DeepInsert: Early Layer Bypass for Efficient and Performant Multimodal Understanding

Moulik Choraria[1,*,*] **Xinbo Wu**[1,*] **Akhil Bhimaraju**[1,*]

Nitesh Sekhar[2] **Yue Wu**[3] **Xu Zhang**[2] **Prateek Singhal**[4]

Lav R. Varshney[5]

[1]UIUC, [2]Amazon, [3]Capital One, [4]Apple, [5] Stony Brook University

Abstract

Hyperscaling of data and parameter count in LLMs is yielding diminishing improvement when weighed against training costs, underlining a growing need for more efficient finetuning and inference without sacrificing performance. This is especially so for multimodal language models (MLMs), where the overhead of processing multimodal tokens can limit their practical viability. Parallely, recent work has uncovered implicit cross-modal alignment in the deeper layers of large MLMs, deepening our understanding of how MLMs process and encode information. Motivated by this, and our observation that MLMs naturally defer most cross-modal token interactions to deeper layers of the model, we propose a simple modification. Instead of concatenation with the language prompt at the start, we insert multimodal tokens directly into the middle, allowing them to entirely bypass the early layers. Our results with diverse modalities, (i) LLaVA & BLIP for vision, (ii) LTU for audio, and (iii) MoLCA for molecular data, and model sizes, starting from 350M to 13B parameters, indicate that our method reduces both training and inference costs, while at least preserving, if not surpassing the performance of existing baselines.

1 Introduction

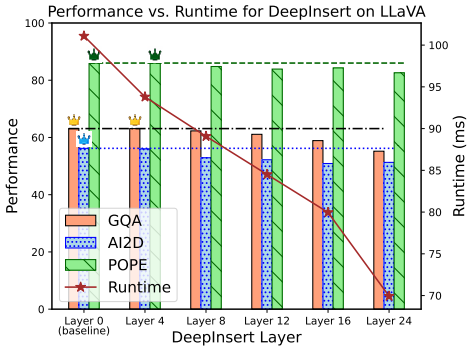
Advances in large language models (LLMs) (Brown et al., 2020; Chung et al., 2022; Touvron et al., 2023; OpenAI et al., 2024) have spurred the development of models capable of multimodal comprehension in diverse application domains. While scaling has been crucial for this success (Kaplan et al., 2020; Zhai et al., 2021; Hoffmann et al., 2022), the computational costs associated with large multimodal language models (MLMs/MLLMs) (Strubell et al., 2019; Grattafiori et al., 2024) necessitates methods for efficient finetuning and inference (Hu et al., 2021; Leviathan et al., 2023). For multimodal tasks such as vision-language understanding (Radford et al., 2021; Li et al., 2022; Liu et al., 2024c), the use of pretrained unimodal models, rather than training from scratch, is effective at reducing training overhead (Qi et al., 2024). However, overall efficiency remains a challenge, as multimodal inputs impose additional computational costs, when processed alongside language prompts.

Another key to multimodal training is effective cross-modal alignment, which is achieved with a combination of pretraining stages, followed by multitask instruction finetuning for downstream tasks (Dai et al., 2023). Preference tuning techniques can be used to produce further improvements (Ouyang et al., 2022; Sun et al., 2023b; Wang et al., 2024). However, most works in this area tend to propose modifications to the training algorithm and/or choice of dataset, while largely treating the pretrained components as replaceable black-box models. To complement this body of work from the efficiency point of view, we begin our investigation by analyzing the internal representations, relying on pretrained VLMs as a starting point.

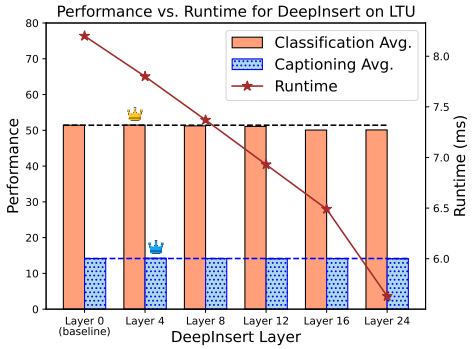
Our initial inspiration comes from Huh et al. (2024), who argue that representations from different unimodal models converge towards a universal platonic representation. Although the platonic hypothesis itself is contentious, the underlying evidence suggests that model representations across modalities show increasing alignment in the deeper layers (see Figs. 4, 5 in the Appendix). This, coupled with the inner product mechanism of self-attention, invites the next question: in a multimodal setting, are cross-modal interactions more confined to deeper layers? The attention maps indeed suggest that the bulk of the activity is within the middle and latter layers (see Fig. 2). The preceding exposition culminates in our central investigation: *Are*

*Correspondence: moulikc2@illinois.edu; *Equal contribution; Code: https://github.com/MoulikChoraria/DeepInsert

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7059–7079
March 24-29, 2026 ©2026 Association for Computational Linguistics



(a) Performance vs runtime in LLaVA (vision)



(b) Performance vs runtime in LTU (audio)

Figure 1: Tradeoff between performance and computational efficiency (inference) in (a) LLaVA v1.5-7B and (b) LTU-7B highlight our two contributions. First, we can reduce multimodal processing (efficiency gains), while either maintaining or improving performance with DeepInsert (see layer 4). Second, it is possible to stay competitive with the baseline, while having significant efficiency gains ($\sim$ 20–25% speedup, see layer 12).

the early MLLM layers necessary for multimodal token processing, and can we skip them without sacrificing performance? More specifically,

1. We demonstrate redundancy in early layers of multimodal LLMs, and that letting multimodal tokens bypass them **entirely** offers significant efficiency gains, with virtually no degradation. We refer to this framework as DeepInsert.

2. We showcase DeepInsert's versatility across modalities, via popular open-source MLLMs: vision (LLaVA, BLIP (Liu et al., 2024a; Li et al., 2023a)), audio (LTU (Gong et al., 2024)), and molecules (MolCA (Liu et al., 2024e)). We also reveal a natural performance-efficiency tradeoff (see Fig. 1), enabling practitioners to select models based on their specific needs.

The remainder of the paper is organized as follows. Sec. 2 places our work in the context of broader literature. Sec. 3 motivates our framework, while Sec. 4 details the general notion of the architectural modification, highlighting implementation and design considerations. Experiments in Sec. 5 demonstrate the versatility and effectiveness across a variety of popular open-source multimodal models. Sec. 6 concludes with promising directions for broader application.

2 Related Work

2.1 MLLMs & Alignment

The general architecture of MLLMs considered here comprises a pretrained LLM, frozen perceptual encoders for different modalities (image, audio, video, etc. (Gong et al., 2021; Radford et al., 2021; Sun et al., 2023a; Zhao et al., 2024)), and a trainable mapping module (can be a lightweight MLP or a larger Perceiver (Jaegle et al., 2021)) that aligns the modality with the LLM input space. The LLM prompts consist of text and multimodal tokens, the latter of which are obtained from the encoder via the mapping module. This general regime encompasses a wide range of works (Alayrac et al., 2022; Li et al., 2023a; Zhang et al., 2023; Bai et al., 2023; Liu et al., 2024a; Gong et al., 2024; Ghosh et al., 2024; Microsoft et al., 2025), and thus, DeepInsert is fairly universally applicable.

Based on where alignment is achieved, our method relates best to the idea of model stitching (Lenc and Vedaldi, 2015; Moschella et al., 2023), that is, *grafting* initial layers of a neural network onto the latter half of another with an affine layer. Our work can thus be interpreted as multimodal model stitching for efficiency, without any additional bells and whistles.

2.2 Efficiency via parameter reduction

Low-rank adapters (LoRA) (Hu et al., 2021; Dettmers et al., 2023) are commonly used for computationally-efficient finetuning of LLM-based models. The low-rank constraint enables finetuning large models with only a small set of additive parameters, reducing memory usage. However, inference remains just as costly. An alternative is model compression, where an existing model is effectively distilled, leading to accelerated inference and almost-at-par capabilities (Fang et al., 2021; Wang et al., 2022). While these techniques exploit

parametric redundancies in LLMs, our approach is motivated via functional redundancy, and therefore largely complementary.

2.3 Layer skipping & Token reduction

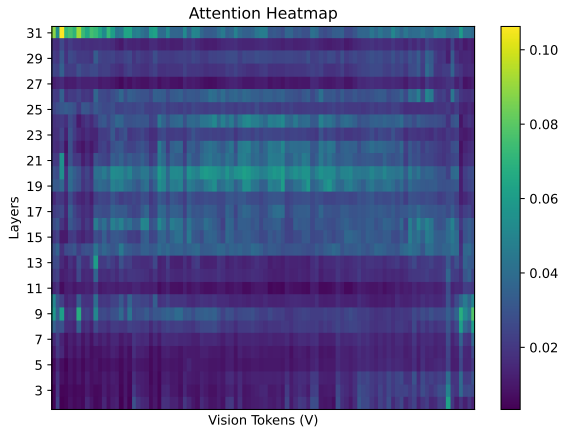
Layer skipping, which has been employed in LLMs for speculative decoding and speeding up inference (Corro et al., 2023; Elhoushi et al., 2024; Din et al., 2024), has been recently applied to MLLMs (Shukor and Cord, 2024; Zeng et al., 2025). Specifically, Shukor and Cord (2024) combine layer skipping with compression and pruning of the base LLMs to demonstrate impressive efficiency gains in their custom setup. However, it is unclear if their method scales well as it falls $10 - 20\%$ short of baseline performance when trained in the LLaVA setup. On the other hand, Zeng et al. (2025) propose adaptive token skipping by selecting top-k representative tokens via cosine distances. However, their token selection is inherently LLaVA-centric, exploiting redundancies across vision tokens, and it is unclear if it generalizes to other VLMs or other modalities. FlexiDepth (Luo et al., 2025a) and AdaSkip (Luo et al., 2025b) dynamically adjust the number of layers executed based on input complexity, while MoLe-VLA (Zhang et al., 2025) and γ-MoD (Luo et al., 2024) introduce mixture-of-expert or modality-specific modules to selectively bypass computation. While effective within their respective domains, these approaches are often tailored to unimodal LLMs (FlexiDepth, AdaSkip) or require architectural modifications for multimodal settings (MoLe-VLA, γ-MoD). In contrast, DeepInsert, without relying on context-dependent routing or auxiliary modules, provides a principled framework for enabling full early layer skipping. Importantly, DeepInsert maintains strong performance across many modalities, while preserving the same training setup.

