NVILA: Efficient Frontier Visual Language Models

Zhijian Liu[1,†] Ligeng Zhu[1,†] Baifeng Shi[3] Zhuoyang Zhang[2] Yuming Lou[6]
Shang Yang[2] Haocheng Xi[3] Shiyi Cao[3] Yuxian Gu[2,6] Dacheng Li[3] Xiuyu Li[3]
Haotian Tang[2] Yunhao Fang[4] Yukang Chen[1] Cheng-Yu Hsieh[5] De-An Huang[1]
An-Chieh Cheng[4] Jinyi Hu[2,6] Sifei Liu[1] Ranjay Krishna[5] Pavlo Molchanov[1]
Jan Kautz[1] Hongxu Yin[1,‡] Song Han[1,2,‡] Yao Lu[1,†,‡]

[1]NVIDIA [2]MIT [3]UC Berkeley [4]UC San Diego
[5]University of Washington [6]Tsinghua University
[†]Equal contribution [‡]Equal advisory

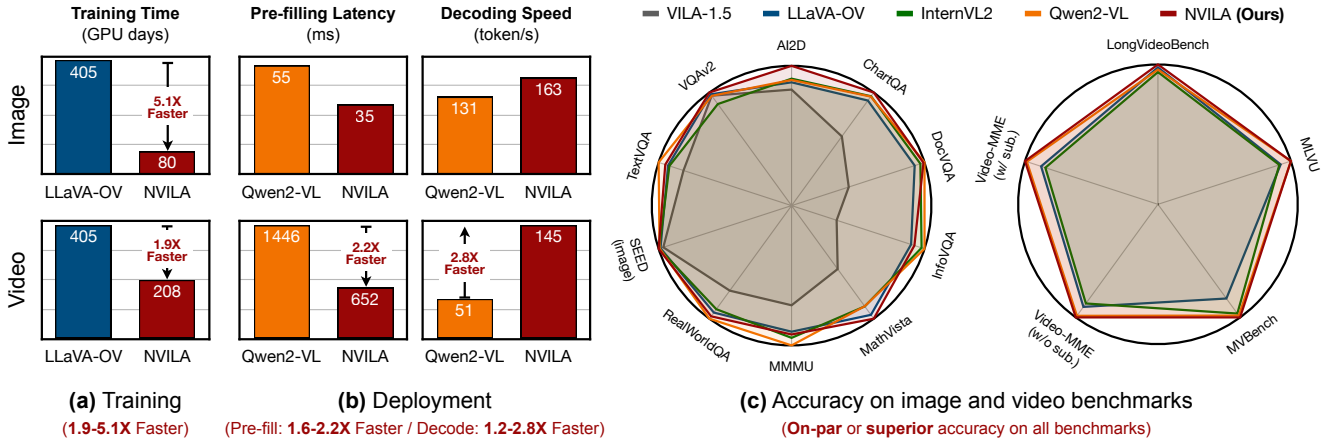

Figure 1. **NVILA – Efficient Frontier VLMs**. **(a)** NVILA trains image and video models 5.1× and 1.9× faster, respectively, than LLaVA-OneVision (OV), which is the only baseline model with publicly disclosed training costs. **(b)** Against Qwen2-VL, NVILA achieves a 1.6–2.2× measured speedup in the pre-filling stage and a 1.2–2.8× speedup during the decoding stage. **(c)** NVILA's efficiency is achieved without compromising accuracy; in fact, it delivers comparable or even superior accuracy across image and video benchmarks. All models in this table have 8B parameters. Training time in (a) is measured using NVIDIA H100 GPUs, while inference speed in (b) is measured using a single NVIDIA GeForce RTX 4090 GPU. Accuracy numbers in (c) are normalized relative to the highest score for each benchmark.

Abstract

Visual language models (VLMs) have made significant advances in accuracy in recent years. However, their efficiency has received much less attention. This paper introduces ***NVILA****, a family of open VLMs designed to optimize both efficiency and accuracy. Building on top of VILA, we improve its model architecture by first* ***scaling up*** *the spatial and temporal resolutions, and then* ***compressing*** *visual tokens. This "scale-then-compress" approach enables NVILA to efficiently process high-resolution images and long videos. We also conduct a systematic investigation to enhance the efficiency of NVILA throughout its entire lifecycle, from training to deployment. NVILA matches or surpasses the accuracy of many leading open and proprietary VLMs across a wide range of image and video benchmarks. At the same time, it reduces training costs by* ***1.9-5.1****×, prefilling latency by* ***1.6-2.2****×, and decoding latency by* ***1.2-2.8****×.*

1. Introduction

Visual language models (VLMs) have shown remarkable abilities in processing and integrating both visual and textual information, enabling advanced vision-language interactions and dialogues. In recent years, the research community has made tremendous progress in enhancing their accuracy [1–5]

