

A. Results on Additional Model Pairs

We conducted experiments on additional model pairs, summarized in Table 9, which highlights the cumulative performance gains across tasks for six different model pairs. The model pairs include: (1) merging LLaVA-OneVision [12] into Qwen2-VL [24] (Table 1), (2) merging LLaVA-v1.5 [14] into CogVLM [25] (Table 2), (3) merging mPLUG-Owl2 into LLaVA-v1.5, (4) merging LLaVA-v1.5 into mPLUG-Owl2 [28] (Table 11), (5) merging CogVLM into mPLUG-Owl2 (Table 12) and (6) merging mPLUG-Owl2 into CogVLM (Table 13). The performance gain for each task is computed as the difference between the performance of our method (or baselines) and the average performance of the two original models, with positive values indicating an improvement. In Table 9, the SUM column presents the total performance gains across all tasks, where AdaMMS outperforms all baselines, achieving +91.92 performance gain, and consistently ranks among the top two in performance gains across all benchmarks. It is noteworthy that on GQA [9] and VizWiz [7] benchmarks in Table 11 and Table 12, all model merging methods experience a performance drop. We attribute this decline to the significant performance gap between the original models on these benchmarks. In these scenarios, AdaMMS demonstrates the smallest performance decrease among them. In Table 11, Table 12 and Table 13, AdaMMS obtains the second best result in the sum of all benchmarks, with a small gap compared to the best baseline.

To investigate the effect of altering base models on performances, we analyze experiments on merging the same model pair with different base models. For the model pair of mPLUG-Owl2 and CogVLM, results in Table 12 use mPLUG-Owl2 as the base model, and results in Table 13 use CogVLM as the base model. On benchmarks where the original models exhibit a significant performance gap, such as OCRBench [15] and TextVQA [18], model merging methods, including AdaMMS, achieve only marginal performance improvements. In contrast, on benchmarks where the original models have comparable performance, AdaMMS consistently enhances the base model’s performance (with the exception of GQA [9] for the mPLUG-Owl2 architecture), irrespective of the choice of base model. Notably, even when merging a weaker model into a stronger one for a specific task, AdaMMS can sometimes boost the stronger model’s performance. For instance, this effect is observed on SEEDBench [11], OKVQA [16], and GQA [9] in Table 13. These results highlight that our model merging technique can further optimize the performance of a strong model, even when another model demonstrates weaker performance on the same task.

Additionally, to demonstrate the effectiveness of our method on larger models, we conducted experiments on Cambrian and Yi-VL with 34B language model size. Ta-

ble 6 shows that AdaMMS also merges the abilities in larger MLLMs effectively.

Model	OCRBench	MME
Cambrian(base)	58.70	72.50
Yi-VL	29.70	73.65
AVG	44.20	73.08
AdaMMS	59.20	74.07

Table 6. Results on merging Yi-VL into Cambrian.

B. Implementation Details of AdaMMS

The implementation details of AdaMMS are as follows:

Mapping In this step, we identify parameters in the language models that account for additional weights. For CogVLM [25], all weights within the visual experts in the attention mechanism (including the QKV matrix and the FFN of the visual expert) are treated as additional weights. For mPLUG-Owl2 [28], vision representation weights within the Modality-Adaptive Modules (such as the decoupled vision layer-norm and KV matrix) are considered additional weights. For different vision encoders, the vision encoder weights of the base model are retained as the final weights after merging, regardless of the vision encoder in the other model.

Merging During this step, we first merge the weights in the language model of the base model. If the weights are not classified as additional weights in the Mapping step, they are merged using linear interpolation or other baseline merging techniques. For weights categorized as additional weights, we check whether the other model has duplicated the same weights. Based on this, we (1) merge the weights if duplicates exist, or (2) retain the original weights in the base model if no duplicates are found.

Searching In the final step, we randomly select a subset of 100 test inputs to determine the optimal α . For each α candidate, we generate model responses for the selected inputs. To select the best α , we apply the Exact Match metric for the total difference score: for each input, if the merged model’s response with a given α matches the response with adjacent α values, the difference score is 0; otherwise, it is 1. The total difference score is the sum of scores across all inputs in the subset. The α with the lowest total difference score is selected as the final choice. Note that the small subset of 100 inputs is randomly sampled using the method in LMMs-Eval framework [31]. We have repeated the sampling process to ensure that the randomness in sampling does not affect the performance of our method.