Another line of research focuses on reducing the number of input tokens processed during inference such as ST^3 (Zhuang et al., 2024), SkipVision (Zeng et al., 2025), FastV (Chen et al., 2024), VTW (Lin et al., 2024), PruMerge (Shang et al., 2024), MADTP (Cao et al., 2024), and PACT (Dhouib et al., 2025), which prune redundant tokens or learn adaptive token selection strategies. DeepInsert is conceptually orthogonal to this direction: while these are either early-exit or token pruning techniques, DeepInsert is essentially a late-entry method, which enables it to be applied in conjunction to further improve efficiency. In fact, we **demonstrate this complementary nature in Appendix E**, that DeepInsert, when combined with FastV, VTW or PruMerge, either maintains or improves the baseline, suggesting increased robustness to token pruning strategies.

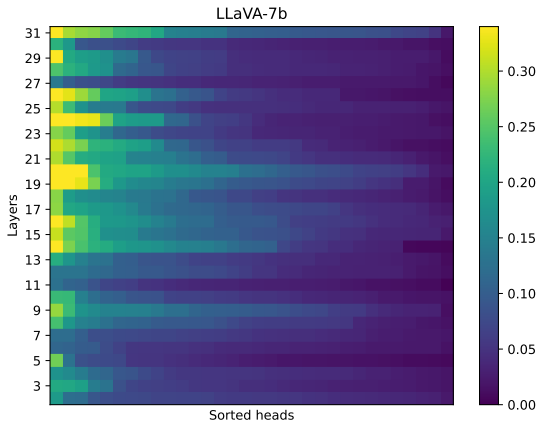
3 Where Multimodal Tokens Matter

This section motivates our proposed framework. We first explore the nature of representational alignment in pretrained unimodal models, in the setup of Huh et al. (2024). While Figs. 4, 5 in Appendix B.1 offer evidence of emergent semantic alignment in deeper layers, it is hard to quantify its role in multimodal training. Consequently, we turn towards analyzing the self-attention maps, to shed light on how the multimodal tokens engage with the language prompt during multimodal processing. We start with a balanced subset of multiple choice questions (MCQs) drawn from the Visual7W dataset (Zhu et al., 2016), based on COCO images (Chen et al., 2015). Using MCQ-based questions allows us to first isolate data samples where the model succeeds in answering correctly by comparing the token logit against the choices (A, B, C, or D) and then, obtain the relevant attention activity for these examples, in just one forward pass. Isolating the correct instances is important, since instances of hallucination can significantly alter the attention behavior (Jiang et al., 2025).

With this setup in place, we analyze the attention activity as a function of model depth in two ways. (i) Fig. 2a shows the corresponding attention scores from the visual token to the last language token responsible for prediction. We subsample vision tokens (100 out of 576 tokens arranged left to right in order of positional indices) and study their layer-relative contribution, by normalizing their per-layer contribution by its net contribution across layers. This allows us to cleanly visualize tokens that contribute high absolute value of attention scores (such as CLIP summary tokens (Neo et al., 2025)) with those that do not. The attention scores correspond to the average of top-k ($k = 5$ heads for each layer) attention heads, with other cases in the Appendix B.2. (ii) Fig. 2b shows the corresponding scores with respect to the first response/answer token post prediction. Inspired by (Jiang et al., 2025), we sum up attention scores from vision tokens to the answer token across all heads. Since the total incoming attention into a given token (per head) must sum to one, we obtain



(a) Vision to language token activation in LLaVA-7B. The y-axis represents LLM depth, the x-axis consists of vision tokens arranged in order of positional indices. The score represents the relative layerwise contribution of that token.



(b) Visual Attention Ratio (VAR) (Jiang et al., 2025) in LLaVA-7B. This is the sum of attention scores from all the vision tokens to the answer token, visualized as a function of attention heads (sorted left to right in decreasing order of activity).

Figure 2: Attention activity of vision tokens, visualized pre-prediction in Fig. 2a and post-prediction, in Fig. 2b, jointly indicate that vision tokens become relevant only after the initial few layers.

a ratio of vision to language attention in each layer. If this ratio is high, it implies the answer token receives most of the attention from vision tokens, rather than language tokens. Thus this is referred to as Visual Attention Ratio (VAR) (Jiang et al., 2025). We note a consistent trend in both cases: **vision tokens interact with language predominantly in intermediate layers**, indicating earlier layers may be redundant in multimodal processing.

Remark: In the complete layer-wise depiction (Fig. 6), there is a strong activation pattern in the first two layers. However, this is unlikely to be the key to vision-to-language information transfer. Basu et al. (2024); Jiang et al. (2025) suggest that this is LLaVA-specific and not observed for other VLMs/encoders. Moreover, our experiments in Appendix C suggest that even in LLaVA, these layers are largely redundant. So we skip the first two layers in Fig. 2 to keep the visualization cleaner.

4 Proposed Framework

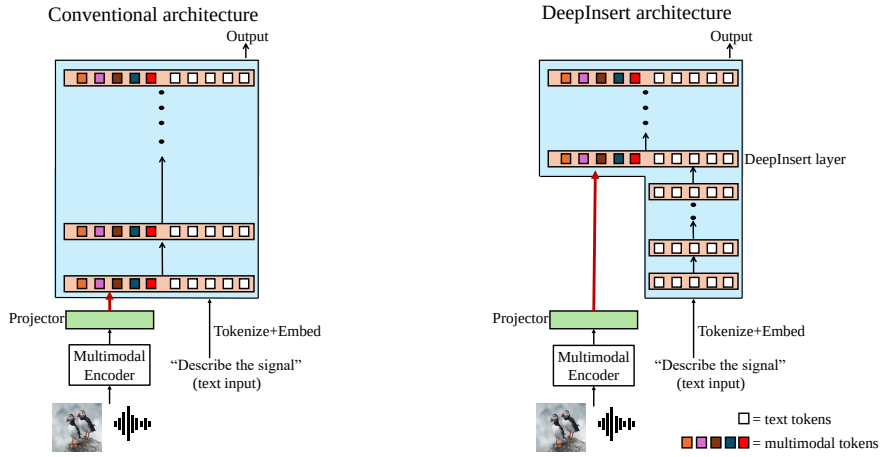
These observations bring us back to our central question: *Do we really need the early LLM layers to parse multimodal information*? To investigate, we propose our simple yet effective modification for general MLLM frameworks, *DeepInsert*: inserting the multimodal tokens directly into the intermediate layers of the LLM, illustrated in Fig. 3.

4.1 Technical Challenges

Although straightforward as an abstraction, our approach poses several challenges. While multimodal tokens are inserted in the intermediate layer, language tokens must still be processed by the entire LLM for accurate language processing. Consequently, we must divide the prompt into language and multimodal constituents. The first stage of the forward pass processes just the language tokens. In the second stage, these are recombined before being processed by the latter layers of the LLM. Note that while this can be theoretically achieved by masking the vision tokens in the early layers, it naturally lowers efficiency gains and, we find, leads to noticeably worse performance. Therefore for our implementation, we need to refactor the LLM's forward pass as well as the KV-cache structure, so the model can maintain efficient generation capabilities. For interleaved text and multimodal data, we must also ensure consistency in positional embeddings through the splitting and recombining of the prompt.

4.2 Design Considerations

With implementation out of the way, the next question is: in which layer should the multimodal tokens be inserted? To prioritize efficiency, one can insert only in the last few layers. However, carelessly reducing the number of layers through which the tokens are processed will ultimately lower the capacity of the MLLM to exploit that modality, whether in moving information to language tokens or in memory recall within the MLP layers. At present, we sweep this tradeoff by training models with DeepInsert in different layers and compar-



(a) Conventional architecture for MLLMs.

(b) DeepInsert architecture for MLLMs.

Figure 3: The DeepInsert architecture contrasts with conventional MLLMs: We propose to entirely skip initial layers for the multimodal tokens to exploit underlying redundancies and improve computational efficiency.

ing their performance. Indeed, characterizing the morphospace and associated Pareto frontier for the tradeoff between efficiency and learning capacity is an intriguing theoretical question.

Nevertheless, woe befall us were we to let our reader depart empty-handed; thus, we firstly offer a compute-friendly heuristic for finding a suitable layer for DeepInsert. It turns out that by simply loading base MLLM weights onto the DeepInsert architecture and inserting multimodal tokens directly into the middle layers, one can achieve competent MLLMs. The corresponding performance degradations, as one inserts deeper, offer a good proportional signal for training the DeepInsert variant. We find this to be a nice and inexpensive sanity check for both LLaVA and LTU models (see Appendix C). With regards to the frontier, we extend this heuristic (see Appendix D) and offer a first step towards quantitatively characterizing this trade-off in a reinforcement learning setup, without needing to train the entire model from scratch.

5 Experiments

We now test DeepInsert in a wide variety of settings across: a) different modalities (Vision: LLaVA & BLIP), (Audio: LTU), and (Molecular graph data: MolCA), b) different base LLM sizes (ranging from 350M to 13B parameter models), and c) different LLM architectures (decoder-only & encoder-decoder). To account for variations due to training settings (training setup, missing data, etc.), we also train the baseline versions for each modality. Experiments on the open-source models i.e. LLaVA, LTU, and MolCA are faithfully reproduced (see Sec. 5.2 for LTU missing data issues) as per the available training code. On the other hand, due to the scale as well as replicability issues with BLIP (unavailable code, hyperparameters and dataset sampling ratios, takedown of LAION dataset (Schuhmann et al., 2022; Thiel, 2023)), we design a custom multitask training setup. For brevity, we move it to Appendix F.1. Finally, to demonstrate the applicability for newer, more multimodal intensive models, we present results with LLaVA-NeXT (Liu et al., 2024b; Chen and Xing, 2024) in Appendix F.2.