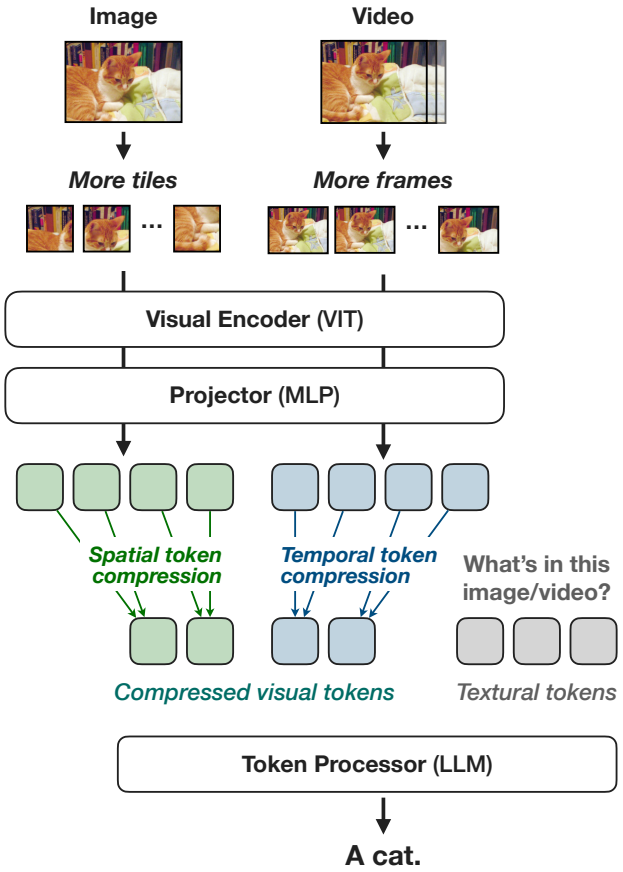

Figure 2. **Model architecture**.

and broadening their applications across diverse domains, including robotics [6–8], autonomous driving [9], and medical applications [10, 11]. However, there has been much less focus on improving their efficiency.

VLMs are expensive across multiple dimensions. First, *training a VLM is time-consuming*. For example, training a state-of-the-art 7B VLM [4] can take up to 400 GPU days, let alone even larger models. This creates a significant entry barrier for researchers. Second, VLMs often require adaptation when applied to specialized domains (*e.g.*, medical imaging), but *fine-tuning a VLM is memory-intensive*. For example, fully fine-tuning a 7B VLM can require over 64GB of GPU memory, far beyond the available memory of most consumer-level GPUs. Finally, VLMs are often deployed in edge applications with limited computational budget (*e.g.*, laptops, robots), so *deploying a VLM is resource-constrained*. Addressing these challenges requires a systematic solution to improve VLM efficiency across all these dimensions.

In this paper, we introduce **NVILA**, a family of open VLMs designed to optimize both efficiency and accuracy. Building on VILA [2], we improve its model architecture by first scaling up the spatial and temporal resolution, followed by compressing visual tokens. "Scaling" preserves more details from visual inputs, raising the accuracy upper bound, while "compression" squeezes visual information to fewer tokens, improving computational efficiency. This "*scale-then-compress*" strategy allows NVILA to process high-resolution images and long videos both effectively and efficiently. In addition, we conduct a systematic study to optimize the efficiency of NVILA throughout its entire lifecycle, including training, fine-tuning, and deployment.

Thanks to these innovations, NVILA is efficient and accurate. It reduces training costs by **1.9-5.1**×, prefilling latency by **1.6–2.2**×, and decoding latency by **1.2–2.8**×. It also matches or surpasses the accuracy of leading open VLMs [2, 3, 5] and proprietary VLMs [12, 13] across a wide range of image and video benchmarks.

2. Approach

In this section, we begin by designing an efficient model architecture for NVILA, first by *scaling up* spatial and temporal resolutions, and then by *compressing* the visual tokens. Next, we present strategies to improve NVILA's efficiency across its *entire lifecycle*—from training and fine-tuning to deployment. Unless otherwise specified, all analysis in this section will be based on the 8B model.

2.1. Efficient Model Architecture

We build NVILA on top of VILA [2]. As in Figure 2, it is an auto-regressive VLM composed of three components: a *visual encoder* that extracts features from visual inputs (*e.g.*, images, videos); a *projector* that aligns embeddings across visual and language modalities; and a *token processor*, typically instantiated with a LLM, which takes both visual and language tokens as input and outputs language tokens. Specifically, NVILA uses SigLIP [14] as its vision encoder, a two-layer MLP as its projector, and Qwen2 [15] of different sizes as its token processor.

The original VILA has very *limited spatial and temporal resolutions*: *i.e.*, it resizes all images to 448×448, regardless of their original size or aspect ratio, and samples up to 14 frames from videos*. Both spatial resizing and temporal sampling will introduce significant loss of information, limiting the model's capability to effectively process larger images and longer videos. This can also be observed in Table 7 and Table 8, where VILA lags behind leading VLMs, especially on text-heavy and long-video benchmarks.

