$w^* = \arg\max_w \left[g_{\theta^*}(w)\right] \tag{4}$$

Should we have a good estimate g_θ, we can find the optimal data mixture distribution defined in Eq.(2) by solving the optimization problem in Eq.(4).

2.3 Constructing data mixture models

The complex characteristics of $f_R(x, P_w)$ pose additional challenges in solving Eq.(3), since g_θ is actually predicting the outcome of fine-tuning. And it contains the following steps: (a) We need to parameterize g_θ as a class of function. (b) Sample a collection of $\{w_i^0\}_{i=1}^{n_0}$ as "seed" samples. (c) Performing n_0 RLVR training runs and curate a collection of training samples $\{(w_1^0, f_{\mathcal{R}}(x, P_{w_1^0})), (w_2^0, f_{\mathcal{R}}(x, P_{w_2^0})), ...,\}$. (d) Split these samples into training and validation, and fit g_θ through the empirical risk minimization (ERM) objective. (e) (Optional) Using the estimated $g_{\hat{\theta}}$, seek the predicted optimal data mixture $\hat{w}$ and collect more training samples to refine g_θ (f) Finally, output the optimal weight w^* via optimizing Eq.(4).

1. Collecting such dataset is expensive as it requires doing a full cycle of RL fine-tuning and evaluation. Thus, we carefully select the seed distributions $\{w_i^0\}_{i=1}^{n_0}$ to maximize the information, e.g. $w_1^0 = [1,0,0,0,0]^\top$, $w_2^0 = [0,1,0,0,0]^\top$, and $w_1^0 = [0, 0.25, 0.25, 0.25, 0.25]^\top$. This is motivated by hypothesis that these different domains might have independent or counterfactual effects on evaluation tasks.
2. While we start from parameterizing $g_{\theta:\{a,b\}} := b + a^\top w$ as a linear function, we also try $g_{\theta:\{b,a,C\}} = b + \mathbf{a}^\top\mathbf{w} + \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{C}\,\mathbf{w}$ to capture the non-linear counterfactual relationship between different domains.

3 Experiments

We begin by describing the experimental setup, including dataset curation, reasoning mode, reward models, training and testing strategies, data sampling, and evaluation metrics in Sec. 3.1. Subsequently, Sec. 3.2 explains the design of 3 groups of data mixture strategies examined in our paper, i.e.

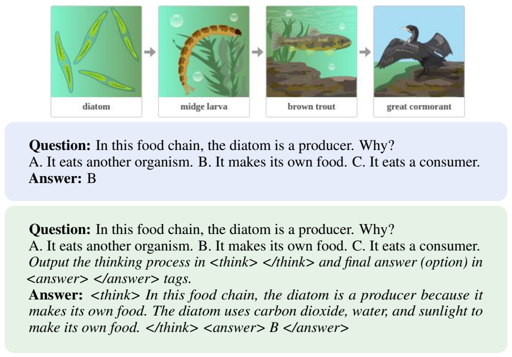
Table 1: **Training and Testing datasets.** The top section shows **Training** datasets processed for RLVR Post-Training with Data Mixture. The bottom two sections show **Testing** datasets containing training datasets' in-distribution test sets (**In**) and common VLM test datasets (**Out**).

Type		Dataset	Domain	Answer Type	Rewards	# samples
Training		COCO [29]	Object Recognition	2D Bounding Box	IoU, Format	5997
		LISA-train [23]	Referring Expression	2D Bounding Box	IoU, Format	1326
		GeoQAV [26]	Math VQA	Multiple Choice	Acc, Format	1969
		SAT-train [52]	Spatial VQA	Natural Language	Acc, Format	15000
		ScienceQA-train [37]	Science VQA	Multiple Choice	Acc, Format	6218
Testing	**In**	LISA-test [23]	Referring Expression	2D Bounding Box		3397
		SAT-test [52]	Spatial VQA	Natural Language		1928
		ScienceQA-test [37]	Science VQA	Multiple Choice		2017
	Out	ChartQA [40]	Chart VQA	Natural Language		2500
		InfoVQA [41]	Infographics VQA	Natural Language		2801
		MathVista [38]	Math VQA	Multiple Choice		1000
		MMMU [73]	General VQA	Multiple Choice		900

seed, heuristic, and model-based. Finally, Sec. 3.3 presents a comprehensive analysis of the results, demonstrating the importance of data mixture in Multimodal LLM RLVR and the efficacy of different data mixture strategies.

3.1 Experimental Setup

Data Curation To expose our model to a broad range of vision-language competencies—from scientific reasoning to fine-grained visual grounding—we assemble a diverse mixture of training corpora where each dataset's question-answer pairs are *verifiable*: the ground-truth answer can be deterministically checked either by exact-string comparison (for multiple-choice letters or short free-form text) or by straight-forward programmatic evaluation (for numeric responses). This guarantees an unambiguous reward signal during RLVR Post-Training.



Question: In this food chain, the diatom is a producer. Why?
A. It eats another organism. B. It makes its own food. C. It eats a consumer.
Answer: B

Question: In this food chain, the diatom is a producer. Why?
A. It eats another organism. B. It makes its own food. C. It eats a consumer.
Output the thinking process in <think> </think> and final answer (option) in <answer> </answer> tags.
Answer: *<think> In this food chain, the diatom is a producer because it makes its own food. The diatom uses carbon dioxide, water, and sunlight to make its own food. </think> <answer> B </answer>*

Figure 2: Demonstration of a General Question-Answer Pair **With** and **Without** Reasoning.

In addition to training datasets' in-distribution test splits (if exist), we adopt *out-of-distribution* benchmarks that have been widely used to evaluate Multimodal LLMs [75]. Statistics for all training and evaluation sets appear in Tab. 1.

Pre-Processing of Individual Datasets (*some training datasets have corresponding testing datasets*):

1. **COCO** [29]: we adopt the full object-category subset used by Liu et al. [35].
2. **LISA** [23]: each image-bounding-box pair may be associated with multiple questions; we treat every question as an independent sample and shuffle the flattened list. Images whose longer side exceeds 640 pixels are resized while maintaining the aspect ratio. The same pipeline is applied to both the official train and test splits, yielding our LISA-train/test sets.
3. **GeoQAV** [26]: starting from the 8k-example math subset, we remove corrupted items by retaining only those whose answers match the expected multiple-choice regex pattern; oversized images are resized as in LISA.
4. **SAT** [52]: we use the original train split for GRPO Post-Training and val split for evaluation.
5. **ScienceQA** [37]: we keep only problems that reference an image. All answer options are concatenated with the original question into a single multiple-choice prompt, and the ground truth answer is converted to its corresponding letter label for style matching.

Reasoning Mode Rather than permitting the multimodal LLM to emit an answer immediately after observing the image-question pair, we place it in a two-step *reasoning mode*: (1) generate a free-form chain-of-thought, and (2) commit to a concise final answer. We trigger this mode by appending a specialised reasoning prompt to every question. Sec. 3.1 is an example.

Reward Models After the above steps, we extract the final answer via a regular-expression parse. ❶ **Format**: if the regex succeeds, this reward yields 1; otherwise, {Format, Acc, IoU} are all set to zero. ❷ Accuracy (**Acc**): a binary reward indicating an exact match. ❸ Intersection-over-Union (**IoU**): for bounding boxes, a real-valued reward in $[0, 1]$. The specific combination of rewards applied to each training dataset is summarised in Tab. 1.

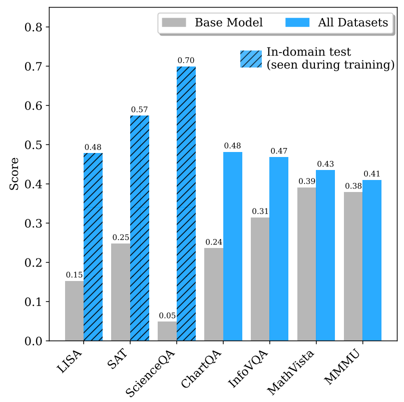


Figure 3: Model Performance before / after GRPO training on *All* data mixture.