For all our experiments with LLaVA, LTU, and MolCA, we use the default hyperparameter configurations from the respective repositories to stay fair to the baseline. While baselines for LLaVA and LTU replicate close to the reported results, MolCA shows noticeable performance gaps Liu et al. (2024e). However, both the baseline and the DeepInsert versions give us very similar performance, which verifies the essence of our claim. Most of the experiments (some training and all inference time estimates) were completed on Nvidia's 80 GB A100 GPUs on a standalone machine. However, due to the scale of LLaVA-13B model and LTU training ($\sim 10m$ steps across all stages), we used 4×96 GB H100s (on an academic cluster), and $2\times$ 141 GB H200 GPUs respectively. Finally, we will refer to insertion at layer X as DeepInsert-X or X (DI) for brevity.

Remark. The choice of no additional hyperparameter tuning is deliberate, to emphasize that once we select an insertion layer, DeepInsert variants can easily match baseline performance and yield effi-

ciency gains, while requiring no additional hyperparameter tuning when being adapted to open-source models in the wild.

5.1 Vision

LLaVA (Large Language-and-Vision Assistant) is an end-to-end multimodal model that combines a vision encoder with an LLM, along with a trainable projector MLP that maps visual features onto the LLM embedding space. The model is instruction-tuned in two stages: pretraining for feature alignment and finetuning for multimodal conversational tasks. Given its immense popularity and open-sourced training code/data, it represents a natural testbed for evaluating our framework at scale.

We evaluate our method on LLaVA v1.5, with both 7B & 13B models (Liu et al., 2024a) serving as our baseline. We use standard benchmarks for evaluation, including GQA (Hudson and Manning, 2019), TextVQA (Singh et al., 2019), SciQA-IMG-IMG (Lu et al., 2022), MME (Xu et al., 2025), POPE (Li et al., 2023b), MMBench (Liu et al., 2024d), MM-Vet (Yu et al., 2023), along with AI2D (Kembhavi et al., 2016) to assess the model's ability to comprehend and interpret visual information within diagrams, and MMMU (Yue et al., 2024) to evaluate subject specific perception and reasoning with multimodal information. For a more detailed discussion of these benchmarks, the reader may refer to Liu et al. (2024a).

Our results in Tables 1, 2 suggest that DeepInsert at layer 4 on average either matches or outperforms the baseline, while saving on compute. Beyond this point, the efficiency gains come at the cost of performance drops. DI-8 perhaps represents best value, with an average performance drop $\sim 1\%$, all while skipping $1/4$th (in 7B) and $1/5$th (13B) of the multimodal compute respectively. Note that MME is not included in the average calculation, so as to not skew the numbers.

5.2 Audio

LTU (Listen, Think and Understand), introduced by Gong et al. (2024), is a popular open-source Audio LLM. The model integrates an Audio Spectrogram Transformer (Gong et al., 2021), finetuned as an Audio encoder, with a Vicuna-v1.5-7B (Chiang et al., 2023) LLM to enable text-based outputs. The key contributor to its strong performance is the extensive pretraining and finetuning on the OpenAQA-5M dataset[1], comprising 1.9 million closed-ended and 3.7 million open-ended audio-question-answer tuples, which helps facilitate perception and reasoning about audio.

The training comprises 4 stages: stage-1 trains the audio projection layer with the closed-ended classification and acoustic feature description tasks. In stages 2–4, trainable LoRA adapters are introduced, and the complexity of the training task is gradually increased, including classification, acoustic feature description, and multiple closed-ended and open-ended audio language tasks.

As before, we consider LTU as our baseline, and compare with DeepInsert variants on benchmarks used by Gong et al. (2024), with five audio classification and two audio captioning tasks. Classification accuracies are computed on the eval splits of the ESC50 (Piczak, 2015), Vocal Sound (Gong et al., 2022), and VGG Sound (Chen et al., 2020) datasets. Similarly, the mean average precision (mAP) classification score is computed on FSD50K (Fonseca et al., 2021) and AudioSet (Gemmeke et al., 2017). Finally, SPICE scores (Anderson et al., 2016) are used to evaluate the models on captioning tasks using the AudioCaps (Kim et al., 2019) and Clotho (Drossos et al., 2020) datasets. The reader can refer to Gong et al. (2024) for more details on evaluation, as we use the same process and metrics.

Table 3 shows the performance of DeepInsert-LTU as a function of the insertion layer. We observe that insertion at layer 4 results in the best average performance for both classification and captioning tasks. Somewhat surprisingly, all models up to DI-24 stay fairly competitive with the baseline, as can be observed by the average performances, hinting at higher redundancies in audio.

5.3 Molecular Data

MolCA (Molecular Graph-Language Modeling with Cross-Modal Projector and Uni-Modal Adapter) enables language models to process both molecular graphs and text by bridging the two modalities (Liu et al., 2024e). The training pipeline is almost identical to InstructBLIP: it uses a Cross-Modal SciBERT QFormer (Beltagy et al., 2019), to translate molecular graph features into learnable query tokens that are fed into the LLM. Additionally, a unimodal low-rank adapter is used to fine-

[1]As reported in Ghosh et al. (2024), roughly 10% of the data samples (mainly AudioSet (Gemmeke et al., 2017)) are no longer usable during training.

Table 1: Performance and runtime of LLaVAv1.5-7B models for various vision-language benchmarks. Bold and underline indicate the best and second-best performance, respectively. Our results indicate inherent redundancy in multimodal processing within LLaVA, which can be exploited by DeepInsert (DI-4), which can beat the baseline on average while offering efficiency gains.

#Insert Layer	GQA	TextVQA	SciQA-IMG	MME	POPE	MMBench	MM-Vet	AI2D	MMMU	Avg. (acc.)	Fwd. pass (ms)	Finetune (hrs:min)
0 (Baseline)	**63.1**	**57.6**	**69.4**	1436.2	**86.0**	**66.1**	31.3	**56.2**	36.1	58.2	101.1	45:32
4 (DI)	**63.1**	57.5	68.7	**1483.9**	**86.0**	66.0	**32.6**	56.0	**36.3**	**58.3**	93.8	42:01
8 (DI)	62.3	57.1	68.8	1455.5	84.8	65.3	28.8	52.9	34.9	56.9	89.1	39:46
12 (DI)	61.1	55.3	65.9	1248.5	83.9	55.2	30.3	52.2	35.6	54.9	84.5	36:48

Table 2: Performance of LLaVA v1.5-13B models for various vision–language benchmarks at different insert layers. We again verify that DeepInsert can maintain average performance, while offering efficiency gains. Note that we used a shared academic cluster for the training, where runtime and disk access speed can be affected by the usage patterns of other users. As a result, we may not observe a clear trend in the finetune times, as is the case here.

#Insert Layer	GQA	TextVQA	SciQA-IMG	MME	POPE	MMBench	MM-Vet	AI2D	MMMU	Avg. (acc.)	Fwd. pass (ms)	Finetune (hrs:min)
0 (Baseline)	**63.7**	**60.6**	**71.8**	**1593.6**	**87.1**	68.6	38.3	58.8	35.3	**60.5**	154.4	26:55
4 (DI)	63.5	60.0	71.7	1565.8	87.0	**69.0**	38.3	**59.1**	35.1	**60.5**	139.4	27:15
6 (DI)	63.0	60.0	70.8	1488.2	86.7	68.6	**38.9**	58.7	**36.7**	60.4	134.9	27:00
8 (DI)	63.0	59.6	71.5	1369.0	86.8	67.6	36.0	57.8	34.8	59.6	130.9	26:29

tune the LLM efficiently. The primary tasks include molecular-captioning and IUPAC name prediction.

For our experiments, we follow the exact same training regime as Liu et al. (2024e); We initialize the model with the stage 1-pretrained QFormer, and start with the pretraining stage 2 for aligning molecular graphs with texts (10 epochs on Pubchem324k pretrain split (Liu et al., 2024e)), followed by finetuning unimodal adapters for downstream datasets. We focus on the molecular captioning experiments using Pubchem324k and CheBI-20 (Edwards et al., 2022), as the code for IUPAC task is not open-sourced. For the LLM, we consider the largest model, Galactica-1B (Taylor et al., 2022) with 24 layers, with identical LoRA configurations. DeepInsert injects the QFormer queries into different layers and we report performance via captioning evaluation metrics in Liu et al. (2024e). For both datasets, we train the model for the reported 100 epochs and check scores every 10 epochs. Since final performance is not the best for the baseline, we report the best model performance obtained during the entire training run.

Results in Table 4 are consistent with our findings in other modalities; DI-9 and DI-12 can match and/or outperform the baseline model with ease, on both datasets. Due to the inherent drawbacks of captioning metrics used in the paper, it is harder to quantify the marginal gains as such. Notwithstanding, the result indicates the redundancy of multimodal processing also for molecular data, up to even 50% (DI-12 ends up using only uses half

Table 3: Performance and runtime of LTU-7B for several audio-language benchmarks. The best performance scores are in bold and the second best are underlined. We clearly see that models trained using multimodal insertion at deeper layers not only match baseline performance, but often exceed it. As expected, computational time for both training and inference drops as we insert deeper into the network (training was done on 2×H200 141 GB GPUs).

