We sincerely appreciate the constructive feedback provided by the reviewer, which is invaluable for enhancing the quality of our paper. Below are our responses to the raised concerns. We also commit to accurately reflecting these points in the revised manuscript.

W1:

To address the reviewer's concern, we would like to clarify that incorporating additional new compoents did not significantly raise the training costs. (In fact, our method significantly improved the training efficiency compared to LLaVA-Next.)

(1) The VQ-VAE model is employed solely for re-annotating the open-source dataset, which involves collecting the necessary labels for cognitive alignment. It does not directly participate in the training process, thereby not introducing any additional computational overhead.

(2) As for SA-Perceiver, which comprising four $\mathbb{R}^{1024\times1024}$ linear layers and one $\mathbb{R}^{1024\times4096}$ linear layer (save 60% parameters compared with the projection module in LLaVA-Next, which consists of a $\mathbb{R}^{1024\times4096}$ and a $\mathbb{R}^{4096\times4096}$ linear layer), to integrate high-resolution image information into low-resolution image features at a lower cost. Only the low-resolution features are then utilized as input to the LLM. Given that the projection module (including MLP, Q-former, and our SA-Perceiver) has a significantly lower parameter count and computational complexity than the LLM, the overall system latency is predominantly dictated by the computation delay of the LLM. As SA-Perceiver enables a reduction of visual sequence length up to four-fold, and the time complexity of LLM is $O\left(n^2\right)$, our method can theoretically achieve a maximum reduction in latency by a factor of 16. However, system latency is also affected by factors such as the number of input text tokens, the length of the generated sentences, and other intricate system dynamics. To more accurately assess the impact of SA-Perceiver in reducing computational overhead, we randomly select 1,000 images, remove their textual instructions, resize them to various resolutions, and compare the latency and FLOPs of our method and LLaVA-Next during the feedforward process.

Latency of processing 1000 images (seconds):

Method	**336x336**	**672x336**	**1008x336**	**672x672**
LLaVA-Next	449	475	564	738
VLSA	**373**	**377**	**385**	**399**

FLOPs in processing 1000 images (GFLOPs):

Method	**336x336**	**672x336**	**1008x336**	**672x672**
LLaVA-Next	18798.9	27444.3	36462.3	45840.6
VLSA	**9842.3**	**9884.7**	**10190.8**	**10539.3**

(3) During the reconstructive training, the LDM is required to perform only a single denoising step per iteration, in contrast to the multi-step denoising process used for image generation. Consequently, its computational overhead is much lower than the feedforward process of the LLM, and the additional costs brought by LDM are substantially outweighed by the efficiencies gained through our compressive image encoding (SA-Perceiver). In reference to the comparison method outlined in W1(2), we have quantified the impact of incorporating reconstructive training on both latency and FLOPs. We also report the effects of reconstructive training on the total training time of instruction tuning stage (with our 980K dataset on 16 Nvidia A100). These results validate that our method is indeed highly computationally efficient.

Latency of processing 1000 images (seconds):

Method	**336x336**	**672x336**	**1008x336**	**672x672**
LLaVA-Next	449	475	564	738
VLSA(w/o. Reconstruct)	**373**	**377**	**385**	**399**
VLSA(w/. Reconstruct)	391	394	408	426

FLOPs in processing 1000 images (GFLOPs):

Method	**336x336**	**672x336**	**1008x336**	**672x672**
LLaVA-Next	18798.9	27444.3	36462.3	45840.6
VLSA(w/o. Reconstruct)	**9842.3**	**9884.7**	**10190.8**	**10539.3**
VLSA(w/. Reconstruct)	11646.1	12118.5	12545.0	13443.8