C. Evaluation Details

We utilize LMMs-Eval [31] and VLMEvalKit [3], two open-source evaluation frameworks for MLLMs, to as-

sess our models. Specifically, for evaluating MMMU [30], MME [5], SEEDBench [11], OCRBench [15], and TextVQA [18] within the Qwen2-VL [24] architecture, we use the VLMEvalKit framework, while LMMs-Eval is employed for the others. To ensure consistency with the reported results for LLaVA and mPLUG-Owl2 on OK-VQA [16], we adapted the prompt template in the evaluation framework, as detailed in Table 7. Other prompt templates remains the same in the evaluation frameworks.

D. Comparing Supervised and Unsupervised

We compared AdaMMS with baseline merging methods with supervised hyper-parameter selection. Due to the absence of separate test sets, we trained the supervised baseline on either a subset or the entirety of the evaluation set. **This implies that the supervised baseline was in a more favorable position compared to our method, as our method does not have access to the groudtruth labels.** Table 8 shows that AdaMMS still outperforms it, indicating the superiority of our unsupervised method.

E. Intermediate Results in Searching

We present an example of the intermediate results during the selection of α . As shown in Figure 14, AdaMMS effectively identifies a near-optimal α , achieving performance close to the best possible outcome. Specifically, our unsupervised hyper-parameter selection method successfully chooses the optimal α candidate in half of the benchmarks and maintains a deviation of no more than 0.2 from the best α in the remaining cases.

Figure 5 illustrates the relationship between model performance and generation consistency across MMMU, MME, SeedBench, and OCRBench when merging LLaVA-OneVision into Qwen2-VL. The observed trends validate our approach in the search step, where model performance is approximated using generation consistency without relying on labeled data. Notably, for these tasks, the α selected by our method corresponding to the highest generation consistency deviates from the α achieving the best performance by no more than 0.1, showing that our hyper-parameter selection method achieves near-optimal performance.

Framework	Base Model	Prompt
LMMs-Eval	LLaVA	Answer the question using a single word or phrase.
	mPLUG-Owl2	

Table 7. Altered prompt for evaluation on OK-VQA.

F. Supplementary Proof

We provide the following proof as the theoretical justification for relationship between generation consistency correlates and model performance.

Proof. Using the notation in Section 3.3, for an arbitrary task t_i , let $S_{t_i}(\alpha)$ be the ratio of correct answer at position α , and $D_{t_i}(\alpha; \alpha^-)$ be the ratio of the difference in generated responses between position α and its adjacent candidate α^- . Since the difference in $S_{t_i}(\alpha)$ is only influenced by the subset of generated responses where the correctness status changes (i.e., transitions between correct and incorrect), we have $|S_{t_i}(\alpha) - S_{t_i}(\alpha^-)| \leq D_{t_i}(\alpha; \alpha^-)$. For the same reason with α^+ , we can prove $|S_{t_i}(\alpha) - S_{t_i}(\alpha^-)| + |S_{t_i}(\alpha) - S_{t_i}(\alpha^+)| \leq 2D_{t_i}(\alpha; \alpha^-, \alpha^+)$. Therefore, a higher generation consistency with small $D_{t_i}(\alpha; \alpha^-, \alpha^+)$ implies a higher model performance $S_{t_i}(\alpha)$, due to its convexity.

G. Experimental Results in Granularity for α

Figure 4 presents the result of AdaMMS at different granularities of α . The point in stars indicates the best α by our unsupervised parameter selection method. The result shows that these granularities in $\{0.02, 0.05, 0.10\}$ behave similarly in terms of the final performance, indicating the robustness of AdaMMS. Therefore, in practice we choose a larger α so that we have fewer α candidates, which reduces the computation cost.