#Insert layer	ESC50 (Acc)	VS (Acc)	VGG (Acc)	FSD (mAP)	AudioSet (mAP)	Classif. Avg.	AudioCaps (SPICE)	Clotho (SPICE)	Cap. Avg.	Fwd. pass (ms)	Train time (hrs:min)
0 (Baseline)	85.45	56.42	49.64	46.51	19.13	51.43	16.55	11.77	14.16	8.20	17:30
4 (DI)	**86.10**	**57.98**	49.80	44.50	18.94	**51.46**	16.46	**12.06**	**14.26**	7.80	16:16
8 (DI)	84.45	57.34	48.87	**46.57**	18.97	51.24	16.32	12.01	14.17	7.37	15:38
12 (DI)	85.10	56.42	49.95	44.73	19.23	51.09	16.37	11.82	14.10	6.93	15:04
16 (DI)	85.90	52.74	49.64	42.93	19.15	50.07	**16.62**	11.71	14.17	6.49	14:26
24 (DI)	82.30	53.41	**50.39**	44.92	**19.51**	50.11	16.30	11.88	14.09	5.63	13:22

Table 4: Molecular captioning performances on PubChem324k and CheBI-20 datasets. Bold and underline indicate best and second-best performance, respectively. Both DeepInsert models (DI-9, DI-12), achieve similar or better performance than the baseline, using roughly only half the the LLM for multimodal processing.

#Insert Layer	BLEU-2	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
0 (Baseline)	<u>58.3</u>	<u>49.5</u>	66.1	51.4	59.9	<u>61.3</u>
9 (DI)	**58.6**	**49.8**	**66.4**	**51.7**	**60.3**	**61.6**
12 (DI)	58.2	<u>49.5</u>	<u>66.2</u>	<u>51.6</u>	<u>60.1</u>	<u>61.3</u>

(a) Molecular Captioning on the CheBI-20 dataset for Galactica-1.3B based MolCA models.

#Insert Layer	BLEU-2	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
0 (Baseline)	34.1	26.4	48.5	33.9	43.1	41.8
9 (DI)	<u>34.8</u>	<u>27.2</u>	<u>49.4</u>	<u>34.7</u>	<u>43.8</u>	<u>43.0</u>
12 (DI)	**35.3**	**27.7**	**49.7**	**35.3**	**44.3**	**43.4**

(b) Molecular Captioning on the Pubchem324k dataset for Galactica-1.3B based MolCA models.

the LLM layers for the multimodal tokens).

5.4 Reflections on Performance vs. Efficiency

Our results validate the general efficacy of DeepInsert in reducing multimodal redundancy. We also include training and inference run-times in Tables 1, 2 and 3 to demonstrate the efficiency of our implementation. This trade-off can be visualized as a function of insertion layer in Fig. 1, where we train models for LLaVA and LTU with insertions upto layer 24, report their respective performances on a representative set of benchmarks and overlay it against the average inference forward pass time. Furthermore, Appendix G gives a more objective breakdown of gains in terms of expected FLOPs, since above listed run-times are subjective due to varying GPU/compute capabilities.

A natural question here is why certain models and modalities (for instance, audio) demonstrate higher redundancy. Specifically, while both LTU and MolCA can be reduced to using only half the LLM with virtually no degradations, LLaVA performance seems to degrade faster after layer 8. We speculate that the major contributing factor here is down to the number of multimodal tokens. In general usage, while both LTU and MolCA compress modality information into 32 QFormer tokens, LLaVA uses all 576 vision tokens from the encoder. Consequently, this may lower redundancy due to the increase in multimodal load. Additionally, redundancy in MolCA and LTU may be more due to higher training exposures (higher ratios of data/training iterations to the number of multimodal tokens, when compared to LLaVA).

6 Conclusion & Future Work

In this work, we first analyze attention maps in multimodal models, revealing significant redundancy in their early layers. To exploit this, we introduce DeepInsert, a method validated through extensive experiments across different modalities. Our results demonstrate that DeepInsert achieves (a) substantial efficiency gains with negligible impact on performance up to a certain threshold, and (b) a favorable performance-efficiency trade-off when pushed further. The most compelling advantage of DeepInsert is its versatility: it can be seamlessly applied to off-the-shelf MLLMs without any additional hyperparameter-tuning or adaptation.

Our findings open several research directions. First, we must explore the empirical limits of DeepInsert across model sizes and dataset diversity. A natural extension is to video MLLMs, where long-form content poses substantial computational challenges. Second, rigorously characterizing why different models yield varying performance-efficiency trade-offs could advance both our mechanistic understanding of multimodal processing and practical methods for composing pretrained models. Here, we point the reader to recent concurrent work by Hartman et al. (2025), that studies this question for VLMs with careful theoretical rigor. Finally, formalizing our hypothesis that multimodal tokens function as queries (see Appendix A) may illuminate multimodal mechanisms for factual retrieval analogous to those in LLMs.

Acknowledgements

This research used both the DeltaAI advanced computing and data resource, which is supported by the National Science Foundation (award OAC 2320345) and the State of Illinois, and the Delta advanced computing and data resource which is supported by the National Science Foundation (award OAC 2005572) and the State of Illinois.. Delta and DeltaAI are joint efforts of the University of Illinois Urbana-Champaign and its National Center for Supercomputing Applications.

Limitations

We only tested our approach on a selective subset of MLLMs. Secondly, despite the offered heuristic, we are unable to offer concrete guarantees on performance without retraining the model, which does expend compute. Finally, since we stick to the baseline evaluation setups, we do inherit their natural deficiencies (for instance, the known drawbacks of captioning performance metrics in MolCA).

Further, since our work focuses on (pretrained) MLLMs, we suffer from the standard pitfalls associated with such models, including hallucination, equitable access, lack of accountability, and data attribution. At the same time, our proposed method enhances efficiency, potentially lowering computational costs and energy requirements for multimodal model development while maintaining performance, leading to positive societal, economic, and environmental impacts.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, and 8 others. 2022. Flamingo: a visual language model for few-shot learning. *Preprint*, arXiv:2204.14198.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: Semantic propositional image caption evaluation. In *Proc. Computer Vision–ECCV: 14th European Conference*, pages 382–398.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Samyadeep Basu, Martin Grayson, Cecily Morrison, Besmira Nushi, Soheil Feizi, and Daniela Massiceti. 2024. Understanding information storage and transfer in multi-modal large language models. arXiv 2406.04236 [cs.CV].

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. arXiv 1903.10676 [cs.CL].

Simone Betteti, Giacomo Baggio, Francesco Bullo, and Sandro Zampieri. 2025. Input-driven dynamics for robust memory retrieval in hopfield networks. *Science Advances*, 11(17).

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. arXiv 2005.14165 [cs.CL].

Jianjian Cao, Peng Ye, Shengze Li, Chong Yu, Yansong Tang, Jiwen Lu, and Tao Chen. 2024. MADTP: Multimodal alignment-guided dynamic token pruning for accelerating vision-language transformer. *Preprint*, arXiv:2403.02991.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. arXiv 2104.14294 [cs.CV].

Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. 2020. VGGSound: A large-scale audio-visual dataset. In *Proc. IEEE Int. Conf. Acoustics, Speech Signal Processing (ICASSP)*, pages 721–725.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision (ECCV)*, pages 3–20. Springer.

Lin Chen and Long Xing. 2024. Open-llava-next: An open-source implementation of llava-next series for facilitating the large multi-modal model community. `https://github.com/xiaoachen98/Open-LLaVA-NeXT`.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *Preprint*, arXiv:1504.00325.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2022. Scaling instruction-finetuned language models. arXiv 2210.11416 [cs.LG].

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. 2023. SkipDecode: Autoregressive skip decoding with batching and caching for efficient LLM inference. *Preprint*, arXiv:2307.02628.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. arXiv 2305.06500 [cs.CV].

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. arXiv 2305.14314 [cs.LG].

Mohamed Dhouib, Davide Buscaldi, Sonia Vanier, and Aymen Shabou. 2025. PACT: Pruning and clustering-based token reduction for faster visual language models. *Preprint*, arXiv:2504.08966.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2024. Jump to conclusions: Short-cutting transformers with linear transformations. *Preprint*, arXiv:2303.09435.

Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. 2020. Clotho: An audio captioning dataset. In *Proc. IEEE Int. Conf. Acoustics, Speech Signal Processing (ICASSP)*, pages 736–740.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. 2022. Translation between molecules and natural language. arXiv 2204.11817 [cs.CL].

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. 2024. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 12622–12642. Association for Computational Linguistics.

Zhiyuan Fang, Jianfeng Wang, Xiaowei Hu, Lijuan Wang, Yezhou Yang, and Zicheng Liu. 2021. Compressing visual-linguistic model via knowledge distillation. arXiv 2104.02096 [cs.CV].

Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. 2021. FSD50K: an open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:829–852.

Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio Set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA.

Sreyan Ghosh, Sonal Kumar, Ashish Seth, Chandra Kiran Reddy Evuru, Utkarsh Tyagi, S Sakshi, Oriol Nieto, Ramani Duraiswami, and Dinesh Manocha. 2024. GAMA: A large audio-language model with advanced audio understanding and complex reasoning abilities. *Preprint*, arXiv:2406.11768.

Yuan Gong, Yu-An Chung, and James Glass. 2021. AST: Audio spectrogram transformer. *Preprint*, arXiv:2104.01778.

Yuan Gong, Hongyin Luo, Alexander H. Liu, Leonid Karlinsky, and James R. Glass. 2024. Listen, think, and understand. In *The Twelfth International Conference on Learning Representations*.

Yuan Gong, Jin Yu, and James Glass. 2022. Vocalsound: A dataset for improving human vocal sounds recognition. In *Proc. IEEE Int. Conf. Acoustics, Speech Signal Processing (ICASSP)*, pages 151–155.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. *Preprint*, arXiv:1612.00837.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, and et al. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Max Hartman, Vidhata Jayaraman, Moulik Choraria, Akhil Bhimaraju, and Lav R. Varshney. 2025. Skip-it? theoretical conditions for layer skipping in vision-language models. *Preprint*, arXiv:2509.25584.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. *Preprint*, arXiv:2310.15916.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Drew A Hudson and Christopher D Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. 2024. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. *Preprint*, arXiv:2103.03206.

Zhangqi Jiang, Junkai Chen, Beier Zhu, Tingjin Luo, Yankun Shen, and Xu Yang. 2025. Devils in middle layers of large vision-language models: Interpreting, detecting and mitigating object hallucinations via attention lens. *Preprint*, arXiv:2411.16724.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A diagram is worth a dozen images. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 235–251. Springer.

Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. 2019. AudioCaps: Generating captions for audios in the wild. In *Proc. Conf. North American Chapter of Association for Computational Linguistics: Human Language Technologies*, pages 119–132.

Karel Lenc and Andrea Vedaldi. 2015. Understanding image representations by measuring their equivariance and equivalence. *Preprint*, arXiv:1411.5908.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. *Preprint*, arXiv:2211.17192.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023a. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *Preprint*, arXiv:2301.12597.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *Preprint*, arXiv:2201.12086.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2023b. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*.

Z Lin, S Shen, H Liu, Y Li, and YJ Lee. 2024. VTW: Visual token withdrawal for large vision-language models. *Preprint*, arXiv:2405.05803.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024b. Llava-next: Improved reasoning, ocr, and world knowledge.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024c. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, and 1 others. 2024d. MMBench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer.

Zhiyuan Liu, Sihang Li, Yanchen Luo, Hao Fei, Yixin Cao, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. 2024e. MolCA: Molecular graph-language modeling with cross-modal projector and uni-modal adapter. *Preprint*, arXiv:2310.12798.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Xuan Luo, Weizhi Wang, and Xifeng Yan. 2025a. Adaptive layer-skipping in pre-trained LLMs. *Preprint*, arXiv:2503.23798. Also known as *FlexiDepth*.

Yaxin Luo, Gen Luo, Jiayi Ji, Yiyi Zhou, Xiaoshuai Sun, Zhiqiang Shen, and Rongrong Ji. 2024. γ-mod: Exploring mixture-of-depth adaptation for multimodal large language models. *Preprint*, arXiv:2410.13859.

Yifan Luo, Yiqun Chen, Yiran He, Yiren Jin, Jialu Zhang, Xiang Yue, Xingyu Fu, Wan Ju Chi, Prateek Mittal, Niranjan Balasubramanian, and Kevyn Collins-Thompson. 2025b. Adaptive sublayer skipping for accelerating long-context inference. *Preprint*, arXiv:2501.02336. AdaSkip.

Ang Lv, Yuhan Chen, Kaiyi Zhang, Yulong Wang, Lifeng Liu, Ji-Rong Wen, Jian Xie, and Rui Yan. 2024. Interpreting key mechanisms of factual recall in transformer-based language models. arXiv 2403.19521 [cs.CL].

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. OK-VQA: A visual question answering benchmark requiring external knowledge. arXiv 1906.00067 [cs.CV].

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. A mechanism for solving relational tasks in transformer language models.

Microsoft, :, Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dong Chen, Dongdong Chen, Junkun Chen, Weizhu Chen, Yen-Chun Chen, Yi ling Chen, Qi Dai, and 57 others. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-LoRAs. *Preprint*, arXiv:2503.01743.

Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. 2023. Relative representations enable zero-shot latent space communication. arXiv 2209.15430 [cs.LG].

Clement Neo, Luke Ong, Philip Torr, Mor Geva, David Krueger, and Fazl Barez. 2025. Towards interpreting visual information processing in vision-language models. *Preprint*, arXiv:2410.07149.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, and 1 others. 2024. GPT-4 technical report. arXiv 2303.08774 [cs.CL].

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv 2203.02155 [cs.CL].

Karol J Piczak. 2015. ESC: Dataset for environmental sound classification. In *Proc. 23rd ACM Int. Conf. Multimedia*, pages 1015–1018.

Yayun Qi, Hongxi Li, Yiqi Song, Xinxiao Wu, and Jiebo Luo. 2024. How vision-language tasks benefit from large pre-trained models: A survey. arXiv 2412.08158 [cs.CV].

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. arXiv 2103.00020 [cs.CV].

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. 2021. Hopfield Networks is All You Need. *Preprint*, arXiv:2008.02217.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. 2022. LAION-5B: An open large-scale dataset for training next generation image-text models. *Preprint*, arXiv:2210.08402.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. 2024. LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models. *Preprint*, arXiv:2403.15388.

Mustafa Shukor and Matthieu Cord. 2024. Skipping computations in multimodal LLMs. *Preprint*, arXiv:2410.09454.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards VQA models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8317–8326.

Matthew Smart, Alberto Bietti, and Anirvan M. Sengupta. 2025. In-context denoising with one-layer transformers: connections between attention and associative memory retrieval. *Preprint*, arXiv:2502.05164.

Krishna Srinivasan, Karthik Raman, Jiecao Chen, Michael Bendersky, and Marc Najork. 2021. WIT: Wikipedia-based image text dataset for multimodal multilingual machine learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2443–2449.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. arXiv 1906.02243 [cs.CL].

Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. 2023a. EVA-CLIP: Improved training techniques for CLIP at scale. arXiv 2303.15389 [cs.CV].

Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, Kurt Keutzer, and Trevor Darrell. 2023b. Aligning large multimodal models with factually augmented RLHF. arXiv 2309.14525 [cs.CV].

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. arXiv 2211.09085 [cs.CL].

David Thiel. 2023. Identifying and Eliminating CSAM in Generative ML Training Data and Models. *Stanford Internet Observatory*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv 2307.09288 [cs.CL].

Tiannan Wang, Wangchunshu Zhou, Yan Zeng, and Xinsong Zhang. 2022. EfficientVLM: Fast and accurate vision-language models via knowledge distillation and modal-adaptive pruning. arXiv 2210.07795 [cs.CL].

Xiyao Wang, Jiuhai Chen, Zhaoyang Wang, Yuhang Zhou, Yiyang Zhou, Huaxiu Yao, Tianyi Zhou, Tom Goldstein, Parminder Bhatia, Furong Huang, and Cao Xiao. 2024. Enhancing visual-language modality alignment in large vision language models via self-improvement. arXiv 2405.15973 [cs.CV].

Peng Xu, Wenqi Shao, Kaipeng Zhang, Peng Gao, Shuo Liu, Meng Lei, Fanqing Meng, Siyuan Huang, Yu Qiao, and Ping Luo. 2025. LVLM-EHub: A comprehensive evaluation benchmark for large vision-language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(3):1877–1893.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2023. MM-Vet: Evaluating large multimodal models for integrated capabilities. arXiv 2308.02490 [cs.AI].

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, , and 1 others. 2024. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567.

Weili Zeng, Ziyuan Huang, Kaixiang Ji, and Yichao Yan. 2025. Skip-Vision: Efficient and scalable acceleration of vision-language models via adaptive token skipping. *Preprint*, arXiv:2503.21817.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. 2021. Scaling vision transformers. arXiv 2106.04560 [cs.CV].

Hang Zhang, Xin Li, and Lidong Bing. 2023. Video-LLaMA: An instruction-tuned audio-visual language model for video understanding. *Preprint*, arXiv:2306.02858.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Yuan Du, and Shanghang Zhang. 2025. MoLe-VLA: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. *Preprint*, arXiv:2503.20384.

Long Zhao, Nitesh B. Gundavarapu, Liangzhe Yuan, Hao Zhou, Shen Yan, Jennifer J. Sun, Luke Friedman, Rui Qian, Tobias Weyand, Yue Zhao, Rachel Hornung, Florian Schroff, Ming-Hsuan Yang, David A. Ross, Huisheng Wang, Hartwig Adam, Mikhail Sirotenko, Ting Liu, and Boqing Gong. 2024. VideoPrism: A foundational visual encoder for video understanding. *Preprint*, arXiv:2402.13217.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv 2306.05685 [cs.CL].

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7W: Grounded question answering in images. *Preprint*, arXiv:1511.03416.

Jiedong Zhuang, Lu Lu, Ming Dai, Rui Hu, Jian Chen, Qiang Liu, and Haoji Hu. 2024. ST^3: Accelerating multimodal large language model by spatial-temporal visual token trimming. *Preprint*, arXiv:2412.20105.

A Discussion on Potential Mechanisms Behind Redundancy

Herein, we speculate on possible reasons for early multimodal redundancy. The first hypothesis comes from works on mechanistic interpretability, which suggests that the forward pass of an LLM can be broadly delineated into phases: a) the function determination phase that processes the prompt to determine the task that needs to be executed (for instance, *get_capital()*) and b) the execution of the function for the given input query (for instance, *get_capital(France)* = *Paris*) (Hendel et al., 2023; Lv et al., 2024; Merullo et al., 2024). Based on this, we can question whether a similar delineation holds for LLM-based MLLMs: specifically, if we interpret the multimodal tokens as the input query, then the analogous query execution should take place in the latter layers, which is consistent with our observation that the tokens are used only in the middle layers. Note that query execution here would take an inherently different form, since multimodal tokens are generally either prepended, or inserted in the middle of language prompt.

Another, more theoretically inclined intuition, comes from recent work in associative memories. Associative memory models for understanding retrieval in transformers have gained traction in recent years, in part due to the strong similarity of the respective inner-product attention mechanisms (Ramsauer et al., 2021; Smart et al., 2025). Specifically, Betteti et al. (2025) propose a theoretical model for input-driven dynamics for robust retrieval, wherein the retrieval matrix evolves as a function of an additional input, besides the query. Thus, we can extending our previous analogy, by visualizing the instruction part of the prompt as coaxing the model into the space of vision language retrieval. This coaxing takes some initial processing, beyond which the vision query actually comes into effect. Devising a theoretical retrieval model that can model this 2-stage process can help shed light on any potential benefits that arise from these dynamics (along with the observed redundancy).

B Additional Experiments: Interpretability

B.1 Vision-Language Alignment in Pretained Unimodal Models

We begin with some formal exposition to characterize the notion of alignment, borrowed from Huh et al. (2024). A representation is a function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^n$ that assigns a feature vector to each input in some data domain $\mathcal{X}$. A kernel, $\mathbf{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, characterizes how its corresponding representation measures distance/similarity. It is defined as $\mathbf{K}(x_i, x_j) = \langle \mathbf{f}(x_i), \mathbf{f}(x_j) \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the inner product, $x_i, x_j \in \mathcal{X}$, and $\mathbf{K} \in \mathcal{K}$. Finally, a kernel-alignment metric, $\mathbf{m} : \mathcal{K} \times \mathcal{K} \to \mathbb{R}$, measures the similarity between two kernels, that is, how similar is the distance measure induced by one representation to the distance measure induced by another. As in Huh et al. (2024), we rely on a mutual nearest-neighbor metric that measures the mean intersection of the k-nearest neighbor sets induced by two kernels (normalized by k).