Train/Test Strategy To stay within $4\times$ NVIDIA A100 GPUS' budget while retaining adequate representational power, we select Qwen2-VL-2B-Instruct [63] as the base for our GRPO experiments. To further improve efficiency, 1 GPU runs a `vLLM` [22] server that handles all forward generation and periodically sync weight with other processes; the remaining GPUs execute policy update loop. Each of 3 training GPUs processes 2 samples in parallel; 2 gradient accumulation steps give an effective batch of 12 trajectories per update. This provides the $k = 6$ generations needed for GRPO reward while maximizing throughput without exceeding memory limits. The vision encoder is frozen; only the LLM-part parameters are updated. We use AdamW optimizer with β_1=0.9, β_2=0.999, ε=10^{-8}. The learning rate follows a linear schedule: 10% warm-up to a peak $\eta_{\max} = 1 \times 10^{-6}$, then linear decay to zero. Training is performed in `bfloat16` with deterministic seed 42. Regarding GRPO-specific hyperparameters, we use KL coefficient 0.04, reward weights 2.0 for **Acc** and **IoU**, and 1.0 for **Format**. We train each experiment for 1 epoch and report metrics after 2000 steps of training. If a certain mixture training ends early, we use the last checkpoint for evaluation. We also compute metrics after 500 and 1500 steps of training for certain data mixture prediction strategies.

Training Data Sampling Let $w_1, \ldots, w_m$ denote the weights for the m available training sets, with $\sum_{i=1}^{m} w_i = 1$. For each training step, we perform a two-stage draw: (1) sample a domain (D_i) according to $(\Pr(D_i) = w_i)$; (2) sample 1 unseen example from the training split of (D_i). This design aims to force the model to see training samples proportional to the data mixture. If all training samples from a certain dataset have been seen, the training stops.

Evaluation Metrics We compute a score in the range $[0, 1]$ for every test dataset. For LISA-test, we use the mean intersection-over-union (IoU) metric between predicted and ground-truth boxes. For all others, we use the fraction of samples where the answer exactly matches the reference as the score. To provide a concise summary of in/out-of-distribution, we additionally report 2 aggregations. ❶ **In-Score**: the weighted average over datasets that might be seen during training, and ❷ **Out-Score**: the weighted average over datasets that were not used at train time. Weights are proportional to the number of test samples in each dataset (see Tab. 1), ensuring that larger benchmarks contribute more to the overall score.

3.2 Data Mixture Implementation

Seed data mixture We begin with a suite of *seed* mixtures in which every participating dataset is equally sampled at each training step. Three variants are considered:

- *Single*: train on one dataset only ($\alpha_j = 1$ for the chosen dataset, 0 otherwise).
- *Exclude-One*: the mixtures uniformly over all but one dataset; the held-out set's weight is 0.
- *All*: a mixture uniformly over the complete collection ($\alpha_i = 1/m$ for every dataset).

Heuristic data mixture After the seed mixture strategies, we now explore *heuristic* approaches that make assumptions about how each dataset impacts test performance. These data-driven strategies predict weights using empirical performance scores observed in our seed mixture experiments. We employ three heuristic methods:

- *Alpha-family*: These approaches assume independence of datasets, and track each dataset's contribution to performance with a hyperparameter α to balance In-Score and Out-Score priorities:
  - A_{in} ($\alpha = 1.0$) / A_{out} ($\alpha = 0.0$): Preferred for In-Score / Out-Score
  - A_{bal} ($\alpha = 0.5$): Balanced weighting between In-Score and Out-Score
- *Collinearity-Aware Regression (Coli)*: Employs ridge regression and assumes that variance inflation factor (VIF) correction can account for statistical dependencies between datasets.
- *Leave-One-Out Normalization (Norm)*: Assumes that the performance gap when a dataset is excluded compared to *All* is related to its importance in the final mixture.

Model-based data mixture Moving beyond heuristic approaches, we introduce a Covariance Matrix Adaptation Evolution Strategy (CMA-ES) framework that fits a *parametric-model* to approximate the mapping from data mixtures to performance scores.

After fitting empirical observations of both seed and heuristic mixture experiments, the parametric model enables the prediction of performance for any potential mixture without requiring additional empirical observations. By sampling from regions centered on observed mixtures and ranking mixtures based on model estimation, we can discover mixtures of potential that heuristic methods could not reach, particularly when complex synergies exist between datasets.

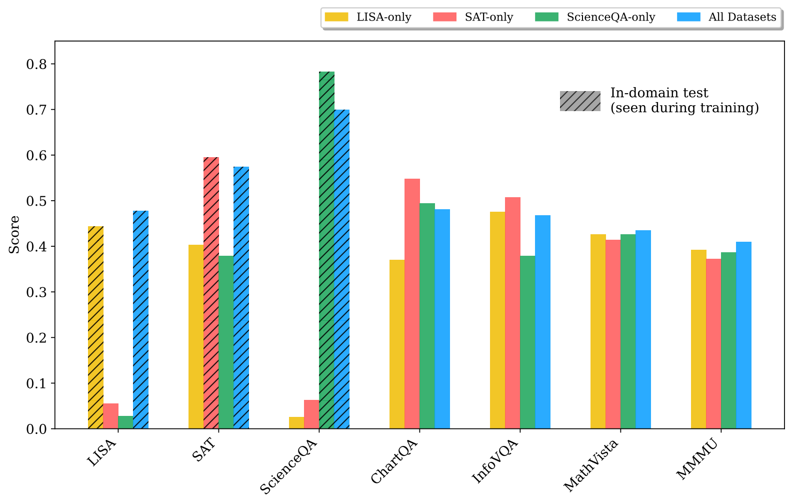


Figure 4: Model Performance Comparison after GRPO training using *All* data mixture and 3 *Single* data mixtures that have in-distribution test set.

3.3 Evaluation Results and Analysis

We begin by showing the generalization potential of basic seed mixtures for multimodal LLM post-training using GRPO. Next, we explore the inherent complexities of combining diverse datasets. Lastly, we justify our parametric model choice for the model-based strategy, and compare the overall efficacy of seed, heuristic, and model-based-yielded data mixtures.

Data Mixture Helps Generalization (Figure 3). Even a simple approach like seed data mixture can yield improvements over the base model, particularly in out-of-domain generalization. GRPO post-training with a seed mixture of all datasets (*All*) enhances model performance across in-domain test sets LISA, SAT, and ScienceQA, where the mixture leads to scores of 0.48, 0.57, and 0.70 respectively, up from the base model's 0.15, 0.25, and 0.05. More importantly, this trend of improvement extends to out-of-domain datasets including ChartQA ($0.24 \rightarrow 0.48$), InfoVQA ($0.31 \rightarrow 0.47$), MathVista ($0.39 \rightarrow 0.43$), and MMMU ($0.38 \rightarrow 0.41$). This indicates that mixture over diverse datasets, even without sophisticated weighting, can enhance both specialized and generalized capabilities.

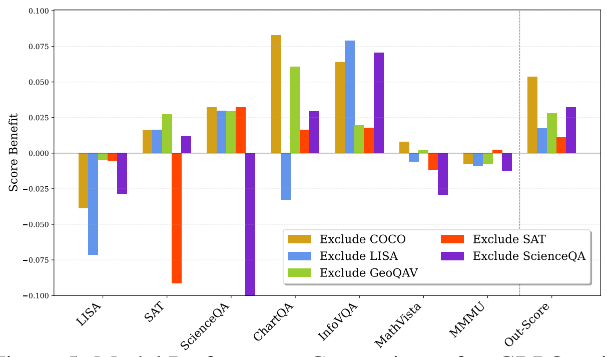


Figure 5: Model Performance Comparison after GRPO training using *All* data mixture and 5 *Exclude-One* data mixtures.