In this paper, we advocate for the "***scale-then-compress***" paradigm, where we first *scale up* the spatial/temporal resolutions to improve *accuracy*, and we then *compress* the visual tokens to improve *efficiency*. Scaling resolutions up improves the performance ceiling, but doing so alone will

*This is the configuration for VILA-1.5 40B. Their other variants, such as VILA-1.5 3B, only use 384×384 resolution and 8 frames.

significantly increase the computational cost. For example, doubling the resolution will double the number of visual tokens, which will increase both training and inference costs by more than 2×, as self-attention scales quadratically with the number of tokens. We can then cut this cost down by compressing spatial/temporal tokens. Compressed visual tokens have a higher information density, allowing us to preserve or even improve spatial and temporal details with fewer total tokens.

2.1.1 Spatial "Scale-Then-Compress"

For spatial scaling, it is very natural to directly increase the image resolution of the vision encoder, for example, to 896×896. While this may improve performance, applying a uniformly high resolution to all images would be inefficient, especially for smaller images that do not require extensive detail. To address this, we apply S^2 [16] to efficiently extract multi-scale high-resolution features with image tiling. For example, given a vision encoder pre-trained at 448^2 resolution and an input image with any size, S^2 first resizes the image into multiple scales (*e.g.*, 448^2, 896^2, 1344^2), and for each scale, it splits the image into tiles of 448^2. Each tile is then individually processed by the encoder. The feature maps of each tile from the same scale are stitched back together into the feature map of the whole image at that scale. Finally, feature maps from different scales are interpolated into the same size and concatenated on the channel dimension.

S^2 always resizes images into square, regardless of the original aspect ratio. This can cause distortion, particularly for images that are either tall and narrow or short and wide. To address this, we propose *Dynamic-S^2*, which adaptively processes images with varying aspect ratios. Dynamic-S^2 follows the approach of S^2 but, at the largest image scale, instead of resizing to a square, it adjusts the image dimensions to the closest size that maintains the original aspect ratio and is divisible by 448^2 tiles. This is inspired by the dynamic resolution strategy in InternVL [17]. After processing the tiles, the feature maps from all scales are interpolated to match the size of the largest scale and concatenated.

Equipped with Dynamic-S^2, the model benefits from high-resolution information from the image, resulting in a up to **30% accuracy improvements** on text-heavy benchmarks (Table 1). Our goal, then, shifts to compressing the spatial tokens. VILA [2] finds that applying a simple 2×2 spatial-to-channel (STC) reshape can reduce the token count by a factor of 4 without sacrificing accuracy. However, pushing this further results in a notable drop in performance: *i.e.*, a nearly 10% decrease in accuracy on DocQA, when reducing the number of minimal tiles and increasing the STC to 3×3. We hypothesize that *more aggressive reductions make the projector significantly harder to train*. To address this, we introduce an additional visual encoder pre-training stage to jointly tune the vision encoder and projectors. This helps recover most of the accuracy loss from spatial token reduction, achieving a **2.4**× speedup in both training and inference.

There are many alternative designs for spatial token compression, such as TokenLearner from RT-1 [6] and Perceiver Resampler from MiniCPM-V [18]. However, with the same token reduction ratio, these learnable compression methods surprisingly do not perform better than the simple spatial-to-channel design, even with an additional stage 1.5. We believe this is more of an optimization problem and is beyond the scope of this paper.

2.1.2 Temporal "Scale-Then-Compress"

For temporal scaling, we simply increase the number of uniformly sampled frames from the input video. Following previous methods [19], we train the model with additional video-supervised fine-tuning (SFT) to extend its capability to process more frames. From Table 8, extending the number of frames from 8 to 32 can increase the model's accuracy on Video-MME by **more than 5%**. However, this will also increase the number of visual tokens by 4×.

Similar to spatial token compression, we will then reduce these visual tokens. Since there is intrinsic temporal continuity in the video, we adopt *temporal averaging* [20] for compression, which first partitions the frames into groups and then temporally pools visual tokens within each group. This will reduce temporal redundancy (since consecutive frames often contain similar information) while still retaining important spatiotemporal information. Empirically, compressing the visual tokens by 4× leads to an acceptable accuracy drop. When compared to the original baseline with the same number of tokens, the first scaled and then expanded result costs almost the same[†], but has much higher accuracy. We have also used this approach to further scale the number of frames and the compression ratio, leading to a state-of-the-art 7B model on this benchmark (see Table 8).

2.2. Efficient Training

While state-of-the-art VLMs boast impressive capabilities, training such a VLM is often costly and compute-intensive. This section explores system-algorithm co-design to enable efficient VLM training. On the algorithm front, we examine a novel unsupervised dataset pruning method to streamline training data. At the system level, we investigate FP8 mixed precision for acceleration.