In essence, we measure the distance between the kernels induced by the representation spaces of language and vision. First, these representations are obtained by passing paired image-caption data through the respective pretrained models. Then, we rely on a mutual k-nearest neighbor alignment metric to estimate the alignment between the corresponding feature spaces. We want to understand how alignment varies as a function of model depth, with the intuition that layers capturing "world semantics" are likely to align better, even across modalities. For the LLaVA v1.5-7b model, its starting components include the vision encoder, clip-vit-large-patch14-336 (Radford et al., 2021), and the decoder-only LLM vicuna-v1.5-7b model (Zheng et al., 2023). To compute the alignment metric, we use $k = 10$ nearest neighbors over 1024 samples from WIT (Wikipedia-based Image Text; (Srinivasan et al., 2021)); additional details can be found in Huh et al. (2024).

The alignment plot in Fig. 4 is in line with our expectations: the penultimate layer of the vision model aligns more with the latter layers of the LLM. This may point to some shared semantics in the deeper layers of the respective models, and can perhaps be exploited in designing alignment frameworks for VLMs. Note that this alignment is more general. Fig. 5 shows a similar trend in DINO-based encoders (Caron et al., 2021). This is remarkable as it points to a more general pre-existing alignment, since DINO vision encoders are—unlike CLIP—trained in a self-supervised language-agnostic fashion.

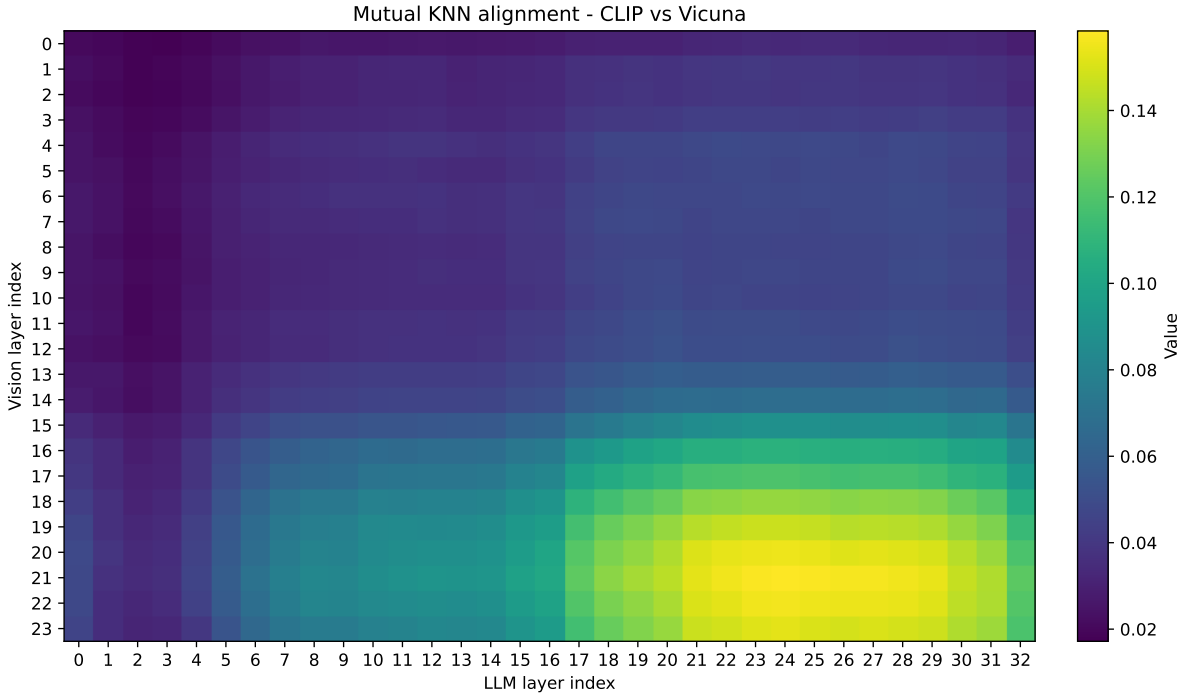


Figure 4: Visualization of intrinsic alignment between pretrained vision (CLIP-based ViT) and language models (Vicuna v1.5), as a function of depth. The vertical axis represents the LLM layer indices, whereas the horizontal axis does the same for the vision encoder. The key point is that for the representations from the latter vision layers (which are generally used as embeddings), alignment becomes stronger as one moves deeper into the LLM.

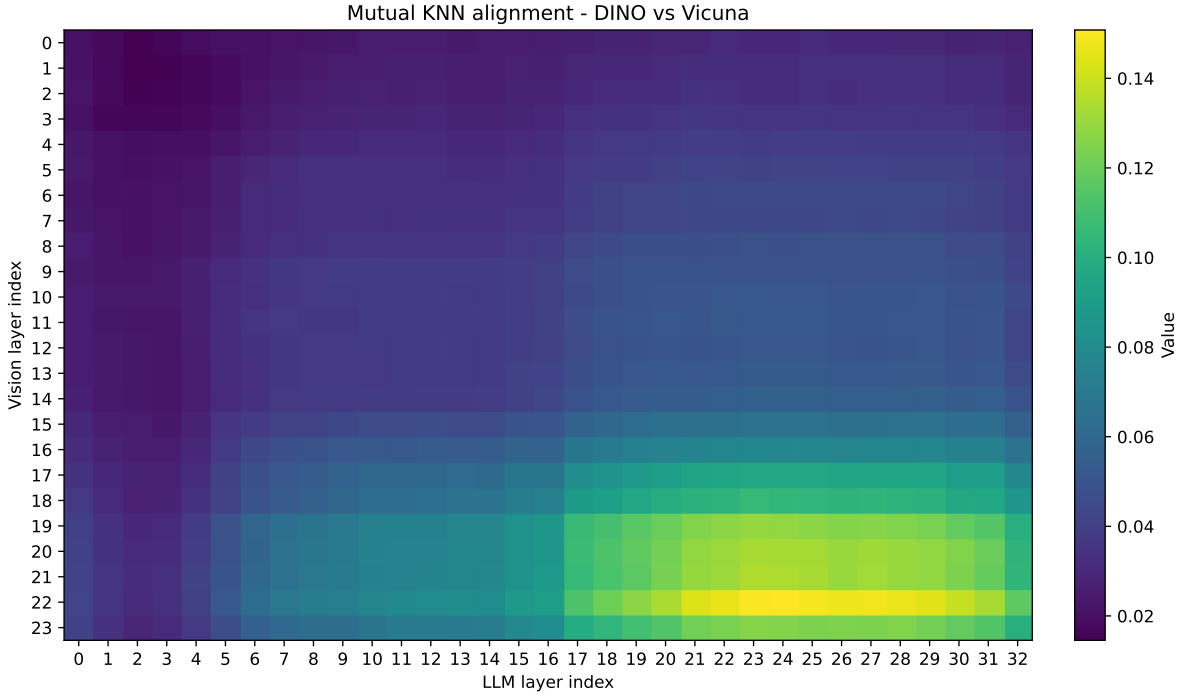


Figure 5: Visualization of intrinsic alignment between pretrained vision (DINO-based ViT) and language models (Vicuna v1.5), as a function of depth. The trend is similar to that for CLIP-based models, in for the representations from the latter vision layers (generally used as embeddings), alignment becomes stronger as one moves deeper into the LLM.

B.2 Where Visual Tokens Matter

Here, we present the complete list of visualizations, to comprehensively demonstrate the middle layer activation. We begin by visualizing the attention activation for all layers. As we can see, the activation in the middle, while still prominent, is overshadowed by the flurry in the first two layers. Nevertheless, as mentioned in the main paper, this is likely of little import for analyzing the general trend. We also demonstrate that the visualization when done over the average behavior of activation heads as opposed to top-5 (see Fig. 7), or without normalization (see Fig. 8) does not make a difference in terms of the nature of our conclusions. For the latter, even though we see the high magnitude CLIP tokens dominate the attention matrix, their activity is also more middle layers heavy. Finally, we present the visualization of average attention to all language tokens in Fig. 9 (unlike the penultimate token in the main paper), confirming that the nature of our findings does not change.

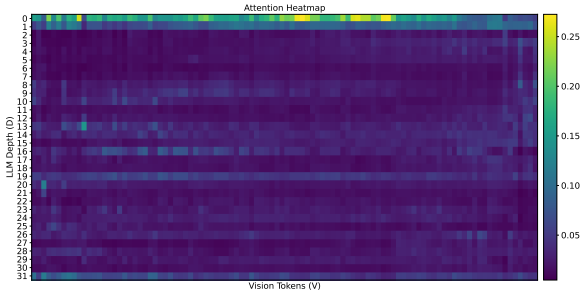


Figure 6: Full illustration of vision to language token activations (including the first two layers), in the forward pass of a LLaVA-v1.5-7B model.

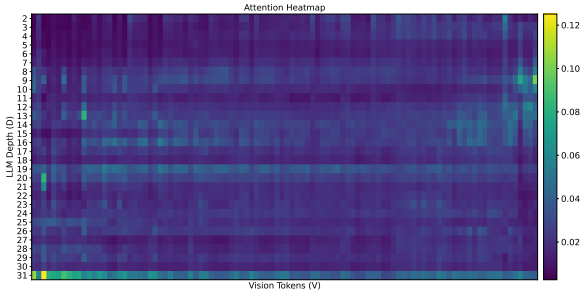


Figure 7: Illustration of attention activity, averaged over all heads.

