

Zero-Shot Vision Encoder Grafting via LLM Surrogates

Kaiyu Yue Vasu Singla Menglin Jia[†] John Kirchenbauer
 Rifaa Qadri Zikui Cai Abhinav Bhatele Furong Huang Tom Goldstein
 University of Maryland [†]Meta

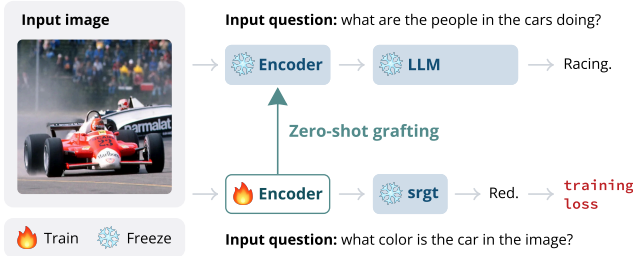


Figure 1. **Zero-shot vision encoder grafting** via a small language surrogate (**srgt**) model to trigger the target LLM to perform visual understanding task without any additional training.

Abstract

Vision language models (VLMs) typically pair a modestly sized vision encoder with a large language model (LLM), e.g., Llama-70B, making the decoder the primary computational burden during training. To reduce costs, a promising strategy is to first train the vision encoder using a small language model before transferring it to the large one. We construct small “surrogate models” that share the same embedding space and representation language as the large target LLM by directly inheriting its shallow layers. Vision encoders trained on the surrogate can then be directly transferred to the larger model, a process we call *zero-shot grafting*¹ – when plugged directly into the full-size target LLM, the grafted pair surpasses the encoder-surrogate pair and, on some benchmarks, even performs on par with full decoder training with the target LLM. Furthermore, our surrogate training approach reduces overall VLM training costs by ~45% when using Llama-70B as the decoder.

Most modern auto-regressive VLMs are built by extracting visual features from images using an encoder like CLIP [33] or SigLIP [40, 45], and placing these features into the context window of an LLM. The image features must

¹We define *zero-shot grafting* as plugging a vision encoder trained on a surrogate model directly into its target LLM without additional training. In contrast, *transferring* involves further fine-tuning after grafting.

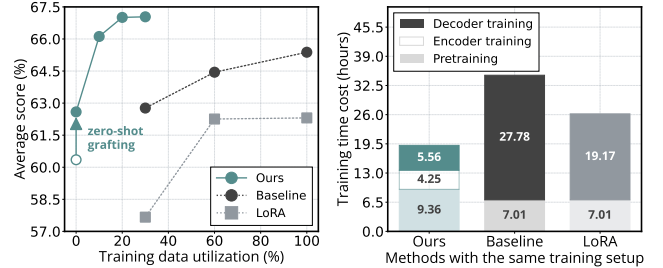


Figure 2. **Reducing full decoder training cost** with our surrogate-trained encoder for Llama-70B in VLMs. Hollow \circ indicates the average score of the surrogate-trained encoder on the left.

be aligned with the representation space of the LLM, and this is achieved by training the entire pipeline end-to-end. The cost of such training is often severely dominated by the language model. For example, plugging CLIP (approx 400M parameters) into Llama-70B [9] results in a pipeline where vision encoder training occupies almost none of the required memory and computation.

In this paper, we explore methods of performing encoder alignment using small lightweight language models, and transferring the resulting features to a large language model. We train small *surrogate* language models with the same representation space as a larger target LLM. After training the vision encoder on this small surrogate model, we can then transfer it to the larger model, either directly (grafting) or with fine-tuning.

A major focus of our work is on understanding how to construct small surrogate models that accurately mock larger target LLMs. Our method of creating such small models stems from analyzing the internal prediction dynamics of LLMs, particularly how predictions evolve across layers. This analysis reveals two distinct phases in the prediction trajectory, separated by a clear transition point. We construct our small models by preserving the layers that participate in the early feature extraction phase of inference, and condensing all other layers. Since the small model inherits its shallow parameters from the target LLM, it shares the

same embedding space as the original larger model and can effectively stand in as its surrogate. Our surrogate model has two major advantages:

Zero-shot grafting capability. Vision features trained on a smaller and less resource-intensive surrogate can be directly used by the larger target LLM without any fine-tuning, as depicted in Figure 1. This *zero-shot grafting* demonstrates these surrogate-trained encoders effectively trigger visual understanding in target LLMs.

Fast-converging VLM training. The encoders trained on surrogate models can be further fine-tuned with the full-size target LLM. Since they are already aligned with the LLM’s embedding space, they achieve high performance with comparatively little full-scale training. Our experiments show a ~45% cost reduction for full decoder training with Llama-70B, as shown in Figure 2, highlighting the efficiency of our surrogate-trained encoders.

Table of Main Contents

- Section 1: We detail the method of constructing our surrogate model, providing analysis that demonstrates how we discovered, developed, and validated our approach through experimental ablations.
- Section 2: We show our surrogate models for giant LLMs like Llama-70B, producing encoders with a strong zero-shot grafting ability, which can also accelerate the full decoder training of giant language models for VLMs.

1. Building Surrogate Models

In this section, we present our approach for building small surrogate models for target LLMs. First, we analyze the LLM’s hidden features to identify the critical transition point between shallow and deep information processing layers. Next, we observe that the second/deep phase of inference contributes very little to encoder transferability, and observe that image features transfer well between models when they share their early/shallow processing layers. Finally, we validate these findings and propose to construct surrogate models by preserving the early-phase layers while replacing late-phase layers with a translator.

1.1. Analyzing the Prediction Trajectory

For a target LLM and input array² of N text token IDs $\mathbf{t} \in \mathbb{Z}^N$, we trace the evolution of features over a forward pass of the model. By propagating these tokens through all L transformer layers, we obtain intermediate hidden states $\mathbf{X}^\ell \in \mathbb{R}^{N \times D}$ from each layer, where $\ell \in [0, L-1]$ denotes the layer index and D is the hidden dimension. The final hidden states \mathbf{X}^{L-1} are passed through a normalization layer

²Bold capital letters denote a matrix \mathbf{X} , and bold lower-case letters a column vector \mathbf{x} . $\mathbf{X}[i, j]$ refers to the element at row i and column j in matrix \mathbf{X} . All non-bold letters represent scalars.

and the final linear layer $\mathbf{W} \in \mathbb{R}^{V \times D}$ to produce the logits, where V is the vocabulary size. The probability distribution for the predicted next token can be computed for all positions:

$$\mathbf{P} = \text{softmax}(\text{norm}(\mathbf{X}^{L-1})\mathbf{W}^\top) \in \mathbb{R}^{N \times V}. \quad (1)$$

The probability for the next output token at each individual position is

$$\mathbf{p} = \mathbf{P}[1 : , \mathbf{t}[: -1]] \in \mathbb{R}^{N-1}, \quad (2)$$

where $\mathbf{P}[1 : , \mathbf{t}[: -1]]$ shifts \mathbf{P} by one position forward and indexes by \mathbf{t} up to the second-to-last position, aligning each token’s probability with its following token in the sequence.