2.2.1 Dataset Pruning

In order to improve model accuracy, previous work [4, 21, 22] kept grabbing high quality SFT datasets from various

[†]We will need to run visual encoder for more frames, but this is usually not the runtime bottleneck.

Table 1. **Spatial "scale-then-compress"**. Increasing the spatial resolution with Dynamic-S^2 can greatly improve the model's accuracy, particularly on text-heavy benchmarks. Compressing the visual tokens with spatial pooling can effectively reduce both the number of tiles and tokens per tile, with moderate accuracy loss. This loss can be further reduced by adding an additional visual encoder pre-training (VEP) stage. In this and following tables, *"IM-10"* refers to the average validation scores from the 10 benchmarks listed in Table 7.

	Spatial Pooling	#Tokens/Tile	#Tiles/Image	AI2D	DocVQA	TextVQA	IM-10
Baseline (VILA-1.5)	2×2	256 (=16×16)	1	87.0	61.3	67.5	61.2
Scale (Dynamic-S^2)	2×2	256 (=16×16)	9-12	90.1	91.1	77.0	71.5
Scale + Compress	3×3	121 (=11×11)	1-12	87.4	82.3	74.1	67.1
Scale + Compress + VEP	3×3	121 (=11×11)	1-12	89.8	88.8	76.1	70.8
Alternative Designs							
TokenLearner	–	121	1-12	90.0	86.5	75.6	69.8
Perceiver Resampler	–	121	1-12	76.8	71.8	65.3	59.4

Table 2. **Temporal "scale-then-compress"**. *Scaling up* the temporal resolution can improve the model's video understanding performance. *Compressing* the visual tokens with temporal averaging can effectively reduce the number of tokens with only a marginal accuracy drop.

	#Frames	Temporal Pooling	#Tokens/Video	Video-MME (w/o sub.)			
				Short	Medium	Long	Overall
Baseline (VILA-1.5)	8	1×	2048 (=16^2×8)	65.4	53.8	47.7	55.7
Scale	32	1×	8192 (=16^2×32)	73.2	58.9	50.9	61.0
Scale + Compress	32	4×	2048 (=16^2×32/4)	73.7	56.7	50.0	60.1
Scale + Compress	256	8×	8192 (=16^2×256/8)	**75.0**	**62.2**	**54.8**	**64.0**

sources and can show improvement on Benchmark scores. However, *not all data contributes equally to the model* and continuous growth of datasets lead to much redundancy. In NVILA, we follow the "Scale-Then-Compress" concept to first increase our SFT dataset mixture and then trying to compress the dataset. However, selecting high-quality examples from various sources is challenging. While there have been explorations of vision inputs [23–25] and text-only inputs [26–28], few studies have addressed this problem in VLM training, where images and texts are mixed during training. NVILA's training involves more than 100M data, making it necessary to prune the training set while maintaining accuracy.

Inspired by recent works in knowledge distillation [29], we leverage *DeltaLoss* to score the training set:

$$D' = \bigcup_{i=1}^{K} \text{top-}K \left\{ \log \frac{p_{\text{large}}(x)}{p_{\text{small}}(x)} \middle| x \in D_i \right\}, \qquad (1)$$

where D_i is the i-th subset of the full fine-tuning datasets and D' is the pruned training set. $p_{\text{large}}(x)$ and $p_{\text{small}}(x)$ are the output probabilities on the answer tokens. The main motivation is to **filter out examples that are either too easy or too hard**. To elaborate,

- If both answer correctly or wrongly, $\log \frac{p_{\text{large}}(x)}{p_{\text{small}}(x)}$ is close to 0.
- When the small model answers correctly but the large model fails, $\log \frac{p_{\text{large}}(x)}{p_{\text{small}}(x)}$ becomes negative, suggesting these examples tend to distract learning and will eventually be forgotten by a more powerful model.
- When the small model answers incorrectly but the large model solves it, $\log \frac{p_{\text{large}}(x)}{p_{\text{small}}(x)}$ is positive, suggesting these examples provide strong supervision, as challenging for small models but learnable by larger ones.

Thereby we can apply *DeltaLoss* to each sub-dataset and prune the training set with different ratios.

To evaluate the data pruning criterion, we compare *DeltaLoss* and the random pruning baseline in Table 3. For random pruning, data is randomly selected and we run the results three times and report the average. For cluster pruning, we apply k-means clustering with siglip features and prune the data evenly across each centroid. Our experiments report the average performance across 10 benchmarks, with a focus on key tasks to demonstrate the method's effectiveness. We examine three pruning threshold 10%, 30% and 50% and notice that *DeltaLoss* consistently outperforms the random baseline, especially on the GQA and DocVQA tasks the random pruning shows a significant performance degradation while DeltaLoss stays accurate. We notice 50% is a

Table 3. **Dataset pruning on NVILA Recipe**. DeltaLoss consistently rivals other data selection methods and shows negligible performance drop when pruning 50% of data.