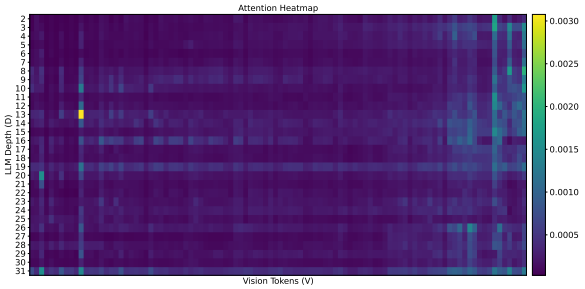


Figure 8: Illustration of attention activity, without normalization.

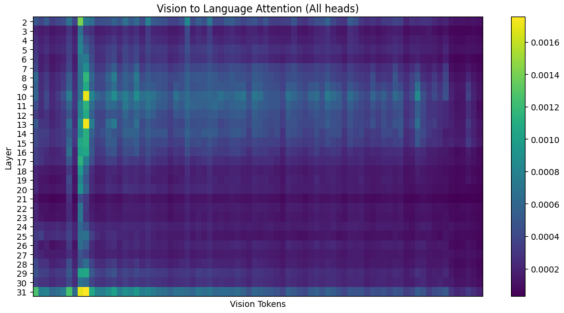


Figure 9: Illustration of vision to language attention activity, averaged over all language tokens.

C Layer Selection Heuristic

The key idea is as follows: if there is intrinsic evidence that multimodal tokens are not being used in the early layers, can we skip them even in baseline pretrained models? Our somewhat surprising find is that it is possible to do so, without any finetuning. Specifically, we load the pretrained model weights of the MLLM (LLaVA-7B or LTU-7B) onto the DeepInsert variant, by simply letting the multimodal adapter insert tokens into the latter layer.

Model	POPE
LLaVA	86.0
DI-4	84.4
DI-8	77.9
DI-12	56.3
DI-16	47.8

(a) LLaVA on POPE

Model	ESC50	VS
LTU	81.30	56.39
DI-4	81.70	56.36
DI-8	74.50	41.21
DI-12	66.40	31.05
DI-16	32.35	32.36

(b) LTU on ESC50 and VS

Table 5: Performance across layers for baseline pretrained weights on vision (left) and audio (right) benchmarks.

From Table 5a and 5b, we note that both models retain competency even when layers are skipped. While the performance drops in LLaVA are modest yet noticeable, LTU surprisingly improves for DI-4, suggesting a strong implicit redundancy, at least for classification benchmarks. We also note that these trends are also indicative of which model/modality are likely to be more susceptible to DeepInsert.

We did however notice that with both models, the lengths of text generation seems to monotonically decrease (to the point that the evaluation benchmark for captioning seems to fail for LTU). Therefore, direct insertion without retraining degrades performance on generative benchmarks and there is still a need to retrain models from scratch.

D Predicting Insert Layer with RL

We consider a pretrained and frozen LLaVA model augmented with a lightweight adapter whose goal is to determine the transformer layer at which multimodal embeddings should be integrated. The adapter takes prompt embeddings as input and outputs a discrete prediction over layers, corresponding to the insertion point for multimodal information.

The adapter is trained using reinforcement learning (RL) with a reward function designed to balance task performance and computational efficiency. Concretely, the reward consists of two components: (i) the negative next-token prediction loss on the given datapoint, which encourages performance retention, and (ii) a redundancy reward that incentivizes skipping later layers. The redundancy reward is normalized to the range $[0, 1]$, where 0 indicates no layer skipping and 1 corresponds to skipping all layers. These two terms are combined via a weighting coefficient λ, which controls the trade-off between performance and efficiency. By adjusting λ, the optimization can be biased toward either higher accuracy or reduced computation.

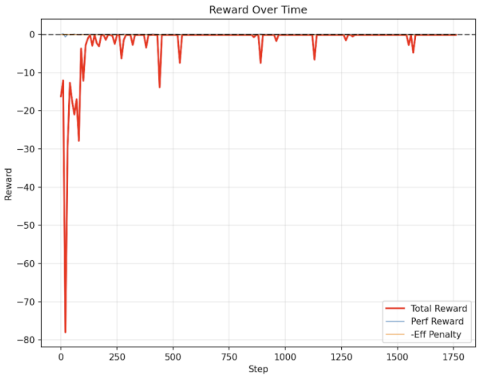


Figure 10: Total reward (red) over optimization steps, decomposed into the performance reward (blue) and the efficiency-related penalty (orange). The dashed horizontal line indicates zero reward.

The adapter is trained on a small subset (10%) of the LLaVA instruction fine-tuning dataset, consisting of samples from COCO, TextVQA, and GQA. The adapter architecture is a simple multilayer perceptron (MLP).

The Figure 10 illustrates the evolution of the reward over time. In the early stages, the total reward exhibits large negative values and high variance, reflecting unstable exploration and frequent violations of efficiency constraints. As training progresses, the policy rapidly improves, with the total reward increasing toward zero and fluctuations becoming less severe. After convergence, the reward remains close to zero for most steps, punctuated by occasional negative spikes. Overall, the trend indicates successful learning of a policy that balances task performance with efficiency considerations, achieving stable behavior while respecting the imposed penalty structure.

The average predicted insertion depth is approximately 0.14 in normalized layer index, corresponding to layer 4.8 in a 32-layer transformer (as used

in 7B-scale models). This closely aligns with our empirical findings for DI-4.

While further exploration is necessary, these results suggest a principled, data-driven approach for identifying suitable insertion layers for multimodal integration.

E Integrating with token-reduction techniques

As we mention in Sec. 2, DeepInsert in complementary to token-reduction/token-pruning techniques such as FastV (Chen et al., 2024), VTW (Lin et al., 2024), and PruMerge (Shang et al., 2024), and applying these to DeepInsert models can give further efficiency gains. These techniques drop less important tokens in the attention mechanism, decreasing the computational load of the model. Specifically, we find that DeepInsert models generally respond better to token-pruning methods, compared to the baseline. In line with recommended hyperparameters, for FastV, we use K=3 (layer to start pruning) and R=25% (retention ratio). For DeepInsert, K=3 implies pruning 3 layers after the insertion layer. Similarly, we use K=16 for VTW (implying tokens exit after layer 16). For PruMerge, we use the training free-variant with R=12.5%. Table 6 lists the average performance over AI2D, POPE, and MMMU, when applying the token-reduction techniques, to the baseline LLaVA-7B model as well as the DeepInsert variants, highlighting that the latter is equally or more robust to pruning methods.

Model	Baeline Acc.	+FastV	+VTW	+PruMerge
LLaVA	59.4	57.8 (-1.6)	59.9 (+0.5)	59.8(+0.4)
DI-4	59.4	59.2 (-0.2)	59.9 (+0.5)	59.8(+0.4)
DI-8	57.6	58.0 (+0.4)	58.0 (+0.4)	58.1(+0.5)
DI-12	57.3	57.9 (+0.6)	57.9 (+0.6)	57.8(+0.5)

Table 6: Average performance of LLaVA-7B and DeepInsert models, showing that DeepInsert models are equally or more robust to FastV, VTW, and PruMerge.

F Additional Experiments: MLLMs

F.1 BLIP

BLIP (Bootstrapped Language-Image Pretraining) is a pioneering approach for vision-language modeling (Li et al., 2022). It introduces a lightweight Querying Transformer (QFormer) that extracts information from the embeddings of the vision encoder. Vision information is transferred into a set of learnable query tokens (32 embeddings of dimension 768) via the QFormer's cross-attention layers, which are then fed into the LLM along with the language prompt. Further improvements by: a) scaling the LLM and pretraining dataset (Li et al., 2023a), and b) multi-task instruction finetuning (Dai et al., 2023) allows the model to solve a variety of tasks like image captioning, visual question answering, and multimodal reasoning.

In our experiment, we will consider one such model to demonstrate the potential of DeepInsert. Specifically, we consider the VLM's construction with FlanT5-base (Chung et al., 2022) (encoder-decoder architecture, 350M parameter model) as our LLM and the Eva-clip-g/14 (Sun et al., 2023a), a CLIP-based vision transformer, for our frozen image encoder. The QFormer follows the default architecture and initialization as in Dai et al. (2023). Since FlanT5-base has an encoder-decoder architecture, insertion bypasses the encoder entirely. Instead of feeding the Qformer outputs (query tokens) as inputs to the encoder, we feed them directly into the decoder by prepending them to the language tokens. The choice of bypassing the encoder is intuitive, since the encoder output space is likely to capture the semantics of the language prompt and therefore align better with the image semantics captured in the QFormer tokens.

Note that we do not intend to follow the training recipe for the InstructBLIP models. Indeed, it is impossible to replicate the model for two reasons: a) lack of information on training recipe, including learning rate schedules and dataset sampling ratios, and b) parts of the LAION dataset, the key component for pretraining, was taken offline so as to comply with U.S. federal law, cf. Thiel (2023). Furthermore, the high-level recipe calls for multi-epoch pretraining on a dataset with roughly 130M image-text pairs, which are beyond the scope of this work. Instead, we define a smaller setup, which still allows us fair testing of DeepInsert but with a reasonable compute budget. Recall that the end-to-end training phase for InstructBLIP involves two steps: the pretraining step with image-caption pairs, followed by the multi-task instruction finetuning step. Consequently, we structure the experiment in a similar vein, by first pretraining efficiency, followed by a comparison post multi-task instruction finetuning.

F.1.1 Pretraining Efficiency

Reflecting the InstructBLIP setup, we consider image-caption pretraining with the COCO captions dataset (Chen et al., 2015) as the first stage. The goal of this experiment is to compare the level of alignment between the QFormer and the LLM, as training proceeds. Specifically, we measure the best possible captioning score achieved within a fixed number of epochs (1, 10 and 20) for each framework. This is done in accordance with different hyperparameter and learning rate configurations available in the BLIP repository and for each variant, we report the highest performance across configurations. As shown in Fig. 11, we find that our framework not only achieves better peak performance, but also does it in much fewer epochs. We limit our report to 20 epochs as both models achieve over 95% of their respective peak scores within that frame (the peak is achieved at around 50 epochs).

