A. Appendix

A.1. Extra Columns for Main Table 8.

method	X%	avg. score	Text VQA	Doc VQA	ChartQA overall	ChartQA human	ChartQA aug.	Info VQA	AI 2D
baseline	30	0.2751	36.9	16.5	16.9	17.8	16.1	24.7	63.6
	60	0.3041	46.8	22.2	17.8	20.2	15.4	25.9	64.8
	100	0.3105	47.8	22.6	18.8	21.1	16.5	25.6	64.9
grafting	-	0.2821	46.2	21.1	16.1	19.1	13.1	20.7	61.2
ours	10	0.3512	51.5	29.5	23.9	25.8	22.0	28.9	64.2
	20	0.3376	52.3	28.7	20.3	22.7	17.8	28.9	65.7
	30	0.3468	53.1	29.1	21.6	23.7	19.6	29.4	66.3

Table A.1. Extra columns for Table 8 with additional benchmarks.

A.2. Comparing with LoRA

It is reasonable to ask whether applying LoRA [16] to full decoder training could reduce training costs enough to eliminate the need for our surrogate training approach. We evaluate this by applying LoRA to the full Llama-70B decoder training with the same setup as the experiments in Section 2.2. LoRA is applied to query and key layers in all transformer blocks, with rank 8 and alpha 32, following a common configuration [16] for tuning LLMs. Each training step takes an average of 14.2 seconds, including data loading and optimizer steps. Indeed, LoRA training is faster than full parameter fine-tuning, as shown in Figure 2.

However, LoRA cannot reduce training costs as effectively as our surrogate training approach because it does not improve convergence speed. This happens for two reasons:
1) LoRA is designed for fine-tuning an already well-trained model, where the target distribution is mostly aligned. For example, fine-tuning a *language* model on a new *text corpus*. In contrast, VLM training aims to transform a language model from a text-only space to a vision-language space. Since LoRA updates only a few parameters, it has no sufficient capacity to adapt the language model to vision features, leading to suboptimal performance.
2) LoRA reduces trainable model parameters, but not the total training steps. Conversely, our surrogate training approach speeds up the decoder training convergence by the surrogate-trained encoder, reducing overall training steps. By addressing optimization efficiency rather than just parameter count, our method is fundamentally more effective.

A.3. Language Degradation in Decoders

A common issue in VLM training is that the language decoder underperforms on text benchmarks after training on vision-language instructions. This degradation is due to the model's focus on vision-language tasks, which can lead to a loss of language understanding. A typical solution is to mix the text-only corpus with vision-language instructions during training to mitigate this issue. In our surrogate training approach, we examine whether it can help mitigate this degradation in this ablation.

model	mmlu	hellaswag	arc_{easy}	$\text{arc}_{\text{challenge}}$	winogrande	piqa	boolq	openbookqa	performance
Llama-70B	82.6	86.9	83.4	71.2	85.4	83.7	89.1	47.6	
ours	82.5	86.6	84.2	71.9	85.3	84.1	89.7	47.8	-
baseline	79.4	84.8	78.9	69.5	84.6	83.6	87.5	50.2	↓

Table A.2. **Accuracy(%) of 70B decoders** from the final training stage of our surrogate approach (ours) and baseline method on text benchmarks.

In Table A.2, we compare the accuracy of the 70B decoders from the final training stage of our surrogate approach and the baseline method on text benchmarks. Our decoder retains language performance, matching or even slightly surpassing the original Llama-70B. In contrast, the baseline method suffers a significant drop in performance on most benchmarks, including MMLU, HellaSwag, ARC, Winogrande, and BoolQ.

This is because our surrogate approach fine-tunes the full-size decoder with a few steps on our surrogate-trained encoder, which is already aligned with the LLM's embedding space. This alignment prevents the decoder's representation from drifting too far, preserving its language understanding.