Method	IM-10	MMMU	DocVQA	TextVQA
100% (baseline)	75.6	48.0	90.1	78.8
50%				
DeltaLoss [29]	**75.5**	**48.1**	**89.7**	**78.4**
Cluster Pruning	74.5	47.8	88.3	77.0
Random Pruning	74.0	47.6	87.1	76.6
30%				
DeltaLoss [29]	**74.0**	**47.8**	**87.9**	**76.4**
Cluster Pruning	73.5	47.7	84.1	76.0
Random Pruning	73.1	47.7	82.9	75.6
10%				
DeltaLoss [29]	**72.4**	47.1	**84.4**	**74.5**
Cluster Pruning	72.2	**47.4**	79.6	73.2
Random Pruning	72.0	47.0	77.3	72.6

Table 4. **FP8 training**. FP8 accelerates the training of NVILA while maintaining the accuracy, especially when gradient checkpointing (GC) is not enabled. In this table, the throughput results are obtained with the maximum achievable batch size (BS) on 64 H100 GPUs. Video-MME results come from an 8-frame setting and with subtitle information.

	GC	BS	Throughput	MMMU	Video-MME
BF16	✗	4	199.2 (1.0×)	47.9	52.9
FP8	✗	16	390.1 (2.0×)	47.0	53.0
BF16	✓	30	491.7 (2.5×)	47.8	53.1
FP8	✓	36	579.9 (2.9×)	47.7	53.0

relatively safe threshold where the average score maintains competitive while the training can be speedup by 2×. Thus we set the threshold to 50% for later experiments.

2.2.2 FP8 Training

FP16 [30] and BF16 [31] are standard precisions for model training, since they offer acceleration without accuracy loss, supported natively by NVIDIA GPUs. With the advent of the NVIDIA Hopper and Blackwell architectures, new GPUs now provide native support for FP8, which has emerged as a promising precision due to its potential for larger computational and memory efficiency.

Many researchers have already applied FP8 to LLM training. NVIDIA's Transformer Engine performs matrix multiplications (GEMM) in FP8 precision, resulting in faster training speeds. FP8-LM [32] builds upon this by also quantizing the gradients, weight master copy, and first-order momentum into FP8, thereby reducing communication overhead and memory footprint. COAT [33] further compresses activations and the optimizer's second-order momentum to enhance memory efficiency while maintaining accuracy.

Table 5. **Fine-tuning recipe**. Our recommendation is to tune the LLM with either LoRA or QLoRA and to tune ViT's layer normalization (LN) layers with a much smaller learning rate. This setup achieves competitive accuracy and is also the most memory- and compute-efficient. All experiments use a batch size of 1 with gradient checkpointing disabled, and throughput is measured on a single NVIDIA A100 80GB GPU. For settings with {1,5,10,50}, we select the learning rate ratio from this set that gives the best results for each benchmark. "*FT-5*" refers to the average accuracy across AITZ [35], ALFRED [36], nuScenes [37], PathVQA [38], and Widget Caption [39].

ViT	LLM	Memory (GB)	Throughput (iter/s)	$\text{LR}_{\text{LLM}}/\text{LR}_{\text{ViT}}$	Accuracy (FT-5)
LoRA	LoRA	20.1	3.4	1	69.2
				{1,5,10,50}	**71.8**
LN	LoRA	19.2	4.5	1	63.5
				{1,5,10,50}	**71.4**
FT	LoRA	21.9	4.2	1	64.0
				{1,5,10,50}	**70.1**
LoRA	QLoRA	11.1	2.6	1	63.0
				{1,5,10,50}	**70.8**
LN	QLoRA	10.2	3.1	1	62.7
				{1,5,10,50}	**70.9**
FT	FT	63.5	6.1	1	77.7

In this paper, we borrow the FP8 implementation from COAT [33] to accelerate the training of NVILA. One key difference between LLM and VLM training workloads lies in the variability of sequence lengths across batches. In LLM training, samples generally have uniform lengths, and increasing the batch size beyond a certain point has minimal effect on training throughput. However, in VLM training, samples can vary significantly in length: video samples may require tens of thousands of tokens, image samples may need hundreds, and text-only samples require far fewer. As a result, workloads with fewer tokens are generally underutilized and can benefit greatly from increasing the batch size. As shown in Table 4, applying FP8 to both weights and activations allows NVILA to increase the batch size from 4 to 16, resulting in a 2× speedup. When gradient checkpointing is enabled, quantizing activations becomes less essential. Instead, we integrate the cross-entropy kernel from Liger [34] to reduce peak memory usage due to Qwen's large vocabulary size. In this case, FP8 training can still provide a 1.2× speedup compared to BF16 training.