Model	MMU _{val}	MME _{sum}	SeedBench _{all}	OCRBench	TextVQA _{val}	OKVQA	GQA	VizWiz _{val}	Sum	Diff
AdaMMS	34.90	69.09	64.12	55.70	76.90	61.11	60.12	37.27	459.21	+31.23
Ties-Merging	34.00	57.29	38.97	55.00	59.73	40.31	51.97	24.36	361.63	-66.35
Ties-Merging (supervised with 100 eval. samples)	37.20	57.29	63.12	55.90	76.50	61.45	55.81	37.98	445.25	+17.27
Ties-Merging (supervised with all eval. data)	37.20	63.96	65.43	55.90	76.55	61.45	57.99	38.21	456.69	+28.71

Table 8. AdaMMS and Ties-Merging with *supervised* hyper-parameter selection via validation set.

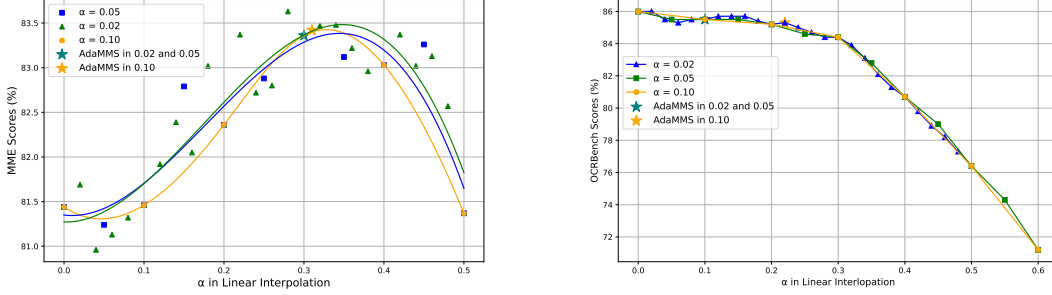


Figure 4. Results on linear interpolation at different granularities of α when merging LLaVA-OneVision-7B into Qwen2-VL-7B-7B. (Left: MME, Right: OCRBench)

Model	MMU _{val}	MME _{sum}	SeedBench _{all}	OCRBench	TextVQA _{val}	OKVQA	GQA	VizWiz _{val}	SUM	Top2
Task Arithmetic	<u>13.21</u>	21.53	14.54	-1.80	-3.74	13.88	-2.95	<u>-7.29</u>	47.45	7
Ties-Merging	-3.32	-24.94	-27.34	1.20	-31.59	-23.23	-29.70	-29.20	-168.05	0
DARE-Linear	8.15	-2.35	10.58	-12.3	-19.23	5.41	-12.76	-21.09	-43.53	0
DARE-Ties	-14.83	-60.56	-6.96	-47.50	-47.12	-31.08	-26.32	-32.45	-266.76	0
MetaGPT	1.44	-2.93	-4.02	15.30	<u>0.37</u>	-6.75	-23.69	-16.73	-36.94	2
AdaMMS	17.68	<u>17.48</u>	<u>12.02</u>	<u>13.60</u>	18.43	<u>13.40</u>	1.40	-2.14	91.92	9

Table 9. Results of the performance gain sum among six model pairs reported in our paper, as described in Appendix A. The performance gain for each task is computed as the difference between the performance of our method (or baselines) and the average performance of the two original models, with positive values indicating an improvement.

Model	Unsupervised	MMU _{val}	MME _{sum}	SeedBench _{all}	OCRBench	TextVQA _{val}	OKVQA	GQA	VizWiz _{val}	SUM	Top2
Original Models											
LLaVA(base)		35.10	66.68	60.52	31.30	46.04	53.42	61.94	54.29	409.29	
mPLUG-Owl2		34.90	62.80	59.41	34.10	55.13	60.98	56.11	32.07	395.50	
Baselines											
Task Arithmetic	×	36.00 (+1.00)	67.00 (+2.26)	61.45 (+1.48)	30.40 (-2.30)	45.75 (-4.84)	56.79 (-0.41)	59.68 (+0.66)	56.49 (+13.31)	413.56 (+11.17)	5
Ties-Merging	×	33.60 (-1.40)	62.14 (-2.60)	60.32 (+0.35)	30.10 (-2.60)	42.85 (-7.73)	52.46 (-4.74)	58.37 (-0.66)	51.30 (+8.12)	391.14 (-11.25)	0
DARE-Linear	×	36.00 (+1.00)	67.00 (+2.26)	61.41 (+1.44)	30.70 (-2.00)	45.84 (-4.74)	57.06 (-0.14)	59.56 (+0.54)	55.90 (+12.72)	413.47 (+11.08)	2
DARE-Ties	×	31.70 (-3.30)	59.81 (-4.93)	60.06 (+0.09)	29.50 (-3.20)	41.90 (-8.69)	46.00 (-11.20)	57.51 (-1.52)	53.27 (+10.09)	379.75 (-22.64)	0
MetaGPT	✓	35.30 (+0.30)	67.62 (+2.88)	<u>61.46 (+1.49)</u>	<u>30.60 (-2.10)</u>	45.80 (-4.79)	56.54 (-0.66)	59.41 (+0.38)	56.66 (+13.48)	413.39 (+11.00)	4
Our Method											
AdaMMS	✓	38.30 (+3.30)	<u>67.01 (+2.27)</u>	61.82 (+1.85)	31.00 (-1.70)	46.49 (-4.09)	55.60 (-1.60)	61.81 (+2.79)	54.64 (+11.46)	416.67 (+14.28)	7