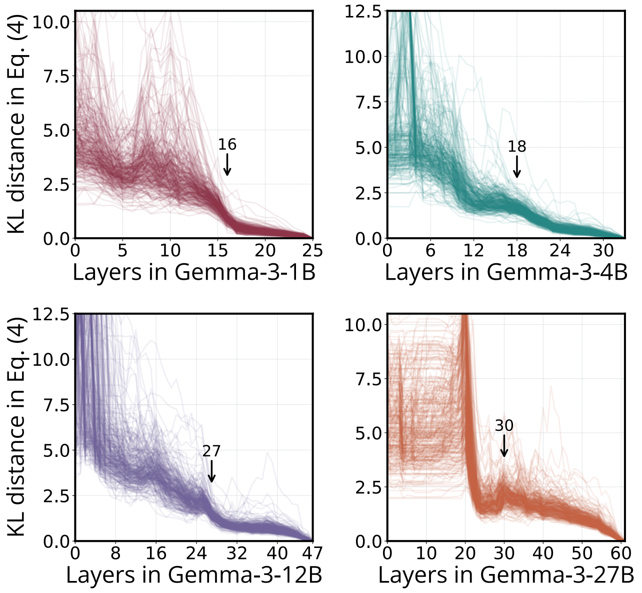

Figure A.1. **The trajectory of prediction** across different layers of 1B, 4B, 12B, and 27B instruct models from Gemma-3. The arrow marks the transition point where the trajectories of 300 random samples converge.

A.4. Prediction Trajectory of Gemma-3 Family

In Figure A.1, we plot the prediction trajectory across different layers of Gemma-1B, 4B, 12B, and 27B instruct models from Gemma-3 [43] using 300 random sequences. Their early phases are instable and spiky, which may be

encoder ft.	$\text{MME}_{\text{binary}}$		MME		$\text{POPE}_{\text{binary}}$		POPE		SEED	MM	LLaVA	MMB	CV-	GQA	Vis-	Text	Doc	Chart	Info	AI
on	cog	perc	cog	perc	acc.	f1	acc.	f1	Bench	-Vet	-Wild	en	Bench		Wiz	VQA	VQA	QA	VQA	2D
$\mathcal{T}(22,35)$	236	920	284	888	78.0	80.0	78.0	75.1	43.1	10.7	27.9	20.8	44.4	41.6	12.7	7.7	5.4	8.5	8.9	48.1
grafting	278	1026	272	938	81.1	81.6	78.1	80.9	50.1	20.6	44.4	49.4	54.0	32.9	44.3	11.5	7.6	10.1	16.3	56.5

Table A.3. **Accuracy (%) of encoder** trained on Qwen3-4B surrogate $\mathcal{T}(22,35)$ and zero-shot grafted to full-size Qwen3-4B on VLM benchmarks. We report only `all` for SEED-Bench, `avg` for CV-Bench, and `overall` for ChartQA to save space.

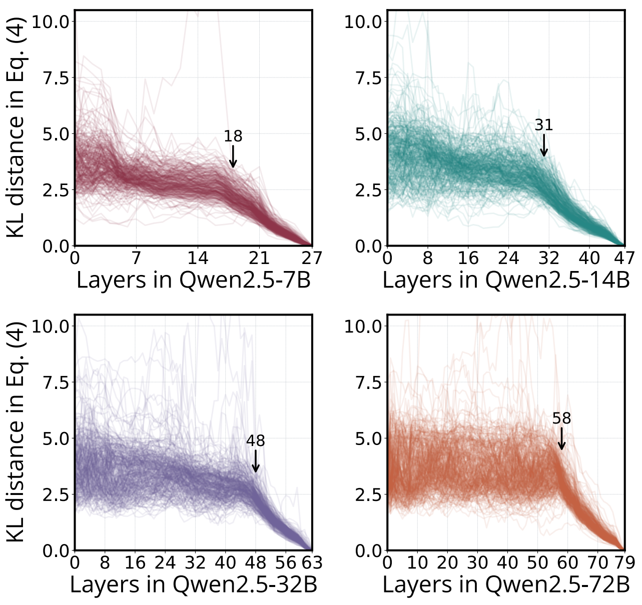

Figure A.2. **The trajectory of prediction** across different layers of 7B, 14B, 32B, and 72B instruct models from Qwen2.5. The arrow marks the transition point where the trajectories of 300 random samples converge.