For each layer’s hidden states \mathbf{X}^ℓ , we compute the intermediate probability distribution \mathbf{q}^ℓ following the same procedure:

$$\mathbf{q}^\ell = \text{softmax}(\text{norm}(\mathbf{X}^\ell)\mathbf{W}^\top)[1 : , \mathbf{t}[: -1]]. \quad (3)$$

To capture the trajectory of evolving predictions, we calculate the KL divergence between the normalized layer-wise distribution \mathbf{q}^ℓ and the final distribution \mathbf{p} :

$$D_{KL}(\mathbf{q}^\ell \parallel \mathbf{p}) = \mathbf{1}^\top (\mathbf{q}^\ell \log \frac{\mathbf{q}^\ell}{\mathbf{p}}), \quad (4)$$

where $\mathbf{1} \in \mathbb{R}^{N-1}$ is a vector of ones, \log is applied element-wise. Eq. (4) quantifies the deviation of each layer’s prediction from the final model output, offering insight into how much each layer’s distribution shifts along the prediction trajectory. This measure enables a deeper understanding of each layer’s role in shaping the model’s eventual output distribution.

In Figure 3, we plot Eq. (4) across different layers of the Llama-3B, 8B, and 70B³ models by feeding⁴ 300 random samples from GenQA [5]. To demonstrate the same curve pattern in a different model family, Gemma-2B is also included. Each model displays a distinct *phase transition* where the curves abruptly coalesce and then monotonically converge to the final distribution. For example, in Llama-8B, this point appears to occur around layer 17 whereas for Llama-70B it is closer to layer 40. We speculate that this point marks a transition in the type of position-wise information processing occurring in the model, where the internal states shift from *early phase* before the transition point to the *late phase* after it. The layers in the early phase process information from individual token embeddings and combine simple representations together to form higher order concepts, then layers in the late phase converge towards a specific next-token prediction.

³Unless stated otherwise, each model mentioned refers to its latest instruct version. For example, Llama-3B indicates Llama-3.2 3B, Llama-70B represents Llama-3.1 70B, and Gemma-2B denotes Gemma-2 2B.

⁴One concern about this teacher-forced manner is ablated in Sec. A.1.

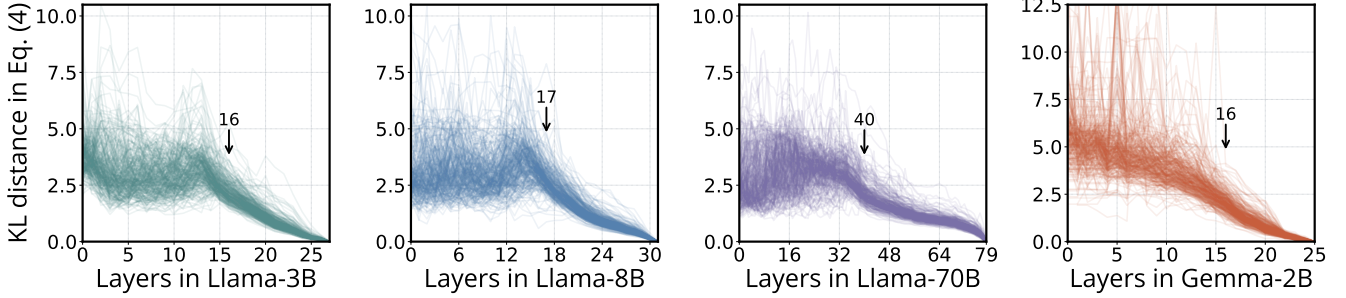


Figure 3. **The trajectory of prediction** across different layers of Llama-3B, 8B, and 70B, and Gemma-2B from a different model family. The arrow marks the transition point where the trajectories of 300 random samples converge.

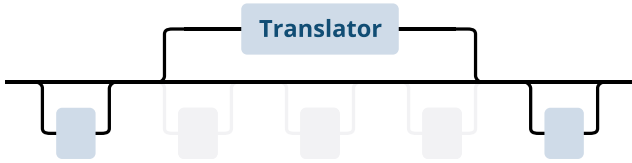


Figure 4. **Replacing layers with a translator.** Despite the relative size in the illustration, our translator is simply an identical transformer layer inherited from the target LLM. The translator bypasses many network layers, and is initialized from the shallowest original layer that it replaced.

1.2. Studying the Transition Phases

To test our hypothesis on the transition point, we experiment with Llama-3B⁵ by replacing consecutive layers of each phase with a single transformer layer called a *translator* (terminology adopted from [3]), as depicted in Figure 4. From Llama-3B’s 28 layers, we preserve the first ($\ell = 0$) and last ($\ell = 27$) layers while replacing two groups of eleven layers each with a translator \mathcal{T} : layers from $\ell = 1$ to 11 indicated as $\mathcal{T}(1, 11)$ for the early phase before Llama-3B’s transition point ($\ell = 16$ in Figure 3) and $\mathcal{T}(16, 26)$ for the late phase after it. Each is a 2B small model.

Next, we examine the two transition phases by evaluating vision encoders trained on models $\mathcal{T}(1, 11)$ and $\mathcal{T}(16, 26)$. To understand their differences and how they affect encoder transferability to the target LLM, we construct two LLaVA-like VLMs using these small models as decoders. We employ a two-stage training approach to conduct the initial experiments:

- 1) First, we *simultaneously* pre-train a vision adapter (a two-layer MLP) and the translator on 1M instructions⁶, combining LLaVA-1.5-665K [26] vision-language instructions and random GenQA [5] 500K text instructions, for one epoch.
- 2) Then, we fine-tune the encoder (ViT-L/14@336px) and vision adapter with the frozen decoders on the LLaVA-1.5-665K instructions for one epoch.

⁵Initial experiments with Gemma-2B showed similar results. In later sections we adapt the method to a 70B model.

⁶This differs from the typical pre-training of vision adapters, which use

	mmlu	hellaswag	arc _{easy}	arc _{challenge}	wino _{grande}	piqa	boolq	openbookqa
small model								
Llama-3B	60.7	73.0	71.1	52.7	70.6	77.1	78.9	39.2
$\mathcal{T}(1, 11)$	26.6	42.5	50.3	27.7	53.5	66.6	57.5	32.4
$\mathcal{T}(16, 26)$	58.9	57.2	54.8	38.5	64.3	67.6	78.2	32.6
$\mathcal{T}(16, 26)^*$	56.9	57.3	57.5	40.7	64.3	70.1	79.9	35.2

Table 1. **Accuracy (%) of small models** for Llama-3B on text benchmarks. * is a control experiment added later in the study.

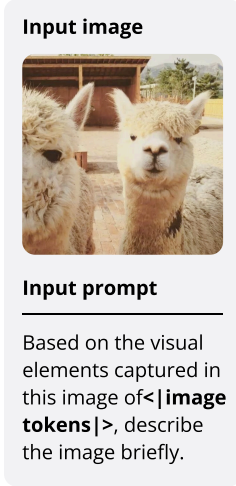
Evaluating decoders. After fine-tuning translators in the first stage, we evaluate models $\mathcal{T}(1, 11)$ and $\mathcal{T}(16, 26)$ on text benchmarks⁷ (Table 1). The first row is the baseline performance of Llama-3B. The second and third rows show the performance of the decoders with early- and late-phase layers replaced, respectively. A significant performance drop occurs when replacing early-phase layers, underscoring their critical role in understanding and generation.