Intricate Nature of Data Mixture (Figures 4 and 5). While the above seed mixture shows improvement on all test benchmarks, finding optimal data mixtures is far from straightforward. First, naive inclusion of more data does not guarantee superior results. For example, Fig. 4 shows that *All*'s performance on ChartQA and InfoVQA is lower than 2 single-dataset mixtures. Furthermore, Fig. 4 illustrates that each dataset might have a distinct impact on model performance. For instance, LISA-only is beneficial for generalization on InfoVQA but hurts ScienceQA score, while ScienceQA-only yields the highest score on the same-domain test set but is less useful for InfoVQA. When there's more than one data source, the interplay between them makes the situation more complicated. Figure 5 displays this complexity by comparing *All* with *Exclude-One* mixtures. The targeted removal of certain datasets might not negatively affect, or could even enhance, performance on particular tasks or overall. For instance, the exclusion of ScienceQA significantly lowers scores on the ScienceQA and ChartQA benchmarks, while its effects elsewhere are mixed. Additionally, excluding one dataset is consistently helpful for overall Out-Score, i.e., useful for out-domain generalization. This variability underscores the need for more intelligent and adaptive data mixture strategies.

Selection of an Effective Parametric data mixture Model (Figure 7). As the seed mixture shows promise but is likely suboptimal due to the aforementioned complexities, we investigated heuristic and model-based strategies to predict optimal data mixture. For model-based strategies, the choice of an appropriate mathematical model to approximate the function from mixture to Out-Score is crucial, as depicted in Fig. 7. A linear (1-dimensional) optimization model, for example, fails to accurately map mixture to performance outcomes even when using all samples for fitting, as evidenced by the significant deviation between "Actual" and "Predicted Scores". Similarly, an analysis using Principal Component Analysis (PCA) on the mixture vectors suggests that the data points are not readily separable in a linearly reduced dimensional space, further pointing to underlying non-linearities. In contrast, a quadratic optimization function is more adept. When employing cross-validation to pick top-1 out of 5 shuffles, the quadratic model (rightmost panel) demonstrates improved ability to fit the training data and, critically, to generalize to unseen test sets. We hypothesize that the advantage of using a quadratic function over 1-dimensional linear function comes from the former's handling of the nuanced dataset interactions. Now, we adopt the quadratic paradigm to guide the search for effective data mixture.

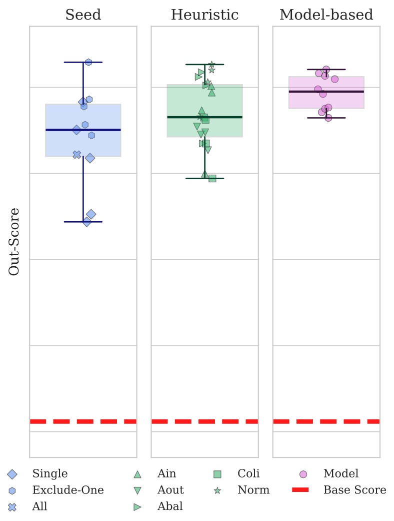


Figure 6: **Data Mixture Strategies' Comparison.** Heuristic (*middle*) and model-based (*right*) achieve higher minima and medians than Seed (*middle*). Model-based further reduces variance.

Comparison of Mixture Strategies (Figure 6). Our investigation of data mixture strategies is summarized in Fig. 6, which presents an aggregated view of out-of-domain scores (Out-Score), grouped by strategy. This figure shows that, as groups, the predicted data mixtures of heuristic and model-based strategies both offer advantages over naive seed mixtures by having higher median and minimum scores. This reinforces the idea that thoughtful data mixing is the key to unlocking better generalization.

When comparing heuristic methods against our model-based approach (which leverages the aforementioned quadratic optimization), the model-based strategy achieves a higher median, a higher minimum, and a competitive maximum. Moreover, the model-based approach exhibits smaller variance compared to both seed and heuristic methods, which is a sign of not only an improvement in average performance but also a gain in reliability.

4 Related Work

RL for aligning language models. Reinforcement Learning from Human Feedback (RLHF) guides LLMs with rewards learned from human-labeled data, most visibly in InstructGPT [47]. Recent variants reduce human effort, e.g., Constitutional AI [3], RLAIF [24], and log-likelihood-based

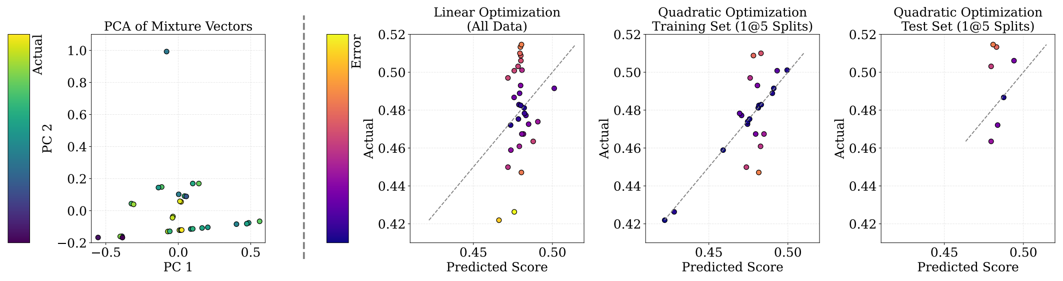


Figure 7: **Model-based mixture optimization.** *Left:* PCA on mixture vectors reveals lack of linear separability, motivating nonlinear modeling. *Middle-left:* Linear regression fails to fit Out-Score accurately, even with all data. *Right:* Quadratic regression fitted on training folds could generalize better to held-out folds, capturing nonlinear interactions critical for mixture prediction.

SimPO [43], or replace value networks with groupwise comparisons (GRPO) and other verifiable-reward methods that markedly improve multi-step reasoning [4, 5, 21].

R1-style multimodal reasoning. DeepSeek-Math/R1 show that GRPO can boost textual reasoning without an explicit reward model [55]. A fast-growing line of work ports this idea to vision–language models, demonstrating domain-specific gains in math, fine-grained recognition, counting, and spatial tasks (e.g., MAYE [39], Visual-RFT [35], R1-V [7]). Follow-up studies refine rewards, curricula, and exploration strategies, achieving further stability and efficiency; yet how to allocate training across heterogeneous multimodal tasks remains open.

Data-mixture strategies. Classic multilingual pretraining up-samples rare languages with temperature sampling [13]. Proxy-model approaches such as DoReMi [66] and RegMix [33] learn mixture weights in two stages, while ODM [2] updates them online. Very recent evidence suggests that even simple heuristics rival these sophisticated schemes [18]. We extend this discussion to vision–language reasoning, treating task-level sampling as a first-class design choice.

Overall, prior studies show that (i) lightweight, verifiable rewards can reliably sharpen reasoning, and (ii) mixture policies critically shape downstream performance. Our work unifies these threads by jointly optimising data mixture and GRPO-style reinforcement for multimodal reasoning. For a full discussion, please refer to the supplementary materials Appendix A.

5 Conclusion

We introduced **MoDoMoDo**, a framework that couples multi-domain data mixtures with reinforcement learning under verifiable rewards (RLVR) for multimodal large language models. Leveraging five complementary image–text datasets equipped with rule-based reward functions, we train a lightweight *quadratic surrogate* to predict post-training performance as a function of the mixture distribution. This surrogate lets us identify an *optimal* mix after only a small set of pilot runs. Models fine-tuned with the learned mixture surpass those trained on any individual dataset or on uniform combinations.

Limitations and Future Work. Our analysis is limited to image–text settings; extending MoDoMoDo to audio, video, and 3-D modalities, as well as to tasks where verifiable signals are sparse or noisy, remains open. On the algorithmic side, exploring surrogate models that incorporate dataset similarity, curriculum schedules, or uncertainty estimates could further reduce pilot-run cost. Finally, connecting our empirical findings to a unified multi-objective RL theory would deepen understanding of why mixture-optimized RLVR generalizes so well.

References

[1] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[2] A. Albalak, L. Pan, C. Raffel, and W. Y. Wang. Efficient online data mixing for language model pre-training. *arXiv preprint arXiv:2312.02406*, 2023.