Table 10. Results on merging mPLUG-Owl2-7B into LLaVA-v1.5-7B.

Model	Unsupervised	MMU _{val}	MME _{sum}	SeedBench _{all}	OCRBench	TextVQA _{val}	OKVQA	GQA	VizWiz _{val}	SUM	Top2
Original Models											
mPLUG-Owl2(base)		34.90	62.80	59.41	34.10	55.13	60.98	56.11	32.07	395.50	
LLaVA		35.10	66.68	60.52	31.30	46.04	53.42	61.94	54.29	409.29	
Baselines											
Task Arithmetic	×	<u>36.90(+1.90)</u>	63.17(-1.57)	60.44(+0.47)	33.00(+0.30)	55.40(+4.81)	63.87(+6.67)	56.97(-2.06)	33.70(-9.48)	403.45(+1.05)	4
Ties-Merging	×	<u>36.90(+1.90)</u>	64.20(-0.54)	60.13(+0.16)	34.40(+1.70)	54.50(+3.91)	62.92(+5.72)	57.55(-1.48)	33.18(-10.00)	403.78(+1.38)	4
DARE-Linear	×	<u>36.20(+1.20)</u>	62.99(-1.75)	<u>60.41(+0.44)</u>	32.60(-0.10)	55.15(+4.56)	<u>63.47(+6.27)</u>	56.73(-2.30)	33.35(-9.83)	400.90(-1.50)	2
DARE-Ties	×	<u>35.30(+0.30)</u>	60.37(-4.37)	58.36(-1.61)	32.00(-0.70)	51.65(+1.06)	58.08(+0.88)	55.57(-3.46)	31.03(-12.15)	382.36(-20.04)	0
MetaGPT	✓	<u>36.00(+1.00)</u>	<u>64.24(-0.50)</u>	60.23(+0.26)	<u>33.90(+1.20)</u>	<u>55.83(+5.24)</u>	62.88(+5.68)	56.53(-2.50)	33.35(-9.83)	402.96(+0.56)	3
Our Method											
AdaMMS	✓	37.60(+2.60)	64.61(-0.13)	60.02(+0.05)	32.20(-0.50)	55.84(+5.25)	63.13(+5.93)	<u>56.98(-2.05)</u>	<u>33.39(-9.79)</u>	<u>403.77(+1.37)</u>	6

Table 11. Results on merging LLaVA-v1.5-7B into mPLUG-Owl2-7B.