Evaluating encoders. During the second stage, encoders are fine-tuned with small models $\mathcal{T}(1, 11)$ and $\mathcal{T}(16, 26)$. We also train an encoder with the full-size Llama-3B as our baseline, listed in the first row of Table 2. For each model, $\mathcal{T}(1, 11)$ and $\mathcal{T}(16, 26)$, we report two results: a) performance with their respective encoders, and b) performance with these encoders *zero-shot grafted* to Llama-3B. For case b), since Llama-3B is never trained on vision-language instructions, it cannot consistently follow special instructions in benchmarks like MME [10] and POPE [25] that expect “yes” or “no” answers by prompting with “single word or phrase”. For these benchmarks, we prompt the model with binary prompts, directing it to answer with “yes” or “no” to ensure measurable responses.

Table 2 clearly shows that the encoder trained with early-phase layers preserved model $\mathcal{T}(16, 26)$ outperforms the one with early-phase layers discarded model $\mathcal{T}(1, 11)$. Remarkably, performance improves further when the encoder

captions rather than instructions.

⁷To ease the benchmarking, we evaluate our instruct models on the same benchmarks as the non-instruct models, i.e., base models, and report accuracy produced by log-likelihood.



Encoder fine-tuned by small model $\mathcal{T}(1, 11)$ \rightarrow Llama 3B

Zero-shot grafting: failure

There is no image provided. The text appears to be a collection of phrases and sentences that seem to be related to a topic, possibly about a image or a visual element, but no actual image is included.

Encoder fine-tuned by small model $\mathcal{T}(16, 26)$ \rightarrow Llama 3B

Zero-shot grafting: success

The image shows two lambs standing side by side in a yard, with one lamb having a distinctive tuft of wool on its forehead. The lambs appear to be in a natural setting, possibly in a farm or a rural area.

Encoder fine-tuned by Llama 3B \rightarrow Llama 3B

The image features two sheep standing next to each other, with one sheep looking directly at the camera. The other sheep is slightly turned away, but still visible in the scene. The sheep are standing on a dirt path, and there is a wooden fence in the background.

Pre-trained Encoder \rightarrow fine-tuned Llama 3B

The image features two adorable baby llamas standing next to each other in a brick courtyard. They are facing the camera, with one of the llamas looking directly at the viewer. The scene appears to be set in a zoo, as there is a bench visible in the background. The two llamas are the main focus of the image, showcasing their unique and charming appearance.

Figure 5. **Qualitative results on zero-shot grafting capability** of encoders trained with small models for Llama-3B. For comparison, we also include responses from the encoder trained with Llama-3B and the fine-tuned Llama-3B. Response is sampled with greedy decoding. The encoder trained on $\mathcal{T}(16, 26)$ achieves strong zero-shot transfer to Llama-3B. A \rightarrow B means plugging A into B.

encoder fine-tuned on	MME _{per} _{binary}	POPE _{binary}	SEED-Bench	MMVet	LLaVA-Wild	MMB _{en}	performance
Llama-3B	1028	81.7	54.2	24.1	42.9	41.8	
model $\mathcal{T}(1, 11)$	599	63.2	25.8	14.3	37.2	0.6	
zero-shot grafting	540	2.7	25.3	6.9	26.3	9.2	↓
model $\mathcal{T}(16, 26)$	923	70.4	53.2	20.6	42.7	45.4	
zero-shot grafting	1022	80.1	53.1	23.1	56.6	47.4	↑
model $\mathcal{T}(16, 26)^*$	1162	84.4	59.8	25.0	48.1	50.3	
zero-shot grafting	714	22.3	40.7	11.2	34.2	30.1	↓

Table 2. **Accuracy (%) of encoders fine-tuned by small models** for Llama-3B on VLM benchmarks. * indicates a control experiment added later in the study.

fine-tuned on $\mathcal{T}(16, 26)$ is zero-shot grafted to Llama-3B, as shown in the third row block. This improvement highlights that the encoder trained with $\mathcal{T}(16, 26)$ can produce image features that are interpretable by Llama-3B.

In Figure 5, we present qualitative results showcasing the zero-shot grafting capability of the encoders trained via $\mathcal{T}(1, 11)$ and $\mathcal{T}(16, 26)$. The responses enhance the above results that replacing the early-phase layers causes the encoder to fail in generating image features that are directly interpretable by the full-size Llama-3B.

Are early layers the most critical for encoder transfer?

The shallow phase of inference plays a crucial role in transferring a pre-trained encoder to the full-size target LLM. To concretely verify this observation, we conduct a control run based on $\mathcal{T}(16, 26)$, in which we unfreeze every other layer before the translator and train them alongside it during the first stage. This control experiment is designed to disrupt the original early-phase parameters, allowing us to

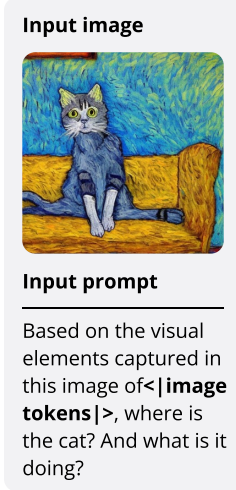
completely assess their impact on encoder transferability. We denote this modified model as $\mathcal{T}(16, 26)^*$.

First, back in Table 1, the last row indicates that fine-tuning additional layers alongside the translator leads to better performance on text benchmarks. However, in Table 2, when evaluating the encoder trained on $\mathcal{T}(16, 26)^*$, we actually observe a huge loss of zero-shot grafting ability. This suggests that modifying early-phase parameters in $\mathcal{T}(16, 26)$ enhance performance on both text and VLM benchmarks when evaluated through itself, but fails to preserve the encoder’s zero-shot grafting capability as the embedding space of $\mathcal{T}(16, 26)^*$ drifts away from the target model.

How many early-phase layers should be preserved? If retaining the original early-phase parameters is necessary, the next question is how many layers to preserve for effective zero-shot grafting. In other words, we seek to confirm the transition point in Figure 3 as the optimal starting point for layer removal and translator insertion.

To ensure generalization, we conduct this ablation with Llama-8B, where the transition point is around layer $\ell = 17$. To validate the transition point, we create three small models by progressively reducing the replaced layers before the transition point: $\mathcal{T}(6, 30)$, $\mathcal{T}(12, 30)$, and $\mathcal{T}(17, 30)$. As shown in Table 3, the performance of these three models on text benchmarks corroborates our findings from Llama-3B, demonstrating that early layers are indeed important. Keeping more early layers leads to better performance, with the best achieved by the model $\mathcal{T}(17, 30)$.