[3] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, C. Chen, C. Olsson, C. Olah, D. Hernandez, D. Drain, D. Ganguli, D. Li, E. Tran-Johnson, E. Perez, J. Kerr, J. Mueller, J. Ladish, J. Landau, K. Ndousse, K. Lukosuite, L. Lovitt, M. Sellitto, N. Elhage, N. Schiefer, N. Mercado, N. DasSarma, R. Lasenby, R. Larson, S. Ringer, S. Johnston, S. Kravec, S. E. Showk, S. Fort, T. Lanham, T. Telleen-Lawton, T. Conerly, T. Henighan, T. Hume, S. R. Bowman, Z. Hatfield-Dodds, B. Mann, D. Amodei, N. Joseph, S. McCandlish, T. Brown, and J. Kaplan. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[4] X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[5] H. Chen, Y. Feng, Z. Liu, W. Yao, A. Prabhakar, S. Heinecke, R. Ho, P. L. Mui, S. Savarese, C. Xiong, and H. Wang. Language models are hidden reasoners: Unlocking latent reasoning capabilities via self-rewarding. *OpenReview (ICLR)*, 2024.

[6] H. Chen, H. Tu, F. Wang, H. Liu, X. Tang, X. Du, Y. Zhou, and C. Xie. Sft or rl? an early investigation into training r1-like reasoning large vision-language models, 2025.

[7] L. Chen, L. Li, H. Zhao, Y. Song, and Vinci. R1-v: Reinforcing super generalization ability in vision-language models with less than $3. `https://github.com/Deep-Agent/R1-V`, 2025. Accessed: 2025-02-02.

[8] X. Chen, W. Li, C. Liu, C. Xie, X. Hu, C. Ma, F. Zhu, and R. Zhao. On the suitability of reinforcement fine-tuning to visual tasks, 2025.

[9] Y. Chen, Y. Ge, R. Wang, Y. Ge, L. Qiu, Y. Shan, and X. Liu. Exploring the effect of reinforcement learning on video understanding: Insights from seed-bench-r1. *ArXiv Preprint*, 2025.

[10] H. W. Chung, N. Constant, X. Garcia, A. Roberts, Y. Tay, S. Narang, and O. Firat. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. *arXiv preprint arXiv:2304.09151*, 2023.

[11] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv Preprint*, 2025.

[12] Y. Deng, H. Bansal, F. Yin, N. Peng, W. Wang, and K.-W. Chang. Openvlthinker: An early exploration to complex vision-language reasoning via iterative self-improvement. *ArXiv Preprint*, 2025.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[14] K. Feng, K. Gong, B. Li, Z. Guo, Y. Wang, T. Peng, B. Wang, and X. Yue. Video-r1: Reinforcing video reasoning in mllms. *ArXiv Preprint*, 2025.

[15] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Wu, Z.-F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Chen, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, J. Li, H. Li, J. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, L. Yang, L. Zhao, and C. Xiong. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[16] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[17] A. Havrilla, Y. Du, S. C. Raparthy, C. Nalmpantis, J. Dwivedi-Yu, M. Zhuravinskyi, E. Hambro, S. Sukhbaatar, and R. Raileanu. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 2024.

[18] W. Held, B. Paranjape, P. S. Koura, M. Lewis, F. Zhang, and T. Mihaylov. Optimizing pretraining data mixtures with llm-estimated utility. *arXiv preprint arXiv:2501.11747*, 2025.

[19] J. Hu, Y. Zhang, Q. Han, D. Jiang, X. Zhang, and H.-Y. Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

[20] A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022.

[21] A. Kazemnejad, M. Aghajohari, E. Portelance, A. Sordoni, S. Reddy, A. Courville, and N. L. Roux. Vineppo: Unlocking rl potential for llm reasoning through refined credit assignment. *OpenReview (ICLR)*, 2024.

[22] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[23] X. Lai, Z. Tian, Y. Chen, Y. Li, Y. Yuan, S. Liu, and J. Jia. Lisa: Reasoning segmentation via large language model. *ArXiv Preprint*, 2023.

[24] H. Lee, S. Phatale, H. Mansoor, T. Mesnard, J. Ferret, K. Lu, C. Bishop, E. Hall, V. Carbune, A. Rastogi, and S. Prakash. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2024.

[25] J. Leng, C. Huang, L. Huang, B. Y. Lin, W. W. Cohen, H. Wang, and J. Huang. Crosswordbench: Evaluating the reasoning capabilities of llms and lvlms with controllable puzzle generation, 2025.

[26] B. Li, K. Zhang, and A. Marafioti. Multimodal open r1. `https://github.com/EvolvingLMMs-Lab/open-r1-multimodal`, 2025. Accessed: 2025-02-08.

[27] Z. Li, X. Wu, H. Du, H. Nghiem, and G. Shi. A survey of state of the art large vision language models: Alignment, benchmark, evaluations and challenges. *ArXiv Preprint*, 2025.

[28] W. Liang, W. Gao, W. Yu, and W. Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision (ECCV)*, 2014.

[30] Z. Lin, Y. Gao, X. Zhao, Y. Yang, and J. Sang. Mind with eyes: from language reasoning to multimodal reasoning. *ArXiv Preprint*, 2025.

[31] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[32] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[33] Q. Liu, X. Zheng, N. Muennighoff, G. Zeng, L. Dou, T. Pang, J. Jiang, and M. Lin. Regmix: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*, 2024.

[34] X. Liu, J. Ni, Z. Wu, C. Du, L. Dou, H. Wang, T. Pang, and M. Q. Shieh. Noisyrollout: Reinforcing visual reasoning with data augmentation, 2025.

[35] Z. Liu, Z. Sun, Y. Zang, X. Dong, Y. Cao, H. Duan, D. Lin, and J. Wang. Visual-rft: Visual reinforcement fine-tuning. *ArXiv Preprint*, 2025.

[36] Z. Liu, Y. Zhang, F. Liu, C. Zhang, Y. Sun, and J. Wang. Othink-mr1: Stimulating multimodal generalized reasoning capabilities through dynamic reinforcement learning. *ArXiv Preprint*, 2025.

[37] P. Lu, S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark, and A. Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[38] P. Lu, H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K.-W. Chang, M. Galley, and J. Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.

[39] Y. Ma, S. Chern, X. Shen, Y. Zhong, and P. Liu. Rethinking rl scaling for vision language models: A transparent, from-scratch framework and comprehensive evaluation scheme. *ArXiv Preprint*, 2025.

[40] A. Masry, D. X. Long, J. Q. Tan, S. Joty, and E. Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022.

[41] M. Mathew, V. Bagal, R. Tito, D. Karatzas, E. Valveny, and C. Jawahar. Infographicvqa. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1697–1706, January 2022.

[42] B. McKinzie, Z. Gan, J.-P. Fauconnier, S. Dodge, B. Zhang, P. Dufter, D. Shah, X. Du, F. Peng, A. Belyi, et al. Mm1: methods, analysis and insights from multimodal llm pre-training. In *European Conference on Computer Vision*, pages 304–323. Springer, 2024.

[43] Y. Meng, M. Xia, and D. Chen. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.

[44] Y. Mroueh. Reinforcement learning with verifiable rewards: Grpo's effective loss, dynamics, and success amplification. *arXiv preprint arXiv:2503.06639*, 2025.

[45] NVIDIA. Cosmos-reason1: From physical common sense to embodied reasoning. *ArXiv Preprint*, 2025.

[46] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL `https://arxiv.org/abs/2303.08774`.

[47] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[48] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022.

[49] Y. Peng, Chris, X. Wang, Y. Wei, J. Pei, W. Qiu, A. Jian, Y. Hao, J. Pan, T. Xie, L. Ge, R. Zhuang, X. Song, Y. Liu, and Y. Zhou. Skywork r1v: Pioneering multimodal reasoning with chain-of-thought, 2025. URL `https://huggingface.co/Skywork/Skywork-R1V-38B`.