Table 6. **Quantization recipe**. While W4A16 quantization on LLM backbone may introduce small accuracy drop, W8A8 quantization on ViT is nearly lossless.

ViT	LLM	AI2D	MMMU	VideoMME	TTFT (s)
FP16	FP16	91.0	50.7	63.9	0.90
FP16	W4A16	90.9	49.2	62.0	0.77
W8A8	W4A16	90.9	49.3	62.1	0.65

2.3. Efficient Fine-Tuning

Once a foundation VLM is trained, domain-specific fine-tuning is needed to adapt the model for specialized tasks or domains. While fine-tuning effectively improves domain-specific vocabulary and concepts, conventional Parameter Efficient Fine-Tuning has been focusing on LLM and text-related tasks. How to best fine-tune a VLM remains less explored. In NVILA, we find that (i) the learning rate should be set differently for ViT and LLMs (ii) the tuning parts should be chosen dependently for different downstream tasks.

When fine-tuning the vision encoder (ViT) and language model (LLM) together using PEFT methods, we observe that the learning rate should be set differently for VE and LLM: the learning rate for the ViT part will be 5-50$\times$ smaller than that for the LLM part. On the other hand, we also observe that fine-tuning the vision encoder with Layernorm can achieve comparable performance as LoRA (Table. 5) while being more computationally efficient: it can reduce the training time by 25% compared to applying LoRA for the vision encoder. With the curated configuration setup, NVILA can be quickly fine-tuned to various downstream tasks under 24 GB memory with on-par performance.

2.4. Efficient Deployment

VLMs are often integrated in edge applications as robotic where computational budget is tight. We develop a specialized inference engine with quantization techniques to efficiently deploy NVILA. The inference process is divided into two phases: prefilling and decoding. In the compute-bounded prefilling stage, we first apply token compression techniques (Section 2.1) to reduce the inference workload for LLM backbone, after which the vision tower becomes the primary bottleneck, accounting for over 90% of the prefilling latency. To tackle this, we implement W8A8 quantization for the vision tower to reduce NVILA's Time-To-First-Token (TTFT) in this compute-bounded stage. For the memory-bounded decoding stage, we follow AWQ [40] for W4A16 quantization of the LLM backbone to accelerate. We further optimize the original AWQ implementation by introducing FP16 accumulation to the W4A16 GEMM kernels, resulting to a total 1.7$\times$ kernel speedup without compromising accuracy. A detailed comparison is in Figure. 3.

3. Experiments

3.1. Training Details

We follow a five-stage pipeline to train NVILA: (1) *projector initialization*, (2) *visual encoder pre-training*, (3) *token processor pre-training*, (4) *image instruction-tuning*, and (5) *video instruction-tuning*. Among them, Stages 1, 3, and 4 are also included in VILA training. The additional Stage 2 is used to recover the accuracy loss due to spatial token compression (as in Table 1), and the additional Stage 5 is helpful for extending the model's long video understanding capability.

Our implementation is built upon PyTorch 2.3.0 [41, 42] and Transformers 4.46.0 [43]. We use DeepSpeed 0.9.5 [44] to shard large models across devices and use gradient checkpointing to reduce memory usage. We adopt FlashAttention-2 [45] to accelerate training in both the LLM and visual encoder. We also implement functional-preserving, on-the-fly sequence packing to fuse samples with different lengths, which leads to an around 30% speedup. We train all models using 128 NVIDIA H100 GPUs with a global batch size of 2048 across all stages. All optimizations are carried out using AdamW with no weight decay. We adopt a cosine learning rate decay schedule with a linear warm-up for the first 3% of the schedule. The initial learning rate varies across stages, as detailed in Table A1.

3.2. Accuracy Results

3.2.1 Image Benchmarks

As presented in Table 7, we conduct comprehensive evaluations across a diverse range of image benchmarks: AI2D [46], ChartQA [47], DocVQA [48], InfographicVQA [49], MathVista [50], MMMU [51] (with zero-shot CoT), RealworldQA [52], SEED-Bench [53], TextVQA [54], and VQAv2 [55]. Our NVILA performs comparably to top open-source models in each size category, including Qwen2-VL [5], InternVL [3], and Pixtral. For general visual question answering tasks (ChartQA, DocVQA, InfoVQA, TextVQA, VQAv2, Seed), NVILA-8B and NVILA-15B achieve competitive or even better results compared to proprietary models (GPT-4o, Gemini). In science-related benchmarks (AI2D), NVILA-8B achieves state-of-the-art performance among open-source models. When scaling to 15B, NVILA demonstrates competitive performance with proprietary models. Furthermore, on reasoning and knowledge benchmarks such as MMMU, RealworldQA, and MathVista, scores improve more when the model size increases. For benchmarks that require OCR capability such as TextVQA, AI2D, ChartQA, DocVQA, InfoVQA, 8B model can also do a great job.