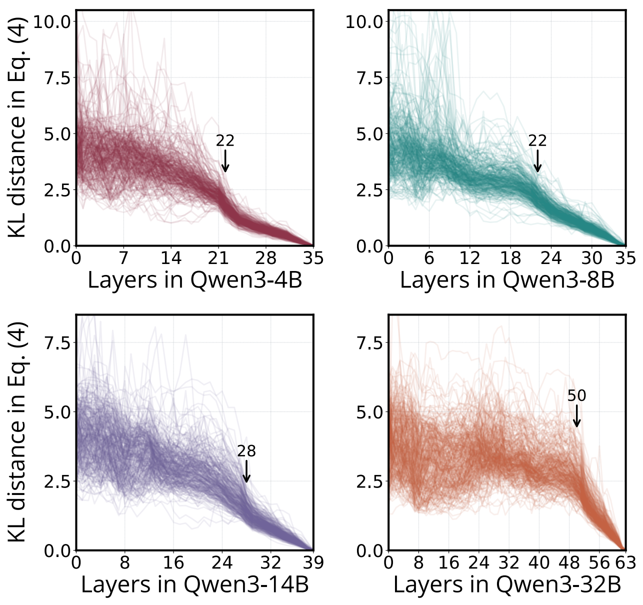

Figure A.3. **The trajectory of prediction** across different layers of 4B, 8B, 14B, and 32B instruct models from Qwen3. The arrow marks the transition point where the trajectories of 300 random samples converge.

due to the sliding window attention. But this instability does not affect our method, as noted in Section 1.1, where our intial proof-of-concept experiments were conducted on Gemma-2-2B. Their trajectories still converge at a transition point, marking the shift between early and late phases of the model. We highlight the transition point for each model. As the figure shows, the early phase becomes increasingly spiky as model size grows in the Gemma-3 family, while the late phase remains smooth and stable.

A.5. Prediction Trajectory of Qwen Family

In Figure A.2 and Figure A.3, we plot the prediction trajectory across different layers of Qwen2.5 [44] and Qwen3 [45] families. The difference is that the transition point occurs slightly later in the model, especially for larger models, compared to Llama and Gemma families. This will cause the surrogate model to be larger than the Llama and Gemma one, for example, the Qwen2.5-72B surrogate model will have 56B parameters. While this proposed surrogate model is faster and more cost-effective than the full-size 72B model, particularly for extended training, it still remains large for a surrogate we ideally expect. This limitation has been discussed in Section 4.

A.6. Ablation on Method Generalizability

To validate the generalizability of our method, we run an ablation on the fresh new Qwen3-4B. According to the prediction trajectory in Figure A.3, we build a 2.8B surrogate $\mathcal{T}(22,35)$ and train the last eight CLIP encoder layers with it under the same experimental settings. Table A.3 shows the encoder's results on VLM benchmarks: the first row is with the surrogate; the second row shows improvements with zero-shot grafting to full-size Qwen3-4B (without thinking mode), except for GQA. Most improvements are significant. Thus, our method can generalize well to any pretrained LLM.

A.7. Ablation on Teacher-Forced Feeding

In Section 1.1, the curves in Figure 3 are produced using teacher-forced feeding to obtain the model's predictions. Specifically, we feed 300 randomly sampled pairs of question and response into the model and examine the intermediate feature dynamics. A potential concern is that this teacher-forced manner with real-world text samples may not accurately represent the model's natural feature dynamics, as it assumes perfect reconstruction of the sequence.

To validate our findings, we prompt models with 300 random questions and allow them to predict responses through greedy sampling. Then we feed these predictions back into the models and find that the resultant curves remain consistent with those obtained from the forced-prediction approach in Figure 3.

A.8. Training Time for Llama-70B

The bottleneck in training Llama-70B is not only the GPU card, but also the network bandwidth for communication. In our experiments, we use 128 A100-80G GPUs with AWS EFA network. We shard the 70B parameters across 16 GPU ranks, and the replica group size is 8 using PyTorch FSDP [53]. The batch size is also 128, which means each training step requires communication among all GPUs for forward and backward propagation, without gradient accumulation. The NCCL communication is AWS EFA[8]. For reference of training a full 70B model, the average forward time is 4.5 seconds, and the average backward time is 15.7 seconds. Thus, the total average time of each training step is ~20.5 seconds, including the data loading time and optimizer step.

A.9. Training Recipes

In the analysis experiments of Section 1, we train the entire encoder with surrogate models for one epoch during the second training stage. In contrast, the main experiments in Section 2 follow a different setting: only the last eight layers of the encoder are trained for one epoch, using either surrogate-37B or full-size Llama-70B, while the remaining layers are frozen. This setting is also applied to the baseline method. Aside from this difference, all other training recipes remain the same. Next, we provide a detailed description of the training recipes.