[50] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[51] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

[52] A. Ray, J. Duan, R. Tan, D. Bashkirova, R. Hendrix, K. Ehsani, A. Kembhavi, B. A. Plummer, R. Krishna, K.-H. Zeng, and K. Saenko. Sat: Spatial aptitude training for multimodal language models. *ArXiv Preprint*, 2024.

[53] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[54] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[55] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL `https://arxiv.org/abs/2402.03300`.

[56] H. Shen, P. Liu, J. Li, C. Fang, Y. Ma, J. Liao, Q. Shen, Z. Zhang, K. Zhao, Q. Zhang, R. Xu, and T. Zhao. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

[57] G. Sheng, C. Zhang, Z. Ye, X. Wu, W. Zhang, R. Zhang, Y. Peng, H. Lin, and C. Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

[58] Y. Song, T. Ou, Y. Kong, Z. Li, G. Neubig, and X. Yue. Visualpuzzles: Decoupling multimodal reasoning evaluation from domain knowledge, 2025.

[59] Z. Sun, S. Shen, S. Cao, H. Liu, C. Li, Y. Shen, C. Gan, L.-Y. Gui, Y.-X. Wang, Y. Yang, et al. Aligning large multimodal models with factually augmented rlhf. *arXiv preprint arXiv:2309.14525*, 2023.

[60] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[61] F. Wang, W. Zhou, J. Y. Huang, N. Xu, S. Zhang, H. Poon, and M. Chen. mdpo: Conditional preference optimization for multimodal large language models. *arXiv preprint arXiv:2406.11839*, 2024.

[62] H. Wang, C. Qu, Z. Huang, W. Chu, F. Lin, and W. Chen. Vl-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. *ArXiv Preprint*, 2025.

[63] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, Y. Fan, K. Dang, M. Du, X. Ren, R. Men, D. Liu, C. Zhou, J. Zhou, and J. Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *ArXiv Preprint*, 2024.

[64] X. Wang, Z. Yang, C. Feng, H. Lu, L. Li, C.-C. Lin, K. Lin, F. Huang, and L. Wang. Sota with less: Mcts-guided sample selection for data-efficient visual reasoning self-improvement. *ArXiv Preprint*, 2025.

[65] W. Xiao, L. Gan, W. Dai, W. He, Z. Huang, H. Li, F. Shu, Z. Yu, P. Zhang, H. Jiang, and F. Wu. Fast-slow thinking for large vision-language model reasoning. *ArXiv Preprint*, 2025.

[66] S. M. Xie, H. Pham, X. Dong, N. Du, H. Liu, Y. Lu, P. S. Liang, Q. V. Le, T. Ma, and A. W. Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

[67] W. Xiong, H. Zhang, C. Ye, L. Chen, N. Jiang, and T. Zhang. Self-rewarding correction for mathematical reasoning. *arXiv preprint arXiv:2502.19613*, 2025.

[68] W. Xu, J. Wang, W. Wang, Z. Chen, W. Zhou, A. Yang, L. Lu, H. Li, X. Wang, X. Zhu, W. Wang, J. Dai, and J. Zhu. Visulogic: A benchmark for evaluating visual reasoning in multi-modal large language models. *ArXiv Preprint*, 2025.

[69] J. Yan, Y. Li, Z. Hu, Z. Wang, G. Cui, X. Qu, Y. Cheng, and Y. Zhang. Learning to reason under off-policy guidance. *ArXiv Preprint*, 2025.

[70] Y. Yang, X. He, H. Pan, X. Jiang, Y. Deng, X. Yang, H. Lu, D. Yin, F. Rao, M. Zhu, B. Zhang, and W. Chen. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. *ArXiv Preprint*, 2025.

[71] J. Ye, P. Liu, T. Sun, J. Zhan, Y. Zhou, and X. Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv preprint arXiv:2403.16952*, 2024.

[72] E. Yu, K. Lin, L. Zhao, J. Yin, Y. Peng, H. Wei, J. Sun, C. Han, Z. Ge, X. Zhang, D. Jiang, J. Wang, and W. Tao. Perception r1: Pioneering perception policy with reinforcement learning. *ArXiv Preprint*, 2025.

[73] X. Yue, Y. Ni, K. Zhang, T. Zheng, R. Liu, G. Zhang, S. Stevens, D. Jiang, W. Ren, Y. Sun, C. Wei, B. Yu, R. Yuan, R. Sun, M. Yin, B. Zheng, Z. Yang, Y. Liu, W. Huang, H. Sun, Y. Su, and W. Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of CVPR*, 2024.

[74] Y. Yue, Z. Chen, R. Lu, A. Zhao, Z. Wang, Y. Yue, S. Song, and G. Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *ArXiv Preprint*, 2025.

[75] K. Zhang, B. Li, P. Zhang, F. Pu, J. A. Cahyono, K. Hu, S. Liu, Y. Zhang, J. Yang, C. Li, and Z. Liu. Lmms-eval: Reality check on the evaluation of large multimodal models, 2024. URL `https://arxiv.org/abs/2407.12772`.

[76] H. Zhou, X. Li, R. Wang, M. Cheng, T. Zhou, and C.-J. Hsieh. R1-zero's "aha moment" in visual reasoning on a 2b non-sft model. *ArXiv Preprint*, 2025.

[77] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Related Work

A.1 Reinforcement Learning for LLM

Reinforcement Learning from Human Feedback (RLHF) has become a foundational technique for aligning LLM outputs with human preferences. Early implementations, such as InstructGPT [47], utilized Proximal Policy Optimization (PPO) guided by reward models trained on human-labeled data. Subsequent approaches like Constitutional AI [3] introduced self-supervised objectives to reduce reliance on human feedback. To further minimize human involvement, Reinforcement Learning from AI Feedback (RLAIF) [24] leverages AI-generated feedback. Simplifying the RLHF pipeline, SimPO [43] replaces explicit reward models with implicit rewards derived from the model's own log-likelihood, streamlining training while maintaining performance.

Building upon these foundations, recent research has focused on enhancing LLM reasoning capabilities through RL via verifiable rewards. DeepSeek's R1 models [4, 28] employ Group Relative Policy Optimization (GRPO), which eschews value networks in favor of comparing groups of outputs to reinforce correct reasoning patterns. Similarly, LaTRO [5] frames reasoning as a latent-variable optimization problem, enabling self-improvement without external feedback. [17] compare several RL algorithms (Expert Iteration, PPO, return-conditioned RL) using sparse correctness rewards or learned reward models, finding broadly similar gains in multi-step reasoning. VinePPO [21] addresses credit assignment in multi-step reasoning by utilizing unbiased Monte Carlo returns, and Xiong et al. train models to iteratively generate, verify, and refine their answers using rule-based self-rewards [67]. [19] show that a minimalistic PPO pipeline steadily increases response length and accuracy. On the systems side, VeRL [57] provides a flexible RL execution framework: it combines single- and multi-controller designs to support complex LLM training dataflows (PPO, GRPO, etc.) and reports 1.5×-20× throughput improvements over prior systems. Together, these advances demonstrate that relatively simple RL setups with rule-based correctness rewards can substantially enhance LLM reasoning when scaled effectively.

A.2 R1-Style Multimodal Reasoning

Large language models (LLMs) are usually *pre-trained* for next token generation and then *post-trained* to follow instructions and reason. The dominant post-training recipe combines supervised fine-tuning (SFT) with RLHF, where PPO is steered by a reward model built from human judgements [48]. Recently, DeepSeek-Math [55] and DeepSeek-R1 [11] skip the costly reward model training and replace PPO with *Group-Relative Policy Optimisation* (GRPO), achieving stronger reasoning at lower compute. While DeepSeek-Math/R1 targeted language-only models, a wave of concurrent studies now adapts similar idea to multimodal large language models (MLLMs) to improve reasoning abilities [27, 30].