Table 7. **Image benchmarks**. We mark the best performance **bold** and the second-best underlined.

		AI2D	ChartQA	DocVQA	InfoVQA	MathVista	MMMU			Real-WorldQA	SEED	TextVQA	VQAv2
		test	test	test	test	testmini	val	test	pro		image	val	testdev
GPT-4o	–	94.2	85.7	92.8	79.2	63.8	69.1	64.7	51.9	75.4	76.2	77.4	78.7
Claude 3.5 Sonnet	–	94.7	90.8	85.2	74.3	67.7	68.3	63.7	51.5	60.1	–	74.1	70.7
Gemini 1.5 Pro	–	94.4	87.2	93.1	81.0	63.9	62.2	57.6	43.5	70.4	–	78.7	80.2
LLaVA-1.5	7B	55.5	17.8	28.1	25.8	25.6	35.7	–	–	54.8	66.1	58.2	78.5
VILA-1.5	8B	76.6	52.7	40.6	25.9	36.7	38.6	32.7	–	52.7	73.8	68.5	83.0
Cambrian-1	8B	73.0	73.3	77.8	41.6	49.0	42.7	–	–	64.2	74.7	71.7	81.2
Florence-VL	8B	74.2	74.7	84.9	51.7	55.5	43.7	–	–	64.2	74.9	74.2	84.7
LLaVA-OneVision	8B	81.4	80.0	87.5	68.8	63.2	48.8	42.8	24.1	66.3	75.4	78.3	84.0
Llama 3.2	11B	91.9	83.4	88.4	–	51.5	50.7	–	–	–	–	–	75.2
InternVL2	8B	83.8	83.3	91.6	74.8	58.3	51.2	42.6	29.0	64.2	76.2	77.4	76.7
Qwen2-VL	8B	83.0	83.0	**94.5**	**76.5**	58.2	**54.1**	**46.6**	**30.5**	**70.1**	76.0	**84.3**	82.9
NVILA	8B	**92.3**	**86.1**	93.7	70.7	**65.4**	49.9	44.4	27.8	68.6	**76.5**	80.1	**85.4**
LLaVA-1.5	13B	61.1	18.2	30.3	29.4	27.7	37.0	–	–	55.3	68.2	61.3	80.0
VILA-1.5	13B	79.9	59.5	58.6	30.4	42.7	37.9	33.6	–	57.5	72.6	65.0	82.8
Cambrian-1	13B	73.6	73.8	76.8	–	48.0	40.0	–	–	63.0	74.4	72.8	–
Pixtral	12B	79.0	81.8	90.7	50.8	58.0	52.5	–	–	65.4	–	75.7	80.2
NVILA	15B	**94.1**	**86.9**	**94.0**	**73.5**	**66.1**	56.7	**51.8**	**33.8**	**69.5**	**76.6**	**80.0**	**84.8**
LLaVA-NeXT	34B	–	–	–	–	46.5	48.1	44.5	22.9	–	75.9	69.5	83.7
Cambrian-1	34B	79.7	75.6	75.5	46.0	53.2	49.7	–	–	67.8	75.3	76.7	83.8
VILA-1.5	40B	88.9	67.8	58.6	38.4	49.3	51.9	46.9	25.0	60.8	69.1	73.6	84.3
InternVL2	40B	87.1	86.2	93.9	78.7	63.7	55.2	47.4	34.2	71.8	78.2	83.0	–
LLaVA-OneVision	72B	85.6	83.7	91.3	74.9	67.5	56.8	52.3	31.0	71.9	75.4	80.5	85.2
NVLM-D-1.0	78B	94.2	86.0	92.6	–	65.2	59.7	54.6	–	69.7	–	82.1	85.4
Llama 3.2	90B	92.3	85.5	90.1	–	57.3	60.3	–	39.5	–	–	–	–

3.2.2 Video Benchmarks

We evaluate our models on a range of video understanding benchmarks [56–59], spanning short videos of a few seconds to longer videos up to an hour in duration. Table 8 presents the performance of NVILA compared to baseline models [4, 5, 19, 60–62]. NVILA features long-context capability and can process up to 256 frames. With the scale-then-compress design, NVILA-8B achieves impressive results, setting new state-of-the-art performance across all benchmarks. NVILA reaches performance levels comparable to GPT-4o mini with only 8B parameters and outperforms many larger models.

3.3. Efficiency Results

NVILA achieves competitive performance on image and video benchmarks while maintaining efficiency through "scale-then-compress". Architecturally, We initially scale up to native resolution (1–12× more tiles), then compress tokens by 2.4×, achieving higher accuracy with slightly more tokens than previous solutions. Dataset-wise, we curate a diverse 10M sample dataset, compress it using DeltaLoss, and prune to a high-quality 5M subset, consistently outperforms LLaVA-OneVision, which trained on 8M+ data. Besides, we integrate FP8 for acceleration, optimize learning rates for fine-tuning, and use W8A8 format to improve latency and throughput. These full-stack optimizations enable NVILA to train with fewer resources while achieving better performance, less memory usage, and faster inference.