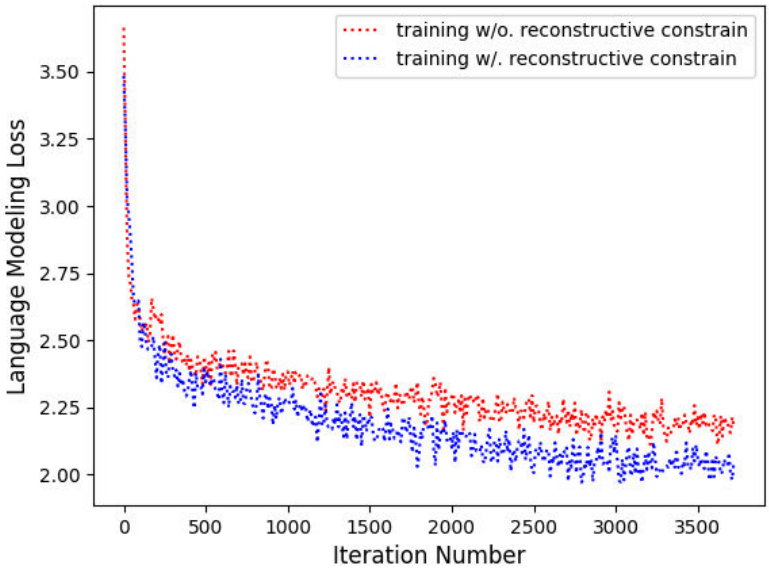
Training time in instruction tuning stage (hours):

Method	Training Time
LLaVA-Next	27.3
VLSA(w/o. Reconstruct)	**14.2**
VLSA(w/. Reconstruct)	16.9

W2:

Our statement in the Limitation section may have been somewhat misleading. While it is indeed the case that the suboptimal combination strategy of the two types of loss can restrict VLSA from realizing its full potential, it is important to highlight that these two losses consistently operate synergistically, even when using the simplest balancing method, where the ratio between the losses is not adjusted, and both losses are applied throughout all training phases. To further illustrate this point, we have included Figure ex1, which depicts the language model loss curves before and after incorporating the reconstruction loss in Stage 1 training of VLSA. The results demonstrate that the introduction of the reconstruction loss leads to both a faster convergence and a more favorable final outcome for the language modeling loss. Therefore, our method does not hinder the optimization efficiency. Our primary focus in this article is to highlight the effectiveness of the proposed method, while further optimization will be addressed in future work.

(Figure ex1) The influence of reconstructive training on language modeling loss.



W3:

To address the reviewer's concern, we integrate VLSA with various language models and adapting it to much higher input resolution.

(1) We report the performance of VLSA with the replacement of the backbone model to Vicuna1.5-7B, Vicuna1.5-13B and Qwen1.5-72B. Additionally, we report the performance of LLaVA-Next with these backbones as references.

Variant	LLM	GQA	AI2D	DocVQA
LLaVA-Next	Vicuna1.5-7B	62.2	66.4	72.5
(ex2) VLSA	Vicuna1.5-7B	63.6	67.5	74.6
LLaVA-Next	Vicuna1.5-13B	65.4	67.0	72.7
(ex3) VLSA	Vicuna1.5-13B	67.2	69.2	76.8
LLaVA-Next	Qwen1.5-72B	71.2	73.4	79.9
(ex4) VLSA	Qwen1.5-72B	**72.6**	**77.1**	**85.1**

(2) We increase the maximum input resolution of VLSA from 672x672 to 4096x4096, and report the preliminary experimental results.

Variant	Res.	GQA	AI2D	DocVQA
LLaVA-Next	672x672	64.6	69.5	73.7
VLSA	672x672	65.3	71.4	75.2
LLaVA-Next	4096x4096	68.4	72.7	76.2
(ex5) VLSA	4096x4096	**69.5**	**76.6**	**80.1**

Q1:

In our previous submission, we explored the following types of test sets: (1) OCR comprehension, e.g., TextVQA, DocVQA; (2) Chart and diagram understanding, e.g., ChartQA, AI2D; (3) Subject-specific question answering, e.g., ScienceQA; and (4) General question answering, e.g., MMVet, MMBench, and MME. During the rebuttal phase, we have added (5) multi-image reasoning with questions involving visual prompts: BLINK, as well as (6) the interleaved benchmark on generalization: DEMON. (The implementation details on new benchmarks will be included in the revised manuscript.) We believe that this diverse and comprehensive array of tests showcases the versatility of our approach.