We evaluate the zero-shot grafting capability of the encoders trained with these three models to Llama-8B in Table 4. Performance improves with more preserved early lay-



Encoder fine-tuned by small model $\mathcal{T}(6, 30)$ → Llama 8B

Zero-shot grafting: failure

It seems like you're having a bit of fun with the text! To answer your question, the image you're referring to is not actually an image, but rather a text prompt. I'm here to help you with any questions or topics you'd like to discuss. If you'd like to start fresh, feel free to ask me anything, and I'll do my best to help!

Encoder fine-tuned by small model $\mathcal{T}(12, 30)$ → Llama 8B

Zero-shot grafting: success

The cat is sitting on a colorful, hand-painted canvas, which is likely a wall or a piece of art. The cat appears to be looking directly at the viewer with a curious expression, as if it is observing its surroundings.

Encoder fine-tuned by small model $\mathcal{T}(17, 30)$ → Llama 8B

Zero-shot grafting: success

The cat is sitting on a colorful, striped cushion or couch, looking directly at the viewer with its eyes. It appears to be relaxed and calm, possibly enjoying the comfortable seating.

Encoder fine-tuned by Llama 8B → Llama 8B

The cat is sitting on a yellow couch, looking directly at the camera.

Pre-trained Encoder → fine-tuned Llama 8B

The cat is sitting on a couch, which is located in a room.

Figure 6. **Qualitative results on zero-shot grafting capability** of encoders trained with surrogate models for Llama-8B. For comparison, we also include responses from the encoder trained with Llama-8B and the fine-tuned Llama-8B. Responses are sampled with greedy decoding. More early-phase layers preserved lead to stronger zero-shot grafting capability. A → B means plugging A into B.

small model	mmlu	hellaswag	arc-easy	arc-challenge	winogrande	piqa	boolq	openbookqa
Llama-8B	68.4	80.5	79.8	61.8	77.3	81.5	85.4	44.8
$\mathcal{T}(6, 30)$	25.5	31.8	36.8	24.7	50.9	58.8	61.3	25.8
$\mathcal{T}(12, 30)$	25.4	42.9	40.5	29.0	59.6	62.6	69.4	29.6
$\mathcal{T}(17, 30)$	66.8	61.2	59.3	44.8	70.9	71.0	69.3	34.2

Table 3. **Accuracy (%) of small models** for Llama-8B on text benchmarks.

encoder fine-tuned by	MME _{perc} _{binary}	POPE _{binary}	SEED-Bench	MMVet	LLaVA-Wild	MMB _{en}	performance
Llama-8B	1165	84.7	57.5	23.2	47.6	44.9	
model $\mathcal{T}(6, 30)$	583	73.3	25.8	8.9	22.4	8.6	
zero-shot grafting	767	20.6	30.9	13.4	28.1	-	↓
model $\mathcal{T}(12, 30)$	983	77.9	26.9	13.6	29.1	0.43	
zero-shot grafting	1022	81.7	50.7	20.5	47.9	45.4	↑
model $\mathcal{T}(17, 30)$	1041	81.3	55.4	20.9	42.0	49.7	
zero-shot grafting	1044	83.4	56.1	25.2	56.8	53.5	↑

Table 4. **Accuracy (%) of encoders fine-tuned by small models** for Llama-8B on VLM benchmarks.

ers, showing particularly strong results when retaining all early-phase layers before the transition point in $\mathcal{T}(17, 30)$.

Figure 6 depicts a qualitative example demonstrating the zero-shot grafting capability of three trained encoders. As expected, the encoder trained by the model $\mathcal{T}(6, 30)$ fails to generate readable image features for Llama-8B as most of the early-phase layers are removed. The encoder trained by $\mathcal{T}(12, 30)$ performs better, but its image features lack fine-grained detail (e.g., no couch in response), which explains the zero-shot performance gap in Table 4. The en-

coder trained by $\mathcal{T}(17, 30)$ generates more detailed and accurate image features, achieving the best zero-shot grafting response among the three, which covers the cat’s color, expression, eye direction, position, the presence of a couch, and even the atmosphere.

In summary, our entire analysis of the prediction trajectory reveals:

- The early phase^a plays a pivotal role in the encoder’s transferability to the target LLM.
- Retaining the original parameters of the early phase is critical for maintaining the encoder’s zero-shot grafting capability.
- The transition point in Figure 3 is a good starting point for late-phase removal and translator insertion.

^aWe believe the early phase of LLMs has potential beyond building surrogate models, enabling more creative applications.

Based on these three key findings, now we define the model $\mathcal{T}(16, 26)$ as our *surrogate model* for Llama-3B, the model $\mathcal{T}(17, 30)$ as our surrogate for Llama-8B, by inheriting the early-phase layers and replacing the late-phase layers with a translator, which can be fine-tuned with a small set of text instructions, e.g., 500K for one epoch.

2. Generalizing to Giant Models

Having validated our approach at relatively small scales, we now expand our methodology to construct surrogate models for giant LLMs – Llama-70B. In this section, our experiments demonstrate two key advantages of our surrogate approach: First, surrogates can bring a strong zero-shot grafting ability to encoders, enabling them to trigger

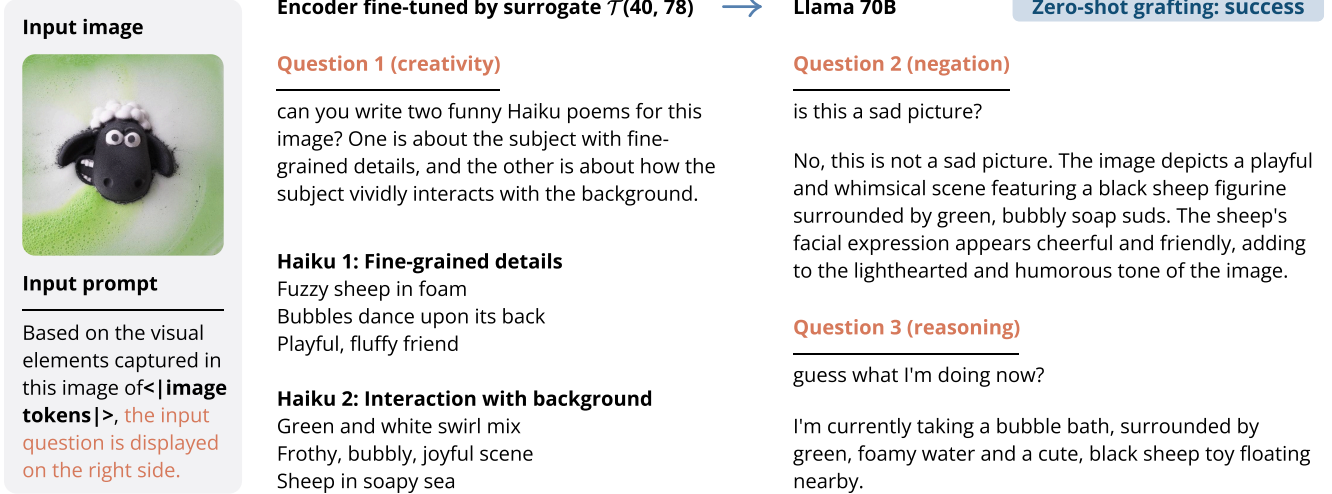


Figure 7. **Qualitative results on strong zero-shot grafting ability** of surrogate-trained encoder for Llama-70B, which produces fine-grained image features to trigger Llama-70B to perform complex visual understanding tasks. Response is sampled with greedy decoding.

target LLMs to perform visual understanding tasks without additional training. Second, training target LLM decoders on surrogate-trained encoders significantly reduces cost by providing a warm start for fine-tuning.