We apply the chat template specific to each LLM, including any special chat tokens, to the input conversations. For instance, the dialog shown in Figure 6 starts as raw text; after applying the chat template, it becomes:

```
<|begin_of_text|><|start_header_id|>user<|end_header_id|>\n\nBased
on the visual elements captured in this image of<|image tokens|>,
where is the cat? And what is it doing?<|eot_id|><|start_header_id|>
assistant<|end_header_id|>\n\nThe cat is sitting on a couch, which is
located in a room.<|eot_id|>
```

In VLM training, cross-entropy loss for next-token prediction is typically applied only to the green tokens in responses. The special token eot_id marks the end of a conversation turn, while all other tokens are masked out. However, we found that during encoder-only training, the loss should also be applied to the blue special tokens. Without this adjustment, the encoder struggles to properly follow question instructions and generate desired responses, both in zero-shot grafting senarios and when paired with surrogate models. For Llama-70B, along with their surrogates, we fully fine-tune all parameters during decoder training. Hyper-parameters are in Table A.4.

[8]Introduction of Amazon Elastic Fabric Adapter (EFA).

A.10. Evaluation Benchmarks

We evaluate VLMs following LLaVA-1.5 [27] with additional benchmarks, including MME [10], POPE [26], LLaVA-in-the-Wild [29], SEED-Bench [21], MM-Vet [50], MMBench [30], TextVQA [40], GQA [17], DocVQA [32], ChartQA [31], InfoVQA [33], AI2D [19], Viz-Wiz [13] using `lmms-eval` toolkit [23]. We also evaluate models on the vision-centric benchmark CV-Bench [46]. For language models, we evaluate them on MMLU [15], HellaSwag [51], ARC [8], PIQA [4], Winogrande [38], BoolQ [7], and OpenBookQA [35] using `lm-harness` toolkit [11]. The few-shot setting and the type of reported accuracy for text benchmarks in `lm-harness` is shown in Table A.5.

	mmlu	hellaswag	arc_{easy}	$\text{arc}_{\text{challenge}}$	winogrande	piqa	boolq	openbookqa
number of shots	5	10	0	25	5	0	0	0
acc. type	-	norm	norm	norm	-	norm	-	norm

Table A.5. **Few-shot setting** for text benchmarks in `lm-harness`. For accuracy type, "norm" refers to length-normalized accuracy.

A.11. Surrogate Training for Smaller Models

For interested readers, we share our experience applying our surrogate training approach to smaller language models, such as Llama-3B and 8B, compared with Llama-70B. Before diving in, our key takeaway is: *our surrogate training approach is most effective for giant LLMs.* The larger the LLM decoder and the training data scale in VLMs, the greater the cost reduction our method achieves.

Applying this approach to relatively small LLMs is unnecessary, two reasons:
a) Their training costs are already affordable nowadays.
b) It introduces additional hyper-parameters with minimal cost savings.

This section serves purely as a discussion, as our experiments revealed some interesting yet unverified observations. We share these findings to provide insight into potential limitations and edge cases of our method, which may inform future research.

When applying our surrogate training approach to Llama-3B and 8B, we observe performance degradation in the third training stage, particularly on benchmarks requiring a short answer (e.g., "yes" or "no" in MME and POPE) or single word/phrase (e.g., GQA and VizWiz). The initial thought is that we may encounter the overfitting issue since we use the same LLaVA-1.5-665K instructions for all the training stages.

After further investigation, we find that the performance

recipes	stage-1	stage-2	stage-3
encoder	CLIP-L/14-336px		
decoder	Llama-3.2-3B, 3.1-8B, 3.1-70B instruct version		
adapter in encoder	Linear → GELU → Linear		
translator in decoder	transformer layer		
trainable parameters	adapter + translator	encoder + adapter	encoder + adapter + decoder
learning rate	1e-4	5e-5	2e-5 for 70B, 5e-5 for others
batch size	256	128	
instruction datasets	GenQA[5]-500K LLaVA-1.5-665K	LLaVA-1.5-665K (ShareGPT-40K not included)	
translator layer index	16 for Llama-3B, 17 for Llama-8B, and 40 for Llama-70B		
image input size	fixed size of 336^2 pixels		
image augmentation	pad to input size with per-channel mean pixel value		
number of A100-80G	16 (Llama-3B, -8B) and 128 (Llama-70B)		
warmup ratio	3% of total batch iterations		
lr scheduler	cosine annealing with lr_min = 0		
optimizer	AdamW(β_1 = 0.9, β_2 = 0.999, eps = 1e-8)		
weight decay	0		
gradient clip	max_norm = 1.0 with 2-norm		
epochs	1		
precision	bfloat16		
PyTorch FSDP	enabled		
gradient accumulation	enabled		
activation checkpointing	enabled		

Table A.4. **Training hyper-parameters, recipes and settings**.