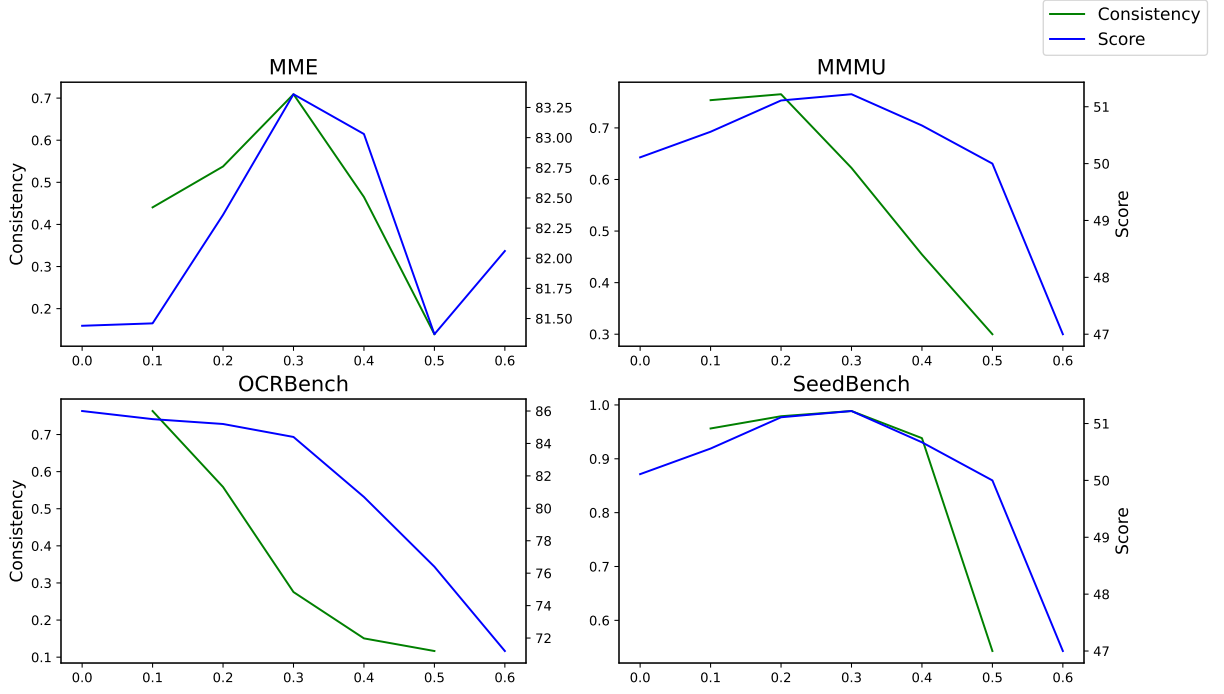


Figure 5. Generation consistency and model performance (score) for MME, MMMU, OCRBench and SeedBench when merging LLaVA-OneVision-7B into Qwen2-VL-7B. Generation consistency is calculated as the reciprocal of the sum of different responses from models with adjacent α candidates. The horizontal axis is the α of the linear interpolation.

Model	Unsupervised	MMMU _{val}	MME _{sum}	SeedBench _{all}	OCRBench	TextVQA _{val}	OKVQA	GQA	VizWiz _{val}	SUM	Top2
Original Models											
mPLUG-Owl2(base)		34.90	62.80	59.41	34.10	55.13	60.98	56.11	32.07	395.50	
CogVLM		34.80	59.23	61.22	56.50	77.57	60.82	59.43	37.09	446.66	
Baselines											
Task Arithmetic	×	38.80(+3.95)	64.65(+3.63)	60.85(+0.53)	31.50(-13.80)	56.99(-9.36)	60.93(+0.03)	54.44(-3.33)	32.76(-1.82)	400.92(-20.16)	8
Ties-Merging	×	27.9(-6.95)	48.96(-12.06)	52.32(-8.00)	24.30(-21.00)	42.10(-24.25)	54.15(-6.75)	43.02(-14.75)	27.56(-7.02)	320.31(-100.77)	0
DARE-Linear	×	37.60(+2.75)	62.44(+1.42)	59.81(-0.51)	30.90(-14.40)	56.41(-9.94)	61.07(+0.17)	54.11(-3.66)	32.42(-2.16)	394.76(-26.32)	1
DARE-Ties	×	32.00(-2.85)	57.90(-3.12)	57.62(-2.70)	24.10(-21.20)	43.84(-22.51)	51.56(-9.34)	52.04(-5.73)	25.67(-8.91)	344.73(-76.35)	0
MetaGPT	✓	31.30(-3.55)	56.81(-4.21)	50.81(-9.51)	29.30(-16.00)	37.96(-28.39)	43.02(-17.88)	34.12(-23.65)	15.84(-18.74)	299.16(-121.92)	0
Our Method											
AdaMMS	✓	39.10(+4.25)	64.65(+3.63)	60.16(-0.16)	30.60(-14.70)	55.88(-10.47)	62.11(+1.21)	55.61(-2.16)	32.69(+1.89)	400.80(-20.28)	9

Table 12. Results on merging CogVLM-7B into mPLUG-Owl2-7B.