2.1. A Surrogate for Llama-70B

We analyze the prediction trajectory for Llama-70B in Figure 3 to identify the transition point that marks the end of token processing, which occurs around layer $\ell = 40$. Then we keep the first ($\ell = 0$) and last layer ($\ell = 79$), insert a translator at $\ell = 40$, and remove the late phase from $\ell = 41$ to $\ell = 78$, to build a 37B surrogate $\mathcal{T}(40, 78)$. Text benchmark results of this surrogate are shown in Table 5, and VLM benchmark results in Table 6. Table 6 shows the performance of the encoder trained using surrogate $\mathcal{T}(40, 78)$ on VLM benchmarks, highlighting a significant improvement through zero-shot grafting. These experiments show that our approach can be scaled up to giant models, holding the same principles of early phase preservation.

2.1.1. Results: Zero-shot Grafting

In Table 7, with encoder-only training, our surrogate outperforms the full-size Llama-70B on most VLM benchmarks, except for VisWiz. This demonstrates the effectiveness of our surrogate models. The last row shows the performance of zero-shot grafting the surrogate-trained encoder into Llama-70B. Notably, the performance of zero-shot grafting surpasses the full-size Llama-70B decoder training on some benchmarks by a big margin, demonstrating that our surrogate-trained encoder effectively prompts LLaMA-70B to handle complex visual understanding tasks.

Figure 7 presents qualitative results showcasing the strong zero-shot grafting capability of our surrogate-trained encoder, including questions about creativity, negation, and

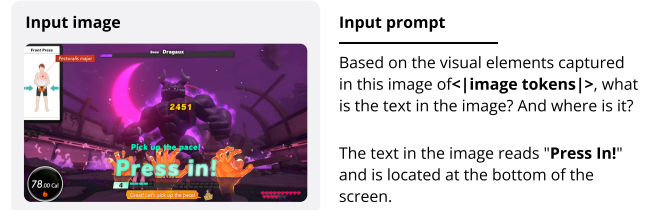


Figure 8. **Qualitative OCR results on strong zero-shot grafting ability** of surrogate-trained encoder for Llama-70B. The input image size is 336^2 .

	mmlu	hellaswag	arc-easy	arc-challenge	wino-grande	piqa	boolq	openbookqa
small model								
Llama-70B	82.6	86.9	83.4	71.2	85.4	83.7	89.1	47.6
surrogate-37B	80.8	70.4	67.3	56.6	77.9	73.9	86.9	37.8

Table 5. **Accuracy(%) of surrogate model** for Llama-70B on text benchmarks.

encoder fine-tuned on	MME _{Perceptual}	POPE _{Binary}	SEED-Bench	MMVet	LLaVA-Wild	MMB ^{en}	performance
Llama-70B	1294	83.4	59.8	27.0	45.6	58.8	
surrogate-37B	1287	84.8	64.2	29.6	54.2	59.5	
zero-shot grafting	1315	86.1	64.1	37.4	59.7	60.7	↑

Table 6. **Accuracy (%) of encoder fine-tuned by surrogate** for Llama-70B on VLM benchmarks.

reasoning. Additionally, Figure 8 demonstrates its effectiveness on OCR tasks, showing that our surrogate models are able to squeeze robust and detailed visual information into encoders.

While the surrogate-trained encoder enables zero-shot conversion of the giant LLM into a VLM, its performance still

training method	MME _{binary}		MME		POPE _{binary}		POPE		SEED-Bench			MM	LLaVA	MMB	CV-Bench			GQA	Vis-Wiz
	cog	perc	cog	perc	acc.	f1	acc.	f1	all	img	vid	-Vet	-Wild	en	2d	3d	avg		
Llama-70B decoder	327	1545	345	1524	84.9	83.1	84.8	82.9	63.6	68.9	43.7	35.5	67.5	71.8	61.8	73.3	67.5	62.4	53.0
Llama-70B encoder	285	1294	288	1321	83.4	82.6	82.7	81.2	59.8	65.4	38.6	27.0	45.6	58.8	62.2	59.4	60.8	54.4	47.4
surrogate-37B encoder	312	1329	291	1250	85.5	83.9	86.3	85.0	65.9	71.1	46.2	28.8	54.3	63.1	64.7	64.0	64.3	56.5	22.7
zero-shot grafting	295	1348	303	1298	86.8	86.1	87.0	86.4	65.4	70.7	45.3	32.8	68.9	65.6	63.2	67.2	65.2	51.9	40.0

Table 7. **Accuracy (%)** for Llama-70B on VLM benchmarks. The **bold** numbers indicate the best performance between the full-size decoder training and our surrogate-trained encoder by zero-shot grafting. A special clarification for LLaVA-Wild is in Sec. A.9.

method	X%	avg. score	MME _{binary}		MME		POPE _{binary}		POPE		SEED-Bench			MM	LLaVA	MMB	CV-Bench			GQA	Vis-Wiz
			cog	perc	cog	perc	acc.	f1	acc.	f1	all	img	vid	-Vet	-Wild	en	2d	3d	avg		
baseline	10	0.5127	301	1178	333	1139	74.5	73.1	74.3	73.9	46.0	49.2	34.1	17.7	30.4	45.6	52.4	61.7	57.1	45.4	43.9
	20	0.5153	255	1061	277	1116	71.4	61.1	71.6	61.3	52.2	55.5	39.5	20.6	41.3	58.3	56.6	66.8	61.7	47.5	38.5
	30	0.6277	314	1447	316	1399	85.5	84.8	85.5	84.8	59.8	64.8	40.9	30.8	57.9	66.8	60.6	71.1	65.9	57.1	55.1
	60	0.6444	353	1511	358	1515	84.8	83.1	84.4	82.4	62.4	67.9	41.5	32.4	64.5	70.9	61.5	72.2	66.8	61.3	48.1
	100	0.6538	327	1545	345	1524	84.9	83.1	84.8	82.9	63.6	68.9	43.7	35.5	67.5	71.8	61.8	73.3	67.5	62.4	52.9
grafting	-	0.6259	295	1348	303	1298	86.8	86.1	87.0	86.4	65.4	70.7	45.3	32.8	68.9	65.6	63.2	67.2	65.2	51.9	40.0
ours	10	0.6612	340	1404	342	1430	87.3	86.8	84.9	82.7	67.1	72.7	45.9	37.6	69.7	70.9	66.6	70.6	68.6	60.1	57.9
	20	0.6701	369	1435	361	1486	86.7	85.6	86.4	84.9	67.2	72.8	46.0	38.8	70.5	73.2	65.7	74.8	70.3	60.6	52.4
	30	0.6704	374	1449	349	1490	87.9	87.7	86.8	85.5	67.0	72.0	47.8	38.9	69.3	73.9	66.6	72.8	69.7	61.4	49.2

Table 8. **Convergence comparison** with using X% of training data between baseline and our surrogate training approach for Llama-70B decoder training. The gray row indicates the results (training hours) reported in the Table 9 with 20% of training data for ours.

lags behind that of full-size decoder training. What benefits can we expect from this surrogate-trained encoder? Next, we demonstrate that it can accelerate training convergence and improves the performance of full-size decoder training.