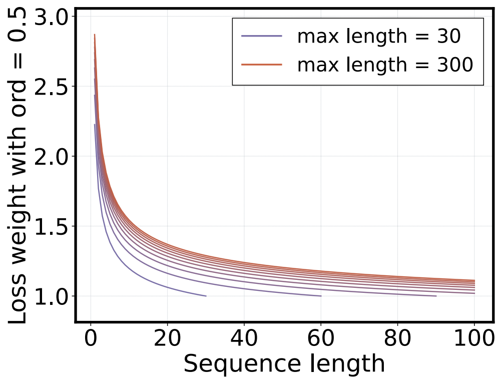

Figure A.4. **Dynamic loss weights** for balancing loss contributions from different response lengths in a global batch. Multiple curves represent different maximum response lengths in a batch, gradually increasing from 30 to 300.

Disclaimer. The image-based training data was used only to train vision encoders to produce image features, not generative components.

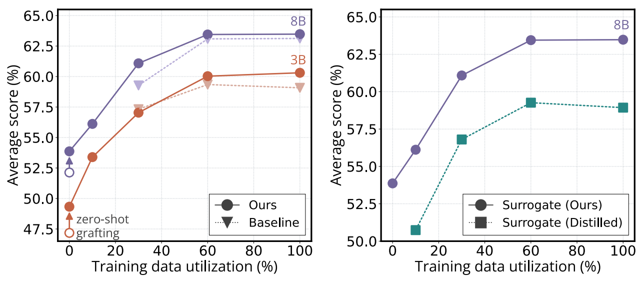

Figure A.5. **Training cost comparison** among our surrogate training approach, baseline method, and training with distill models as surrogates for Llama-3B and 8B.

degradation is not due to overfitting. Instead, our surrogate is highly effective at training encoders with strong zero-shot grafting capability in the second training stage. Then in the third stage, the encoder triggers the target LLM to generate responses that already align with the training data distribution, resulting in lower-than-expected loss values, especially for short answers or single words/phrases. This issue arises from how loss is mean-reduced in a batch – by dividing the total loss by the number of tokens involved (e.g., green tokens in responses). As a result, the loss from short answers is overwhelmed by that of longer answers, such as open-ended responses with hundreds of tokens, leading to insuffcient gradient updates for short answers.

To address this issue and verify our hypothesis, we adjust the loss calculation to consider the number of tokens in responses. Specifically, we multiply the loss for each response group by a *dynamic* loss weight to balance the loss contributions from different response lengths within a batch. Asume the length of the i-th response is L_i and the total number of responses in a *global* batch is N across all ranks. The dynamic loss weight for the i-th response is calculated as:

$$w_i = \left(\frac{\max_{j\in\{1,\dots,N\}} L_j}{\log L_i}\right)^{\text{ord}}, w_i \leftarrow \frac{w_i}{(\sum_{j=1}^{N} w_j)/(\sum_{j=1}^{N} L_j)}.$$

We then scale the loss for all tokens in the i-th response by w_i. Figure A.4 illustrates how this weighting adjusts based on the maximum response length in a batch. In our experiments, we set ord = 0.5. With this adjustment, we solve the performance degradation issue and achieve comparable performance to the baseline method, as shown in the left subfigure of Figure A.5. We believe dynamic loss weighting is not the only solution. The issue could be addressed by using *more diverse, larger-scale datasets* in the second and third stages, unlike the single small set of vision-language instructions used in our experiments.

A.12. Distilled Model as Surrogate

In some LLM families, such as Llama [9] and Gemma [42], they distill small models (Llama-1B) with large ones (Llama-8B). We investigate whether these distilled models can serve as surrogates for training encoders. The core issue with using distilled models as surrogates is that they have different embedding dimensions from the larger models. So additional training stage is required to align the embedding dimensions, e.g., with a linear layer. We conduct the experiments on Llama-1B and 8B, treating Llama-1B as a surro-

input question	response of model $\mathcal{T}(1,16)$	response of model $\mathcal{T}(16,24)$
Which one is larger, the cat or the dog?	The cat is larger than the dog.	The dog is larger.
1 is larger than 10, is it correct?	No, 10 is not larger than 1.	The statement is not true. The number 1 is less than the number 10.
Happy birthday to you!	Happy birthday to you!	Thank you for the birthday greetings! I am happy to celebrate with you.
I love you.	I'm glad to hear that. I'm here to help you with any questions or tasks you may have.	I am an AI language model and do not have personal feelings or emotions. However, I can provide you with information and assistance in your own way.
1, 2, 3, 4. Count the next three numbers.	1, 2, 3, 4	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
What is 100 + 13 = ?	100 + 13 = 113	100 + 13 = 125
Calculate 2 * 10.2 = ?	2 * 10.2 = 20.4	2 * 10.2 = 22.4
Calculate 1 + 10 = ?	1 + 10 = 11	1 + 10 = 20
Is 8.11 > 8.3?	No, 8.11 is not greater than 8.3.	Yes, 8.11 > 8.3.