Comparisions on DEMON:

Method	Multimodal Dialogue	Visual Storytelling	Visual Relation Inference	Multimodal Cloze	Knowledge Grounded QA	Text-Rich Image QA	Multi-Image Reasoning
LLaMA-Adapter V2	14.2	17.5	13.5	18.0	**44.8**	32.0	44.0
(ex6) LLaMA-Adapter V2 + VLSA	**16.0**	**17.9**	15.7	**19.2**	44.7	36.3	**45.5**
LLaVA	7.8	10.7	8.3	15.9	36.2	28.3	41.5
(ex7)LLaVA + VLSA	10.2	11.5	**15.8**	16.1	36.3	**37.2**	44.4

Comparisions on BLINK:

Method	Validation	Test
LLaVA-1.5 7B	37.1	38.0
(ex8) LLaVA-1.5 7B + VLSA	39.3	39.9
LLaVA-1.5 13B	42.7	40.6
(ex9) LLaVA-1.5 13B + VLSA	**46.1**	**45.3**

Q2

In our response to comment W1, we have conducted a thorough analysis demonstrating that VLSA is a highly computationally efficient method, especially when compared to LLaVA-Next, as it significantly reduces both computational complexity and training time. To further adapt VLSA for resource-limited environments, we propose two strategies: (ex10) substituting the LDM in the current reconstruction training with a version that has fewer parameters, and (ex11) adjusting the three-stage training of VLSA to a standard two-stage training process. We have tested both strategies and documented their performance. Additionally, several conventional techniques, such as (1) decreasing the batch size while correspondingly increasing the gradient accumulation steps, and (2) implementing quantized training, will also be highly effective.

Variant	GQA	SQA-I	DocVQA
LLaVA-Next	64.6	75.1	73.7
(ex10) Two-Stage Training	65.2	77.0	74.6

Variant	GQA	SQA-I	DocVQA
(ex11) Smaller LDM	64.8	76.8	75.1
VLSA	**65.3**	**77.5**	**75.2**

Q3:

It seems there has been a misunderstanding regarding our methodology. To clarify, our approach employs continuous image features provided by the CLIP image encoder as the visual input, rather than utilizing codebook indices from VQ-VAE. The codebook indices serve solely as supplementary textual input, acting as ground truth labels for tasks that require predicting these indices, during the instruction tuning process. Following are responses to the remaining concerns.

(1) The name of "cognition alignment":

In VQ-VAE, the codebook functions similarly to token embeddings in a text tokenizer, serving as a repository of various high-level semantics. The code indices produced during the encoding of an image represent the different types of high-level semantics that can be derived from that image. Additionally, as an autoencoder, VQ-VAE's training objective is to ensure that the code indices for each image capture as much semantic information from the original input as possible. As a result, the process through which MLLM generates codebook indices enables it to comprehensively understand high-level semantics from images. Thus, we call this process "cognition alignment." Essentially, MLLM learns how to recognize high-level semantics of images based on abstract visual features provided by the vision encoder from this process.

(2) The improvement of cognitive performance:

In the author's opinion, the existing methods utilizing VQ-based visual encoders experience performance limitations due to the discrete nature of VQ encoding and the restricted size of the codebook. This combination leads to lossy information encoding. In content understanding tasks, the consequences of this information loss are unpredictable, making it difficult to ensure that essential visual information required for executing current commands is preserved during encoding. (In contrast, for generation tasks, the intentional omission of minor details while retaining key information contributes to the creation of more random and diverse images, which is why VQ-type encoders are commonly employed in image generation models.)

In contrast to VQ-based methods, our model utilizes continuous image encodings produced by a CLIP encoder as input, which leads to reduced information loss. This divergency ensures improved performance for our model. In our approach, we treat VQ indices as textualized visual semantic labels that aid the model in comprehending high-level image semantics. Consequently, our method is not hindered by the inherent limitations of VQ-based encoders. Moreover, we integrate a reconstructive training loss for the visual encoder, further minimizing original information loss and subsequently enhancing overall performance.

Q4:

Our paper has already reported the results for TextVQA (Table 1), AI2D (Table 3), and ChartQA (Table 3). Regarding the performance comparison on OCRBench, the results are as follows.

Variant	OCRBench
LLaVA-Next	55.7
VLSA	**58.4**