2.2. Reducing Full Decoder Training Cost

In the previous sections, we conduct the experiments with a two-stage training strategy, where we simultaneously train the vision adapter in encoder and the translator in decoder during the first stage, and then fine-tune the encoder atop the surrogate in the second stage. Currently, we are interested in training the full-size decoder, which is the final third training stage. First, we introduce the training setup, and recipes are introduced in Sec. A.5.

Models. As in previous sections, we use the CLIP-L/14 encoder with an input image size of 336^2 . The vision adapter is a two-layer MLP, consisting of consecutive linear layers with a GELU activation in between. Notably, we maintain a fixed vision adapter size across all model scales, unlike prior works [6, 23] that scale it with model size. This design choice ensures that variations in adapter size do not introduce unknown effects on the encoder’s zero-shot grafting capability, allowing for a controlled initial study. For state-of-the-art performance, however, the vision adapter can be scaled up with the model size. The decoders are our surrogate-37B, i.e., $\mathcal{T}(40, 78)$, and full-size Llama-70B.

Data. In the third training stage, the training data is still the same as in the previous two stages – the LLaVA-1.5-665K [26] instructions (without text-only samples). This choice is based on the following considerations:

1) The first training stage focuses on the adapter and translator. Commonly, vision adapters are trained on captions in-

stead of instructions, but we found no significant difference in experimental outcomes. Thus, to simplify training, we merge the training of the vision adapter and translator into a single stage using vision-language and text-only instructions. When forwarding the text-only instructions, gradients backpropagated to the vision adapter are zero.

2) The second stage trains encoders with surrogates, aiming to efficiently compress data into encoders while preparing to transfer knowledge to the full-size decoder. To ensure consistency, we use the same training data for the second and third stages. It is recommended to use larger and more diverse datasets for those two stages.

2.2.1. Results: Convergence and Training Cost

In Table 8, we compare performance of the typical baseline method and our surrogate training approach across different percentages of training data used for training decoders. The baseline trains Llama-70B with the original CLIP encoder, while ours trains it with our surrogate-trained encoder (the third row in Table 7). First, the gray row represents the performance of zero-shot grafting the surrogate-trained encoder to Llama-70B, which nearly matches the baseline with 30% of the data. Second, after training on just 10% of data, our approach achieves nearly the same performance as the baseline with 100% of the data, except for MME. For other benchmarks, our 10% performance even outperforms the final baseline result. With continued training, performance remains unchanged, suggesting saturation after 20% of the data. In Figure 2, we plot the normalized average score of each X% data utilization for our method and baseline.

We also visualize Table 9 in Figure 2 for a direct comparison of training hours for each training stage. First, our pre-

training time is longer than the baseline because we train both the vision adapter and the translator with additional text instructions. However, the key advantage of our surrogate training approach is seen in the decoder training, which is the real bottleneck in common methods. With 20% of the data used for our decoder training, we achieve a training cost reduction of $\sim 45\%$. This reduction is the minimum, as our performance in the 10% case already exceeds the final result of the baseline, as shown in Table 8.

method	# gpu	pre.	ft. enc.	ft. dec.	total hours
baseline	128	7.01	0.00	27.88	34.79
ours	128	9.36	4.25	5.56	19.17

Table 9. **Training hours comparison** between baseline and our surrogate approach for training VLMs with Llama-70B, including the time for pretraining (pre.), fine-tuning encoder (ft. enc.), and fine-tuning decoder (ft. dec.). Checkpoint loading and saving times are excluded. More details in Sec. A.4.

3. Related Work Overview

Understanding LLMs is a key topic in mechanistic interpretability [35]. [1] uses linear classifiers (*probes*) to understand the dynamics of intermediate layers in neural networks. For LLMs, [32] directly employs the output embedding matrix as a probe to classify layer-wise representations, illustrating how input tokens shift from current positions to next ones. Tuned Lens [3] extends this idea with a trainable probe for broader applicability to modern LLMs. [36] conceptualizes transformer layers as “painters” that iteratively refine representations and suggests that middle layers share the same representation. In contrast, we identify two distinct transition phases in LLMs.

The shared representation in middle layers suggests redundancy. [36] further concludes that some middle layers can be removed without a significant performance drop. **Pruning LLMs** largely is based on such insight of redundancy. Notably, both [12] and [30] found that deep layers are not essential and can be removed. Interestingly, our surrogate models also replace deep layers in the late phase. However, our method differs in how we identify the transition point and in our objective. Unlike pruning, which aims to remove layers while preserving performance, our focus is on the efficiency of surrogate models for encoder transferability. While our surrogate models consistently underperform compared to their target LLMs, they serve a distinct purpose in producing efficient encoders for VLMs.

Our surrogate-trained encoders can directly prompt target LLMs to generate the expected responses without any fine-tuning. This zero-shot grafting ability aligns with the concept of **steering LLMs**, a lightweight alternative to fine-tuning LLMs [14, 19]. Prior works show that language models can be guided to perform specific tasks without extensive fine-tuning. Similarly, in our case, image features

from surrogate-trained encoders act as steering tokens, enabling target LLMs to interpret visual content and answer various complicated questions.

This capability provides a warm start for further decoder fine-tuning, helping to mitigate the **expensive training cost of VLMs** [2, 6, 24, 39, 41, 42]. The costs surged as decoder sizes scale from relatively small models (3B, 8B) to much larger ones, such as 70B [23], 110B [21, 27]. Additionally, increasing the number of image tokens for high-resolution inputs further escalates the computational burden. LoRA [16] could be applied for training VLMs. While LoRA improves efficiency, it underperforms full fine-tuning, especially in giant LLMs, when applied with small rank (e.g., 8) and alpha (e.g., 32) to query and key decoder layers – a common practice in LLM training. Closing this gap needs applying LoRA to *entire* transformer layers with large rank and alpha (e.g., rank 128 with alpha 256 as in [23] for 13B decoder training). Then LoRA takes about the same time as full decoder fine-tuning. This limitation likely explains why current VLMs still rely on full decoder fine-tuning. Critically, contrasting to our surrogate training approach, LoRA does not accelerate convergence. See more in Sec. A.2.

Additionally, the idea of using small models to train encoders before applying them to larger decoders has been depicted in [18]. However, this work is not directly related to ours, as it employs a progressive multi-stage training strategy to just scale up model size and refine image processing from coarse to fine. No further details are provided on this method, leaving it unclear how it reduces costs. In contrast, our approach provides a well-defined framework for constructing efficient surrogate models specifically tailored for any target LLM. Plus, we plug the surrogate-trained encoders directly into their target LLMs, converting them into VLMs without any fine-tuning to perform complex visual understanding tasks. Further, with our surrogate-trained encoders, the decoder needs only a few full-scale fine-tuning steps to achieve the desired performance.