Single-domain MLLM Reasoning The first wave of multimodal R1 work asks a focused question: *can GRPO turn a general-purpose MLLM into a domain specialist?* **Math reasoning** is explored by MAYE [39], R1-OneVision [70], and Multimodal-Open-R1 [26]. **Fine-grained/Open-vocabulary recognition** is addressed by Visual-RFT [35] and VLM-R1 [56]. R1-V [7] targets **visual counting and geometric reasoning**, and VisualThinker-R1-Zero [76] pioneers **spatial reasoning** without any SFT warm-up [52]. Collectively, these studies demonstrate that GRPO is *task-portable*: by pairing a verifiable reward with a modest domain-specific corpus, one can reliably lift MLLM reasoning within that domain.

Reward engineering and algorithmic variants. Once a core stack exists, researchers turn to *how* the policy is optimised, devising alternative rewards, KL schedules, and curricula. OThink-MR1 [36] introduces GRPO-D, an adaptive KL penalty that balances exploration and imitation for counting and geometry. NoisyRollout [34] injects image augmentations during roll-outs to strengthen mathematical reasoning, and ThinkLite-VL [64] shows that a *small but hard* subset can rival large-scale runs. FAST [65] further adapts the reasoning *length* to task difficulty on-the-fly.

Stability, efficiency and exploration strategies. As reward designs grow more sophisticated, training can suffer from vanishing gradients and sample inefficiency, motivating techniques that explicitly stabilise and accelerate GRPO. [8] proposes normalized length reward to mitigate instability caused by compeletion length. VL-Rethinker [62] combats gradient issues via selective sample replay and forced self-reflection, whereas LUFFY [69] blends off-policy imitation with on-policy exploration using regularised importance sampling.

Empirical analyses and large-scale deployments. A complementary thread conducts systematic ablations or scales GRPO to production-grade MLLMs, clarifying its real-world impact. VLAA-Thinking [6] argues that SFT alone induces "pseudo-reasoning" and that mixed-reward GRPO restores genuine logical skill; Limit-of-RLVR [74] notes that GRPO mainly lifts top-1 rather than top-k performance; and Perception-R1 [72] measures transfer to pure perception. In practice, Skywork-R1V [49] and Cosmos-Reason1 [45] add a GRPO stage to large production models, while OpenVLThinker [12] alternates SFT and GRPO over progressively harder datasets. The recipe is already migrating to video—Video-R1 [14], SEED-Bench-R1 [9] and VideoChat-R1—highlighting its modality-agnostic appeal. To track progress, new visual puzzle suites probe the logical depth of MLLMs [58, 25, 68].

Open problem. Despite the rapid proliferation of R1-style methods in MLLM Reasoning, *data mixture strategies*—how to allocate training resources across heterogeneous reasoning tasks—remain unexplored. Our work fills this gap and serves as a first step for future studies.

A.3 Data Mixture for LLM

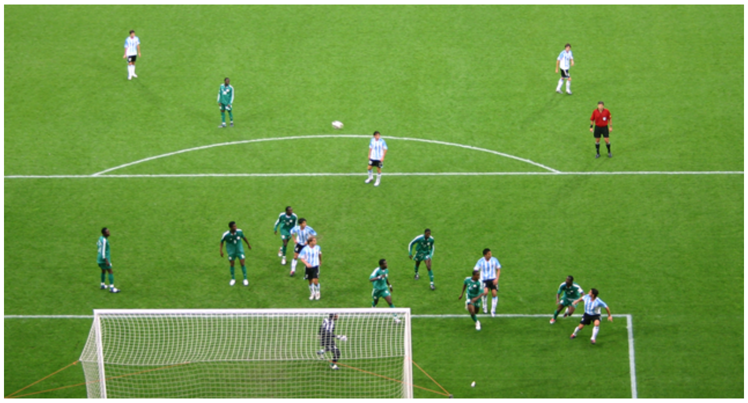
Existing approaches to mixing heterogeneous data sources for pretraining have largely fallen into two categories: language-centric up-sampling and model-guided data mixture. In the multilingual setting, BERT [13] up-samples under-represented languages by raising sampling probabilities with a temperature hyperparameter, but this often causes the lowest-resource languages to be repeated excessively. To address this imbalance problem, UniMax [10] introduces a heuristic that dynamically adjusts language weights to achieve a fairer distribution across languages. In parallel, methods such as data mixing laws [71] learn analytic functions over sample mixtures to predict model performance on unseen combinations without requiring full training runs. DoReMi [66] and RegMix [33] pursue a similar two-stage strategy: a small proxy model is first trained – via GroupDRO [53] in DoReMi or by evaluating multiple mixture configurations in RegMix – to identify optimal mixture weights, before resampling and training a full-scale model. Although effective, these proxy-based methods suffer from reduced efficiency due to multiple training passes, and the weights they learn often fail to generalize across different models. Online Data Mixing (ODM) [2] builds on DoReMi by updating data weights during full-model training, but still relies on the same underlying minimax estimation.

More recent findings suggest that even simple heuristics, such as token-count proportions [18], can outperform both manual and learned data mixture schemes, calling into question the necessity of complex weight estimation. Motivated by this insight, [18] proposes UtiliMax and Model Estimated Data Utility (MEDU), which blend lightweight utility estimates with model-informed adjustments to strike a balance between simplicity and adaptivity. While the aforementioned techniques focus primarily on monolingual or multilingual text corpora drawn from sources like Wikipedia, books, web text, and code, our study shifts attention to the domains of vision-language reasoning tasks. In these multimodal settings, dataset relationships are governed not only by raw sample frequency but also by the semantic and reasoning objectives unique to each task, necessitating novel strategies for sampling and weighting across diverse VLM reasoning benchmarks.

Table 2: Overall Performance Comparison. Seed experiments use digits to denote training datasets: **1**: COCO, **2**: LISA, **3**: GeoQAV, **4**: SAT, **5**: ScienceQA.