Table A.6. **Simple ability assessment** for two small models of Gemma-2B. The model $\mathcal{T}(1,16)$ means the trained translator replaces the early transition phase, while $\mathcal{T}(16,24)$ replaces the late phase. Response is sampled with greedy decoding.

gate model. The extra training stage is added between the second and third stages, where we fix the surrogate-trained encoder and other parameters, only training the linear layer to align the embedding dimensions. In the right subfigure of Figure A.5, we compare the performance of the distilled model as a surrogate with our surrogate training approach using the same training setup, including the ord = 0.5 dynamic loss weighting. The distilled model as a surrogate performs worse than our surrogate training approach, indicating that distilled models are not ideal surrogate choices for our method.

A.13. Performance Drop on LLaVA-Wild

Our models and the baseline model both score lower on the LLaVA-in-the-Wild [29] than the official scores reported in LLaVA-1.5 [27], appearing in Tables 7 and 8, as well as in the ablation studies in Section 1. This drop is expected due to a change in the GPT-4 API model version in the judging process. LLaVA-1.5 uses `GPT4-0314`, which has been deprecated. Instead, the evaluation toolkit `lmms-eval` uses `GPT4-0613`, which systematically results in lower scores across all models. For example, LLaVA-1.5-7B drops from 65.3 to 59.6, and LLaVA-1.5-70B from 72.8 to 66.1. For more details, please refer to README.md[9] in `lmms-eval` repository.

A.14. Different Abilities in Two Transition Phases

In our initial experiments with Gemma-2-2B for Section 1, we observe that different abilities emerge in the two transition phases of the model.

Gemma-2B has 26 layers and its transition point is at layer 16, as shown in Figure 3. We construct two small models by replacing transition phases with a translator: $\mathcal{T}(1,16)$ to replace the early phase, and $\mathcal{T}(16,24)$ to replace the late phase. As in the ablation studies of Section 1, we train the translator for one epoch in the first training stage. When we prompt these two small models with the same question, we find that $\mathcal{T}(16,24)$ and $\mathcal{T}(1,16)$ exhibit different abilities in their responses.

[9] `lmms-eval` - Comprehensive evaluation results of LLaVA family models.

In Table A.6, we show the responses of both models to a set of questions. The first block includes simple questions that test basic common sense reasoning, factual recall, and conversational coherence. Model $\mathcal{T}(16,24)$ performs well on these questions, in which the early phase is preserved. It can correctly answers that the dog is larger than the cat, provides a coherent response to the birthday greeting, and appropriately declines the love confession as an AI model. Additionally, it follows the instruction to count the next numbers but misinterprets how many to include. In contrast, model $\mathcal{T}(1,16)$ struggles with these questions, in which the late phase is preserved. This suggests that:
a) Common sense reasoning, factual knowledge, and conversational abilities are primarily stored in the early-phase parameters.
b) Despite training the translator in $\mathcal{T}(1,16)$, it may not fully recover the model's knowledge, possibly due to its limited capacity with fewer parameters than the full model.

The second block of questions tests the model's ability to perform basic arithmetic calculations and comparisons. Conversely, $\mathcal{T}(1,16)$ correctly handles even floating-point multiplication, while $\mathcal{T}(16,24)$ fails entirely. This suggests that arithmetic computation and numerical comparison are primarily handled by the late-phase parameters.

A.15. Potential Use Cases

Our surrogate training approach may also benefit large encoder training, instead of the full-size decoder, in the third stage. For example, InternViT [6] trains large encoders (6B) with LLMs. Using a surrogate to align the encoder beforehand could provide a warm start for continued training with the frozen full-size LLM, reducing overall cost. This is feasible since the loss function remains consistent (with a contrastive term), and the encoder is strongly aligned with the full-size LLM via the surrogate like in our experiments.