4. Conclusion

In this work, we show that vision encoders trained with our surrogate models can accelerate VLM training. We also note that our surrogate models are not limited to vision encoders. The main limitation of our approach is the need for a well-designed surrogate, which ideally should be small. Although our layer-dropping strategy works in principle for any LLM, resulting models are still half the size of their target LLMs, for example, our surrogate-37B for Llama-70B. This underscores the practical value of surrogate models and highlights the need for ways to create them more efficiently and with better compression.

A. Appendix

A.1. 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.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 only modifies a small number of parameters, it struggles 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 with 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. The common solution is to mix the text corpus with vision-language instructions during training to mitigate this issue. For our surrogate training approach, we are interested in whether it could help alleviate this issue.

model	mmlu	hellaswag	arc-easy	arc-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.1. **Accuracy(%) of 70B decoders** from the final training stage of our surrogate approach (ours) and baseline method on text benchmarks.

In Table A.1, 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.

A.4. 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 [46]. 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 EFA⁸. 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.5. 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

⁸Introduction of Amazon Elastic Fabric Adapter (EFA).

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 scenarios 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.2.

A.6. Evaluation Benchmarks

We evaluate VLMs following LLaVA-1.5 [26], including MME [10], POPE [25], SEED-Bench [20], MM-Vet [43], LLaVA-in-the-Wild [28], MMBench [29], GQA [17], and Viz-Wiz [13] using `lmms-eval` toolkit [22]. Our evaluation setting also contains a vision-centric benchmark – CV-Bench [39]. We evaluate LLMs and our surrogate language models on MMLU [15], HellaSwag [44], ARC [8], Winogrande [34], PIQA [4], BoolQ [7], and OpenBookQA [31] 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.3.

	mmlu	hellaswag	arc _{easy}	arc _{challenge}	wino _{grande}	piqa	boolq	openbookqa
number of shots	5	10	0	25	5	0	0	0
acc. type	-	norm	norm	norm	-	norm	-	norm

Table A.3. **Few-shot setting** for text benchmarks in `lm-harness`. For accuracy type, “norm” refers to length-normalized accuracy.

A.7. 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:

- Their training costs are already affordable nowadays.
- 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 short answers (e.g., “yes” or “no” in MME and POPE) or single words/phrases (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 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, 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 insufficient 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. Assume 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.1 illustrates how this weighting adjusts based on the maximum response length in a batch. In our exper-

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($\beta_1 = 0.9, \beta_2 = 0.999, \text{eps} = 1\text{e-}8$)		
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.2. Training hyper-parameters, recipes and settings.

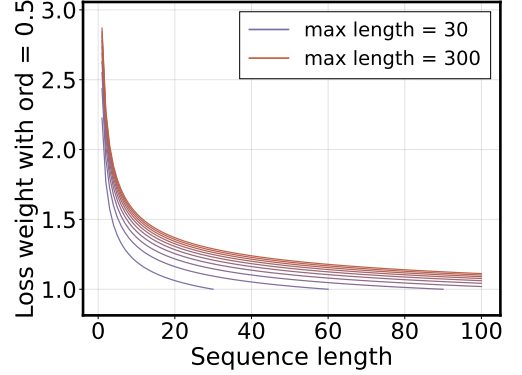


Figure A.1. **Dynamic loss weights** for balancing loss contributions from different response lengths in a global batch. Multiple curves represent different maximum response lengths in a batch, gradually increasing from 30 to 300.

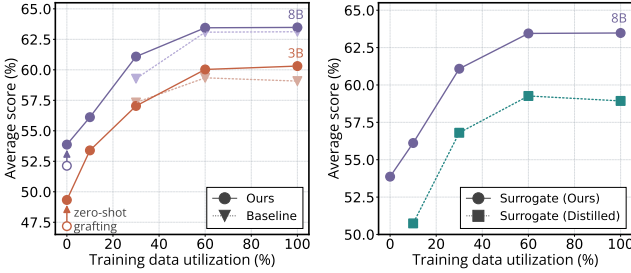


Figure A.2. **Training cost comparison** among our surrogate training approach, baseline method, and training with distill models as surrogates for Llama-3B and 8B.

iments, we set $\text{ord} = 0.5$. 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.2. 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.8. Distilled Model as Surrogate

In some LLM families, such as Llama [9] and Gemma [37], 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 surrogate 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.2, we compare the performance of the distilled model as a surrogate with our surrogate training approach using the same training setup, including the $\text{ord} = 0.5$ 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.9. Performance Drop on LLaVA-Wild

Our models and the baseline model both score lower on the LLaVA-in-the-Wild [28] than the official scores reported in LLaVA-1.5 [26], 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.md⁹ in `lmms-eval` repository.

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

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? 1 is larger than 10, is it correct? Happy birthday to you! I love you.	The cat is larger than the dog. No, 10 is not larger than 1. Happy birthday to you! I'm glad to hear that. I'm here to help you with any questions or tasks you may have.	The dog is larger. The statement is not true. The number 1 is less than the number 10. Thank you for the birthday greetings! I am happy to celebrate with you. 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 $100 + 13 = ?$ Calculate $2 * 10.2 = ?$ Calculate $1 + 10 = ?$ Is $8.11 > 8.3$?	$100 + 13 = 113$ $2 * 10.2 = 20.4$ $1 + 10 = 11$ No, 8.11 is not greater than 8.3.	$100 + 13 = 125$ $2 * 10.2 = 22.4$ $1 + 10 = 20$ Yes, $8.11 > 8.3$.

Table A.4. **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.

A.10. 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.

In Table A.4, 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:

- Common sense reasoning, factual knowledge, and conversational abilities are primarily stored in the early-phase parameters.
- 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.

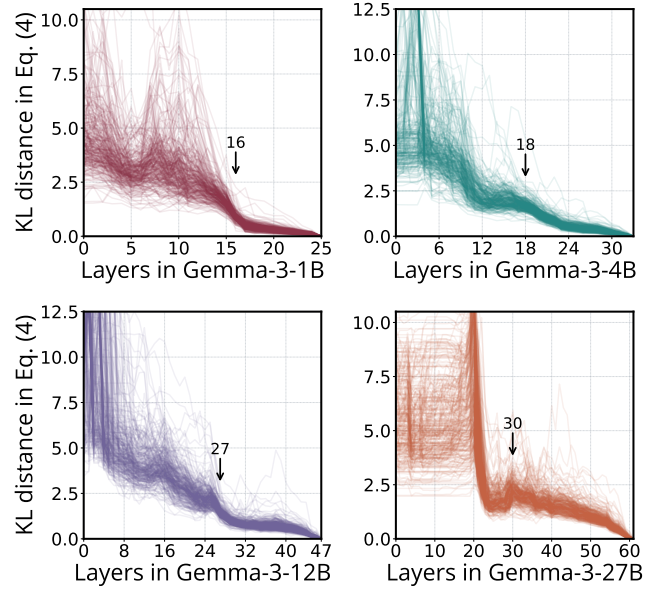


Figure A.3. **The trajectory of prediction** across different layers of 1B, 4B, 12B, and 27B instruct models from Gemma-3. The arrow marks the transition point where the trajectories of 300 random samples converge.