Experiment		**In Tests**			**Out Tests**				**In-Score**	**Out-Score**
		LISA	**SAT**	**ScienceQA**	**ChartQA**	**InfoVQA**	**MathVista**	**MMMU**		
	Base	0.1525	0.2479	0.0486	0.236	0.3144	0.391	0.3789	0.149	0.3059
Seed	*Single*									
	1	0.4015	0.5047	0.0258	0.41	0.5357	0.436	0.3811	0.3254	0.4589
	2	0.4433	0.4035	0.0253	0.3704	0.476	0.426	0.3922	0.318	0.4219
	3	0.0835	0.2547	0.4284	0.4936	0.5086	0.405	0.3989	0.2232	0.4753
	4	0.0551	0.5949	0.063	0.5476	0.5075	0.414	0.3722	0.199	0.4915
	5	0.0279	0.3786	0.7828	0.494	0.3787	0.426	0.3867	0.3274	0.4263
	Exclude-one									
	2345	0.4393	0.5897	0.7313	0.5644	0.5319	0.443	0.4022	0.559	0.5146
	1345	0.4064	0.5902	0.7288	0.4488	0.5471	0.429	0.4011	0.5432	0.4783
	1245	0.4728	0.6011	0.7283	0.5424	0.4875	0.437	0.4022	0.5767	0.4889
	1235	0.4727	0.4824	0.7313	0.498	0.4858	0.423	0.4122	0.5463	0.4721
	1234	0.4493	0.5856	0.4259	0.5108	0.5387	0.406	0.3978	0.4787	0.493
	All									
	12345	0.4778	0.5737	0.6991	0.4816	0.4681	0.435	0.41	0.5638	0.4609
Heuristic-based	*Step-averaged*									
	A_{in}	0.4687	0.5928	0.7476	0.5524	0.5111	0.434	0.4	0.5779	0.5008
	A_{out}	0.4616	0.5788	0.7179	0.488	0.5125	0.422	0.3989	0.5628	0.4772
	A_{bal}	0.4755	0.5892	0.7338	0.552	0.5179	0.441	0.4144	0.5763	0.5061
	Coli	0.4494	0.5773	0.7055	0.5304	0.4851	0.432	0.3978	0.5533	0.4825
	Norm	0.4573	0.5664	0.705	0.564	0.5083	0.467	0.4133	0.554	0.51
	2000-step									
	A_{in}	0.4415	0.5939	0.7581	0.522	0.5036	0.417	0.4133	0.5685	0.4867
	A_{out}	0.4075	0.5799	0.71	0.4788	0.5089	0.429	0.3911	0.5359	0.4726
	A_{bal}	0.4676	0.5877	0.7382	0.5548	0.5063	0.436	0.4078	0.5735	0.5011
	Coli	0.4678	0.5783	0.6083	0.4076	0.5035	0.422	0.4089	0.5354	0.4471
	Norm	0.4732	0.5762	0.7372	0.5764	0.5156	0.449	0.4022	0.5728	0.5133
Heuristic-based (No 1)	*Step-averaged*									
	A_{in}	0.4498	0.402	0.7129	0.5456	0.5065	0.428	0.4089	0.5095	0.497
	A_{out}	0.0873	0.5856	0.7015	0.4692	0.4932	0.421	0.4022	0.3869	0.4635
	A_{bal}	0.4666	0.5892	0.7447	0.5572	0.5307	0.427	0.3967	0.5752	0.5088
	Coli	0.448	0.5923	0.704	0.4836	0.5002	0.402	0.3933	0.5562	0.4674
	Norm	0.4662	0.5794	0.711	0.5072	0.507	0.42	0.41	0.5632	0.4829
	2000-step									
	A_{in}	0.4257	0.4414	0.7204	0.5096	0.4308	0.403	0.3956	0.5108	0.4499
	A_{out}	0.0556	0.5731	0.6896	0.488	0.5053	0.415	0.4022	0.3657	0.4739
	A_{bal}	0.4637	0.5529	0.7248	0.4904	0.4847	0.429	0.3922	0.5589	0.4674
	Coli	0.4815	0.5897	0.7105	0.514	0.4978	0.42	0.4067	0.5728	0.4812
	Norm	0.4065	0.5892	0.7169	0.5596	0.5158	0.424	0.3944	0.5398	0.5031
Model-based	001	0.2274	0.5871	0.7387	0.5196	0.5246	0.453	0.3911	0.4623	0.4962
	002	0.2674	0.5762	0.7288	0.5536	0.525	0.438	0.3956	0.4752	0.5067
	003	0.2225	0.5747	0.7427	0.514	0.5005	0.446	0.4044	0.4579	0.4856
	004	0.438	0.597	0.7154	0.5344	0.5016	0.424	0.3844	0.556	0.4876
	005	0.1219	0.6006	0.7313	0.4844	0.5256	0.414	0.4178	0.415	0.4823
	006	0.4293	0.5949	0.7402	0.5528	0.5126	0.427	0.3867	0.5582	0.4989
	007	0.2382	0.5669	0.7234	0.5412	0.5269	0.443	0.4033	0.4578	0.5048
	008	0.3843	0.5887	0.7214	0.516	0.5067	0.448	0.3989	0.5306	0.4883
	009	0.3955	0.5773	0.7343	0.5648	0.5258	0.434	0.3789	0.5363	0.5082
	010	0.3925	0.5742	0.7338	0.5668	0.5184	0.446	0.4	0.534	0.5104

Figure 8: Comparison of Grounding Question-Answer Pairs **With** and **Without** Reasoning.



Question: Output the bounding box in the image corresponding to the instruction: In a football match, which position primarily focuses on guarding the goal and stopping the opposing team from scoring?
Answer: [422, 781, 464, 926]

Question: Output the bounding box in the image corresponding to the instruction: In a football match, which position primarily focuses on guarding the goal and stopping the opposing team from scoring?
Output the thinking process in <think> </think> and final answer in <answer> </answer> tags.The output answer format should be as follows: <think> ... </think> <answer>['Position': [x1, y1, x2, y2], 'Confidence': number, ...]</answer> Please strictly follow the format.
Answer: *<think> ... </think> <answer>['Position': [422, 781, 464, 926], 'Confidence': 1]</answer>*

Algorithm 1 Heuristic-Alpha: Balanced, In-optimized and Out-optimized

```
Require:
    $\mathcal{D} = \{1, ..., m\}$: Set of $m$ dataset indices
    $R$: Set of performance records from experiments
    $\alpha_{\text{single}} \in [0, 1]$: Scaling factor for single datamix baselines
    $\alpha \in [0, 1]$: Trade-off parameter balancing In and Out set performance
Ensure: Dataset weights $\{w_i\}_{i \in \mathcal{D}}$ where $\sum_{i \in \mathcal{D}} w_i = 1$
 1: $S^{\text{in}} \leftarrow [0]_{1..m}$                                ▷ Initialize In Score sums
 2: $S^{\text{out}} \leftarrow [0]_{1..m}$                               ▷ Initialize Out Score sums
 3: for each record $\{s_r^{\text{in}}, s_r^{\text{out}}\} \in R$ do
 4:     Identify datasets $D_r \subseteq \mathcal{D}$ used in recipe $r$
 5:     if $|D_r| = 1$ then                                            ▷ Single-dataset
 6:         $s_r^{\text{in}} \leftarrow \alpha_{\text{single}} \cdot s_r^{\text{in}}$
 7:         $s_r^{\text{out}} \leftarrow \alpha_{\text{single}} \cdot s_r^{\text{out}}$
 8:     end if
 9:     for each dataset $i \in D_r$ do
10:         $S^{\text{in}}[i] \leftarrow S^{\text{in}}[i] + s_r^{\text{in}}$
11:         $S^{\text{out}}[i] \leftarrow S^{\text{out}}[i] + s_r^{\text{out}}$
12:     end for
13: end for
14: Min-max normalization:
15: $\hat{S}^{\text{in}} \leftarrow \text{MinMax}(S^{\text{in}})$        ▷ $\hat{S}_i^{\text{in}} = \frac{S_i^{\text{in}} - \min_j S_j^{\text{in}}}{\max_j S_j^{\text{in}} - \min_j S_j^{\text{in}}}$
16: $\hat{S}^{\text{out}} \leftarrow \text{MinMax}(S^{\text{out}})$      ▷ $\hat{S}_i^{\text{out}} = \frac{S_i^{\text{out}} - \min_j S_j^{\text{out}}}{\max_j S_j^{\text{out}} - \min_j S_j^{\text{out}}}$
17: Combine scores with trade-off parameter:
18: for each dataset $i \in \mathcal{D}$ do
19:     $C_i \leftarrow \alpha \cdot \hat{S}^{\text{in}}[i] + (1 - \alpha) \cdot \hat{S}^{\text{out}}[i]$
20: end for
21: Normalize to obtain final weights:
22: $C_{\text{sum}} \leftarrow \sum_{i \in \mathcal{D}} C_i$
23: for each dataset $i \in \mathcal{D}$ do
24:     $w_i \leftarrow C_i / C_{\text{sum}}$
25: end for
26: return $\{w_i\}_{i \in \mathcal{D}}$
```