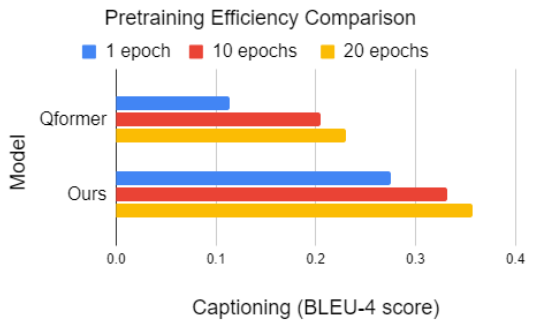


Figure 11: Pretraining comparison for Baseline (QFormer) vs. DeepInsert (Ours) pipelines. Higher score indicates that DeepInsert achieves better alignment, and in much fewer epochs.

F.1.2 Multi-task Training

Next, we consider instruction finetuning for two tasks: Captioning and Visual Question Answering (VQA). For each task, we sample from a fixed set of text prompts in every training iteration.

To emulate InstructBLIP's training stages, we first pretrain with image-caption pairs for 20 epochs, and then perform multi-task instruction finetuning for captioning and VQA for 15 combined epochs (uniformly sampling from the datasets without replacement). For captioning, we consider the COCO captions dataset as before and for visual question answering, we consider VQAv2 (Goyal et al., 2017) for training. Instruction finetuned VLMs can demontrate remarkable zero-shot abilities on unseen datasets, and so we also consider OKVQA (Outside Knowledge VQA) (Marino et al., 2019) for zero-shot evaluation. Coco Captions and VQAv2 share the same train-val-test image split that aims to prevent train-test information leakage for a fair evaluation.

Results are given in Table 7. We find that for the designated tasks, our method achieves significant improvements over the standard QFormer. We also see a remarkable gap in zero-shot OKVQA performance, with the grounded model even outperforming the reported performance of some of the largest BLIP-2 models (Li et al., 2023a). (InstructBLIP includes OKVQA during training, so no direct comparison is possible.)

Model	COCO Cap. (BLEU-4)	VQAv2 (%)	OKVQA (0-shot) (%)
Baseline	20.9	55.4	28.8
DI (Ours)	**36.2**	**66.8**	**39.0**

Table 7: Performance comparison for BLIP-inspired multi-task instruction finetuned models, including 0-shot evaluation on unseen dataset. We find that in our custom setup, bypassing the encoder and inserting directly to the decoder can yield significant performance gains.

F.2 Open-LLaVA-NeXT

Herein, we extend DeepInsert to a the newer LLaVA-Next-7B (the open reproduction) (Liu et al., 2024b; Chen and Xing, 2024), with a much larger vision context yielding performance improvements. Below, we report the average performances in Table 8, indicating the robustness of DeepInsert to exploit early layer redundancy even when number of multimodal tokens are much higher ($\sim$2k). We note here that unlike retraining the baseline models in the main paper, we directly used the checkpoint from Huggingface for the baseline due to limited compute, meaning the DeepInsert models had about 1-2k missing data points, which explains the slight performance drop.

Model	AI2D	GQA	POPE	MMMU	TextVQA (lite)	Avg.
LLaVA-NeXT	64.6	64.2	87.3	39.0	63.4	63.7
DI-4	63.7	63.8	87.4	36.9	64.3	63.2
DI-6	62.7	63.6	87.0	37.1	61.9	62.4

Table 8: Performance of DeepInsert on Open-LLaVA-NeXT-7B.

Figure 12: Comparison of training losses of LLaVA finetuning stage, for the baseline model against best performing grounded model (layer-4), revealing a tendency to achieve lower next token prediction loss and potentially better learning.

F.3 Training Loss Comparison on LLaVA

G Efficiency Analysis

The number of FLOPs performed by a transformer model in a forward pass can be analytically expressed as a function of the transformer's various hyperparameters such as model hidden dimension, feedforward layer dimension, and number of layers. In this section, we extend the analysis to DeepInsert and compare the theroetically expected FLOPs with the empirically measured runtime. Let the transformer have N layers with a hidden dimension of d_{model} and a feedforward dimension of d_{ff}. Let the number of heads for multihead attention be given by n_{head}. Assume the number of multimodal tokens is L_{mm} and the number of text tokens in a certain forward pass is L_{text}. Let the DeepInsert layer be given by N_{DI}.

The total number of FLOPs (additions plus multiplications) in one forward pass can be divided into the following parts: (i) the projections into Query (Q), Key (K), and Value (V) matrices, (ii) the self attention mechanism, (iii) the feedforward layers, and (iv) other computations such as activations, layer norm, etc. Since (iv) is negligible compared to (i), (ii), and (iii), we ignore it here.

Expanding on (i), the input $X \in \mathbb{R}^{L \times d_{\text{model}}}$ needs to be multiplied with projection matrices $W_q, W_k, W_v \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ to give Q, K, V respectively. For an input sequence of length L, each XW matrix multiplies require $2Ld^2_{\text{model}}$ operations including both additions and multiplications, giving us

$$\text{Projection FLOPs} = 6Ld^2_{\text{model}}. \quad (1)$$

Moving to (ii), we first need to compute the $QK^\top$ matrix, and since Q and K are both in $\mathbb{R}^{L \times d_{\text{model}}}$, this requires $n_{\text{head}} \times 2L^2 \frac{d_{\text{model}}}{n_{\text{head}}} = 2L^2 d_{\text{model}}$ FLOPs. We then apply a row-wise softmax to get the $L \times L$ attention matrix for each head, which requires another $2L^2 n_{\text{head}}$ FLOPs. Next, the $L \times L$ attention matrices are multiplied with the value matrices which takes up another $2L^2 d_{\text{model}}$ FLOPs across all heads. Finally, the output projection takes up another $2Ld^2_{\text{model}}$ FLOPs. Adding all these up, and using the approximation $n_{\text{head}} << d_{\text{model}}$, we have

$$\text{Attention FLOPs} = 4L^2 d_{\text{model}} + 2Ld^2_{\text{model}}. \quad (2)$$

Computing (iii) is more straightforward as the feedforward layers are pairs of linear layers, with dimensions $d_{\text{model}} \times d_{\text{ff}}$ and $d_{\text{ff}} \times d_{\text{model}}$. This gives us

$$\text{Feed forward FLOPs} = 4Ld_{\text{model}}d_{\text{ff}}. \quad (3)$$

Adding (1), (2), and (3) gives us our final analytical expression for the number of FLOPs per layer when the sequence length is L:

$$\begin{aligned}&\text{FLOPs per layer for } L \text{ tokens} \\ &= 8Ld^2_{\text{model}} + 4L^2 d_{\text{model}} + 4Ld_{\text{model}}d_{\text{ff}}. \quad (4)\end{aligned}$$

Using DeepInsert, the first N_{DI} layers only see L_{text} tokens, while the remaining $N - N_{\text{DI}}$ layers see all the $L_{\text{mm}} + L_{\text{text}}$ tokens. This give us the total number of FLOPs for deep insertion at layer N_{DI}:

$$\begin{aligned}&\text{FLOPs}(N_{\text{DI}}) \\ &= N\Big(8(L_{\text{text}} + L_{\text{mm}})d^2_{\text{model}} \\ &\qquad + 4(L_{\text{text}} + L_{\text{mm}})^2 d_{\text{model}} \\ &\qquad\qquad + 4(L_{\text{text}} + L_{\text{mm}})d_{\text{model}}d_{\text{ff}}\Big) \\ &- N_{\text{DI}}\Big(8L_{\text{mm}}d^2_{\text{model}} \\ &\qquad + 4(2L_{\text{text}} + L_{\text{mm}})L_{\text{mm}}d_{\text{model}} \\ &\qquad\qquad + 4L_{\text{mm}}d_{\text{model}}d_{\text{ff}}\Big). \quad (5)\end{aligned}$$

As we see from (5), the number of FLOPs decreases monotonically as a function of N_{DI}. Further, a larger number of multimodal tokens L_{mm} leads to a steeper decrease in FLOPs as we insert deeper into the network.

To validate (5), we empirically measure the runtime of LLaVA on 80 GB A100 GPUs and plot the both the runtime and the theoretically estimated FLOPs against the insertion layer in Fig. 13. As we

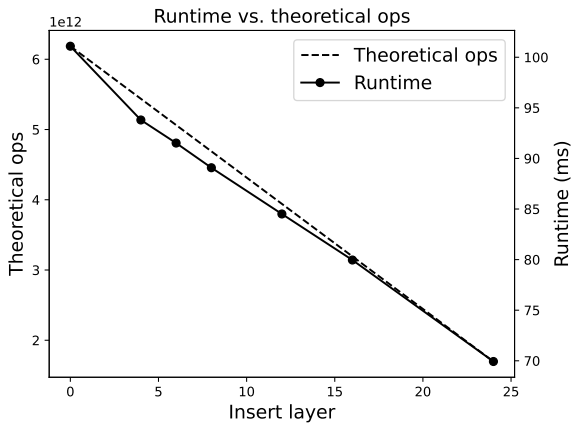


Figure 13: Comparing the model runtime with the theoretically estimated FLOPs from (5). As we see, both curves match very closely.

see in Fig. 13, there is a very close match between the theoretically estimated FLOPs of (5) and the empirically measured runtime of the forward pass of the model, suggesting that (5) can be a useful proxy for the computational efficiency of the model when insertion is done at various deep layers.

H Potential Risks

Efficiency gains can lower deployment costs, potentially enabling wider, sometimes unguarded use. If models hallucinate or provide ungrounded rationales, this may scale harmful outcomes (e.g., disinformation, low-friction generation of misleading content) when safeguards are absent. As with other MLLMs, the system remains vulnerable to adversarial prompts and distribution shifts. Efficiency itself neither increases nor decreases susceptibility, but wider, cheaper access can increase attack surface (e.g., more attempts, faster iteration by adversaries).

I AI Usage

AI models (part of Overleaf native tools) were used in improving the writing of the paper.