A.11. Prediction Trajectory of Gemma-3 Family

In Figure A.3, we plot the prediction trajectory across different layers of Gemma-1B, 4B, 12B, and 27B instruct models from Gemma-3 [38] using 300 random sequences. Despite Gemma-3 using sliding window attention, its prediction trajectory follows a similar pattern to Llama-3. The trajectory converges 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.

We believe Eq. 4 generalizes across different LLM families and that our surrogate training approach can be applied to Gemma-3 using the same procedure as Llama-3.

Acknowledgements

We sincerely appreciate Meta’s support in providing GPUs for our experiments. This work was supported by DAPRA TIAMAT, the ONR MURI program, the National Science Foundation (IIS-2212182), and the NSF TRAILS Institute (2229885). Commercial support was provided by the Amazon Research Award program, Open Philanthropy, and Capital One Bank.

Disclaimer

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

References

- [1] Guillaume Alain and Yoshua Bengio. Understanding Intermediate Layers Using Linear Classifier Probes. In *ICLR*, 2017. 8
- [2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a Visual Language Model for Few-Shot Learning. In *NeurIPS*, 2022. 8
- [3] Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting Latent Predictions From Transformers With the Tuned Lens. *arXiv preprint arXiv:2303.08112*, 2023. 3, 8
- [4] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about Physical Commonsense in Natural Language. In *AAAI*, 2020. 10
- [5] Jiahui Chen, Rifaa Qadri, Yuxin Wen, Neel Jain, John Kirchenbauer, Tianyi Zhou, and Tom Goldstein. GenQA: Generating Millions of Instructions from a Handful of Prompts. *arXiv preprint arXiv:2406.10323*, 2024. 2, 3, 11
- [6] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. InternVL: Scaling up Vision Foundation Models and Aligning for Generic Visual-Linguistic Tasks. In *CVPR*, 2024. 7, 8
- [7] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *ACL Anthology*, 2019. 10
- [8] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018. 10
- [9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024. 1, 11
- [10] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. MME: A Comprehensive Evaluation Benchmark for Multimodal Large Language Models. *arXiv preprint arXiv:2306.13394*, 2023. 3, 10
- [11] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A Framework for Few-Shot Language Model Evaluation, 2024. 10
- [12] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Gloriosio, and Daniel A Roberts. The Unreasonable Ineffectiveness of the Deeper Layers. In *ICLR*, 2025. 8
- [13] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. VizWiz Grand Challenge: Answering Visual Questions from Blind People. In *CVPR*, 2018. 10
- [14] Chi Han, Jialiang Xu, Manling Li, Yi Fung, Chenkai Sun, Nan Jiang, Tarek Abdelzaher, and Heng Ji. Word Embeddings Are Steers for Language Models. In *ACL Anthology*, 2024. 8
- [15] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *ICLR*, 2021. 10
- [16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*, 2022. 8, 9
- [17] Drew A Hudson and Christopher D Manning. GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. In *CVPR*, 2019. 10
- [18] InternVL2. InternVL2: Better than the Best—Expanding Performance Boundaries of Open-Source Multimodal Models with the Progressive Scaling Strategy. <https://internvl.github.io/blog/2024-07-02-InternVL-2.0/>, 2024. 8
- [19] Md Kowsher, Nusrat Jahan Prottasha, and Prakash Bhat. Propulsion: Steering LLM with Tiny Fine-Tuning. In *ACL Anthology*, 2024. 8
- [20] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. SEED-Bench: Benchmarking Multimodal LLMs with Generative Comprehension. In *CVPR*, 2024. 10
- [21] Bo Li, Kaichen Zhang, Hao Zhang, Dong Guo, Renrui Zhang, Feng Li, Yuanhan Zhang, Ziwei Liu, and Chunyuan Li. Llava-next: Stronger llms supercharge multimodal capabilities in the wild. <https://llava-vl.github.io/blog/2024-05-10-llava-next-stronger-llms/>, 2024. 8
- [22] Bo Li, Peiyuan Zhang, Kaichen Zhang, Fanyi Pu, Xinrun Du, Yuhao Dong, Haotian Liu, Yuanhan Zhang, Ge Zhang, Chunyuan Li, and Ziwei Liu. LMMs-Eval: Accelerating the Development of Large Multimodal Models. <https://>

- [//github.com/EvolvingLMs-Lab/lmms-eval](https://github.com/EvolvingLMs-Lab/lmms-eval), 2024. 10
- [23] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. LLaVA-OneVision: Easy Visual Task Transfer. *arXiv preprint arXiv:2408.03326*, 2024. 7, 8
 - [24] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *ICML*, 2023. 8
 - [25] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating Object Hallucination in Large Vision-Language Models. In *EMNLP*, 2023. 3, 10
 - [26] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved Baselines with Visual Instruction Tuning. In *CVPR*, 2024. 3, 7, 10, 11
 - [27] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge. <https://llava-vl.github.io/blog/2024-01-30-llava-next/>, 2024. 8
 - [28] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In *NeurIPS*, 2024. 10, 11
 - [29] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. MMBench: Is Your Multi-modal Model an All-around Player? In *ECCV*, 2025. 10
 - [30] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in Large Language Models are More Redundant Than You Expect. *arXiv preprint arXiv:2403.03853*, 2024. 8
 - [31] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *ACL Anthology*, 2018. 10
 - [32] nostalgebraist. interpreting GPT: the logit lens. <https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens>, 2020. 8
 - [33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, 2021. 1
 - [34] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. In *ACM Communications*, 2021. 10
 - [35] Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, et al. Open Problems in Mechanistic Interpretability. *arXiv preprint arXiv:2501.16496*, 2025. 8
 - [36] Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer Layers as Painters. *arXiv preprint arXiv:2407.09298*, 2024. 8
 - [37] Gemma Team. Gemma-2. *Kaggle*, 2024. 11
 - [38] Gemma Team. Gemma-3. *Kaggle*, 2025. 12
 - [39] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A Fully Open, Vision-Centric Exploration of Multimodal LLMs. In *NeurIPS*, 2025. 8, 10
 - [40] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. SigLIP 2: Multilingual Vision-Language Encoders with Improved Semantic Understanding, Localization, and Dense Features. *arXiv preprint arXiv:2502.14786*, 2025. 1
 - [41] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing Vision-Language Model’s Perception of the World at Any Resolution. *arXiv preprint arXiv:2409.12191*, 2024. 8
 - [42] Weihao Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. CogVLM: Visual Expert for Pretrained Language Models. In *NeurIPS*, 2024. 8
 - [43] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. MM-Vet: Evaluating Large Multimodal Models for Integrated Capabilities. In *ICML*, 2024. 10
 - [44] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In *ACL Anthology*, 2019. 10
 - [45] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid Loss for Language Image Pre-Training. In *ICCV*, 2023. 1
 - [46] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023. 9