Algorithm 2 Heuristic-Colinearity; Variance Inflation Factor (VIF) quantifies multicollinearity.

```
Require:
    $\mathcal{D} = \{1, ..., m\}$: Set of $m$ dataset indices
    $R$: Set of performance records from experiments
    $\lambda \in \mathbb{R}_+$: Ridge regularization parameter (default $10^{-3}$)
Ensure: Dataset weights $\{w_i\}_{i \in \mathcal{D}}$ where $\sum_{i \in \mathcal{D}} w_i = 1$
 1: $X \leftarrow [\,]$                                        ▷ Initialize design matrix
 2: $y \leftarrow [\,]$                                        ▷ Initialize target vector
 3: for each record $\{s_r^{\text{in}}, s_r^{\text{out}}\} \in R$ do
 4:     Identify datasets $D_r \subseteq \mathcal{D}$ used in recipe $r$
 5:     $x_r \leftarrow [0]_{1..m}$                            ▷ Initialize feature vector for this record
 6:     for each dataset $i \in D_r$ do
 7:         $x_r[i] \leftarrow 1$                              ▷ Set indicator to 1 for datasets used in recipe
 8:     end for
 9:     $X \leftarrow X \cup \{x_r\}$                          ▷ Add feature vector to design matrix
10:     $y \leftarrow y \cup \{s_r^{\text{out}}\}$            ▷ Add score to target vector
11: end for
12: Fit ridge regression without intercept:
13: $\beta \leftarrow \arg\min_\beta \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$    ▷ Ridge regression coefficients
14: Compute Variance Inflation Factor (VIF) correction:
15: $I_m \leftarrow$ Identity matrix of size $m \times m$     ▷ $m \times m$ identity matrix
16: $X^T X_{\text{reg}} \leftarrow X^T X + \lambda \cdot I_m$  ▷ Regularized Gram matrix
17: $(X^T X_{\text{reg}})^{-1} \leftarrow \text{Inverse}(X^T X_{\text{reg}})$    ▷ Inverse of Gram matrix
18: VIF $\leftarrow \text{diag}((X^T X_{\text{reg}})^{-1})$   ▷ Diagonal elements give VIF
19: Adjust coefficients using VIF:
20: for each dataset $i \in \mathcal{D}$ do
21:     $s_i \leftarrow \beta_i / \text{VIF}_i$                ▷ Adjust coefficient by its variance inflation
22: end for
23: Apply non-negativity constraint:
24: for each dataset $i \in \mathcal{D}$ do
25:     $s_i \leftarrow \max(0, s_i)$                          ▷ Replace negative values with zero
26: end for
27: Normalize to obtain final weights:
28: $s_{\text{sum}} \leftarrow \sum_{i \in \mathcal{D}} s_i$
29: for each dataset $i \in \mathcal{D}$ do
30:     $w_i \leftarrow s_i / s_{\text{sum}}$
31: end for
32: return $\{w_i\}_{i \in \mathcal{D}}$
```

Algorithm 3 Heuristic-Normalize: Leave-One-Out Dataset Weighting

```
Require:
    $\mathcal{D} = \{1, ..., m\}$: Set of $m$ dataset indices
    $R$: Set of performance records from experiments
    $C$: Set of leave-one-out experiments
Ensure: Dataset weights $\{w_i\}_{i \in \mathcal{D}}$ where $\sum_{i \in \mathcal{D}} w_i = 1$
 1: $S_C \leftarrow [\,]$                                  ▷ Initialize scores for combinations
 2: for each record $\{s_r^{\text{in}}, s_r^{\text{out}}\} \in R$ do
 3:     Extract combination ID $c_r$ from leading digits in $r$
 4:     Extract out score $s_r^{\text{out}}$
 5:     if $c_r \in C$ then                                ▷ If this is a leave-one-out combination
 6:         $S_C \leftarrow S_C \cup \{(c_r, s_r^{\text{out}})\}$          ▷ Store combo and score
 7:     end if
 8: end for
 9: Normalize out scores:
10: $s_{\max} \leftarrow \max_{(c_r, s_r) \in S_C} s_r$                      ▷ Maximum out score
11: $s_{\min} \leftarrow \min_{(c_r, s_r) \in S_C} s_r$                      ▷ Minimum out score
12: for each $(c_r, s_r) \in S_C$ do
13:     $\hat{s}_r \leftarrow \frac{s_r - s_{\min}}{s_{\max} - s_{\min}}$                ▷ Min-max normalization
14:     $s'_r \leftarrow 0.2 - (0.1 \cdot \hat{s}_r)$       ▷ Transform: higher score = lower weight
15: end for
16: Derive dataset weights from leave-one-out scores:
17: $w_{\text{raw}} \leftarrow [0]_{1..m}$                                  ▷ Initialize raw weights
18: for $i \in \{1..m\}$ do
19:     $w_{\text{raw}}[i] \leftarrow s'_r$ where $c_r$ is the combination missing dataset $i$
20: end for
21: Normalize to obtain final weights:
22: $w_{\text{sum}} \leftarrow \sum_{i=1}^{m} w_{\text{raw}}[i]$
23: for $i \in \{1..m\}$ do
24:     $w_i \leftarrow w_{\text{raw}}[i] / w_{\text{sum}}$
25: end for
26: return $\{w_i\}_{i \in \mathcal{D}}$
```

Algorithm 4 QuadSurface: Quadratic Response Surface Optimization

```
Require:
    $\mathcal{D} = \{1, ..., m\}$: Set of $m$ dataset indices
    $R$: Set of performance records from experiments
    $n_{\text{samples}}$: Number of candidate points to evaluate
    $k$: Number of top mixtures to return
Ensure: Top $k$ dataset mixtures $\{w_i\}_{i \in \mathcal{D}}$ optimized for performance
 1: Data preparation:
 2: $X \leftarrow [\,]$                                    ▷ Initialize matrix of mixture weights
 3: $y \leftarrow [\,]$                                    ▷ Initialize vector of performance scores
 4: for each record $r \in R$ do
 5:     Extract weight vector $w_r$ and performance score $s_r$
 6:     $X \leftarrow X \cup \{w_r\}$
 7:     $y \leftarrow y \cup \{s_r\}$
 8: end for
 9: Cross-validation splits:
10: Perform $n_{\text{splits}} = 5$ random train-test splits of $(X, y)$      ▷ $80\%$ train, $20\%$ test
11: for each split $i$ do
12:     $(X_{\text{train}}, y_{\text{train}}), (X_{\text{test}}, y_{\text{test}}) \leftarrow \text{Split}_i(X, y)$
13:     Fit quadratic response surface model:
14:     $F_{\text{train}} \leftarrow [1, X_{\text{train}}, X_{\text{train}} \otimes X_{\text{train}}]$        ▷ Design matrix with quadratic terms
15:     $\beta_i \leftarrow \arg\min_\beta \|y_{\text{train}} - F_{\text{train}}\beta\|_2^2$            ▷ Ordinary least squares
16:     $(b_i, W_i, Q_i) \leftarrow \text{Extract}(\beta_i)$       ▷ Extract intercept, linear, and quadratic terms
17:     Evaluate model:
18:     $\hat{y}_{\text{train}} \leftarrow b_i + X_{\text{train}} W_i + (X_{\text{train}} \otimes X_{\text{train}}) Q_i$
19:     $\hat{y}_{\text{test}} \leftarrow b_i + X_{\text{test}} W_i + (X_{\text{test}} \otimes X_{\text{test}}) Q_i$
20:     $R^2_{\text{train},i}, R^2_{\text{test},i} \leftarrow \text{ComputeR}^2(\hat{y}_{\text{train}}, y_{\text{train}}, \hat{y}_{\text{test}}, y_{\text{test}})$
21: end for
22: Select best model:
23: $i^* \leftarrow \arg\max_i R^2_{\text{test},i}$                  ▷ Choose model with best test performance
24: $(b^*, W^*, Q^*) \leftarrow (b_{i^*}, W_{i^*}, Q_{i^*})$
25: Fit GMM to observed mixtures:
26: GMM $\leftarrow$ FitGaussianMixture$(X, n_{\text{components}} = 1)$        ▷ Fit single-component GMM
27: Generate candidate mixtures via GMM:
28: $X_{\text{raw}} \leftarrow \text{SampleFromGMM}(\text{GMM}, n_{\text{samples}})$
29: $X_{\text{valid}} \leftarrow \{x \in X_{\text{raw}} | x_i \geq 0 \text{ for all } i\}$              ▷ Filter non-negative points
30: $X_{\text{candidates}} \leftarrow \{x / \sum_i x_i | x \in X_{\text{valid}}\}$                 ▷ Normalize to sum to 1
31: Predict performance for all candidates:
32: $\hat{y}_{\text{candidates}} \leftarrow b^* + X_{\text{candidates}} W^* + (X_{\text{candidates}} \otimes X_{\text{candidates}}) Q^*$
33: Select top-performing mixtures:
34: $\text{indices}_{\text{top}} \leftarrow \text{Argsort}(\hat{y}_{\text{candidates}})[-k:]$                 ▷ Indices of top $k$ scores
35: $X_{\text{top}} \leftarrow X_{\text{candidates}}[\text{indices}_{\text{top}}]$                         ▷ Top $k$ mixtures
36: return $X_{\text{top}}$
```