Model	Unsupervised	MMMU _{val}	MME _{sum}	SeedBench _{all}	OCRBench	TextVQA _{val}	OKVQA	GQA	VizWiz _{val}	SUM	Top2
Original Models											
CogVLM(base)		34.80	59.23	61.22	56.50	77.57	60.82	59.43	37.09	446.66	
mPLUG-OWI2		34.90	62.80	59.41	34.10	55.13	60.98	56.11	32.07	395.50	
Baselines											
Task Arithmetic	×	38.30(+3.45)	72.11(+11.09)	67.24(+6.92)	51.90(+6.60)	70.68(+4.33)	63.59(+2.69)	59.98(+2.21)	37.16(+2.58)	460.96(+39.88)	7
Ties-Merging	×	34.60(-0.25)	53.54(-7.48)	61.73(+1.41)	50.70(+5.40)	66.65(+0.30)	58.19(-2.71)	52.66(-5.11)	33.92(-0.66)	411.99(-9.09)	0
DARE-Linear	×	39.20(+4.35)	68.80(+7.78)	66.66(+6.34)	50.90(+5.60)	70.35(+4.00)	63.26(+2.36)	58.80(+1.03)	36.80(+2.22)	454.77(+33.69)	3
DARE-Ties	×	29.00(-5.85)	53.89(-7.12)	61.61(+1.30)	42.90(-2.40)	63.46(-2.89)	54.34(-6.56)	55.54(-2.23)	33.96(-0.62)	394.70(-26.38)	0
MetaGPT	✓	34.90(+0.05)	61.54(+0.52)	62.93(+2.62)	57.30(+12.00)	77.18(+10.83)	61.55(+0.65)	59.93(+2.16)	37.15(+2.57)	452.48(+31.40)	2
Our Method											
AdaMMS	✓	38.10(+3.25)	62.48(+1.46)	66.79(+6.48)	56.30(+11.00)	76.89(+10.54)	61.71(+0.81)	59.96(+2.19)	37.33(+2.75)	459.56(+38.48)	6

Table 13. Results on merging mPLUG-Owl2-7B into CogVLM-7B.

Model	MMU _{val}	MME _{sum}	SeedBench _{all}	OCRBench	TextVQA _{val}	OKVQA	GQA	VizWiz _{val}	SUM
Original Models									
Qwen2-VL(base)	50.11	81.44	75.85	86.00	84.12	51.43	61.80	68.32	559.07
LLaVA-OneVision	43.44	77.04	75.44	69.60	78.47	49.57	59.84	60.97	514.37
AVG	46.78	79.24	75.65	77.80	81.30	50.50	60.82	64.65	536.72
Linear Interpolation									
α -0.1	50.56	81.46	76.20	85.50	83.41	53.56	62.02	68.40	561.11
α -0.2	51.11	82.36	76.23	85.20	81.74	54.76	62.05	67.12	560.57
α -0.3	51.22	83.36	76.34	84.40	78.43	52.03	61.44	63.91	551.13
α -0.4	50.67	83.03	76.06	80.70	71.66	49.83	60.09	58.43	530.47
α -0.5	50.00	81.37	75.63	76.40	59.13	44.96	55.53	52.60	495.62
α -0.6	47.00	82.06	74.76	71.30	39.37	40.31	54.11	46.39	455.30
Our Method									
AdaMMS	51.11	83.36	76.20	85.50	83.41	53.56	62.02	68.40	563.56
Selected α	0.2	0.3	0.1	0.1	0.1	0.1	0.1	0.1	-
Distance with the best α	0.1	0	0.2	0	0	0.1	0.1	0	-

Table 14. Intermediate results on different α candidates in the linear interpolation of AdaMMS, and the α selected by our unsupervised hyper-parameter selection method on merging LLaVA-OneVision-7B into Qwen2-VL-7B. AVG indicates the average performance of the two original models.