We compare NVILA's inference performance against Qwen2-VL [5] as shown in Figure 3. For a fair comparison, both models process video inputs by sampling 64 frames, with all experiments conducted on a single NVIDIA RTX 4090 GPU. Qwen2-VL is quantized to W4A16 and deployed with vLLM [63], a LLM/VLM serving engine with state-of-the-art inference speed. For NVILA, we quantize the LLM backbone to W4A16 and vision tower to W8A8. With our specialized inference engine, NVILA achieves up to 2.2× speedup in pre-filling stage and up to 2.8× higher decoding throughput over Qwen2-VL.

Table 8. **Video benchmarks**.

			ActivityNet-QA		LongVideoBench		MLVU	MVBench	NExT-QA	Video-MME	
		#F	acc.	score	val	test	m-avg	test	mc	w/o sub.	w/ sub.
GPT-4o mini	–	–	–	–	56.5	58.8	–	–	–	64.8	68.9
GPT-4o	–	–	61.9	–	66.7	66.7	64.6	–	–	71.9	77.2
VILA-1.5	8B		–	–							
LLaVA-NeXT-Video	7B	32	53.5	3.2	43.5	43.5	–	33.7	–	46.5	–
Video-XL	7B	2048	–	–	49.5	51.3	64.9	55.3	77.2	55.5	61.0
InternVL2	8B	64	–	–	54.6	–	64.0	65.8	–	56.3	59.3
LLaVA-OneVision	8B	32	56.6	–	56.5	–	64.7	56.7	79.4	58.2	61.5
Oryx-1.5	8B	128	–	–	56.3	–	67.5	67.6	81.8	58.8	64.2
LongVILA	7B	256	59.5	–	57.1	–	–	67.1	80.7	60.1	65.1
LongVU	7B	1fps	–	–	–	–	65.4	66.9	–	60.6	–
Qwen2-VL	8B	2fps	–	–	55.6	56.8	65.5	67.0	–	63.3	69.0
NVILA	8B	256	**60.9**	**3.7**	**57.7**	**58.7**	**70.1**	**68.1**	**82.2**	**64.2**	**70.0**

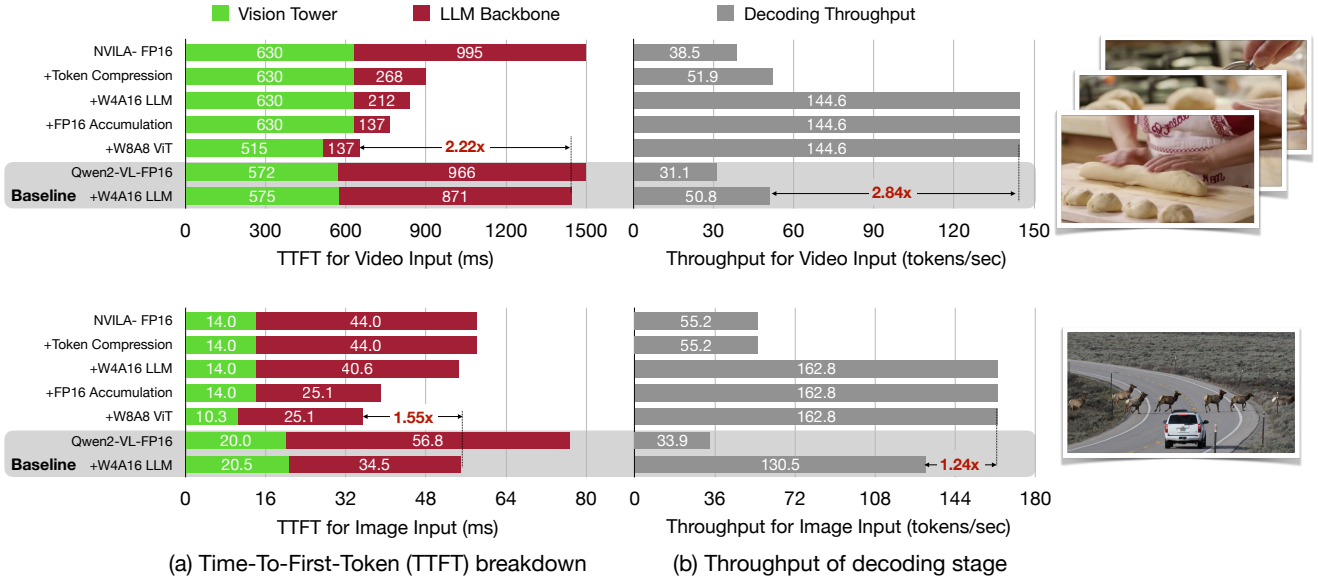

Figure 3. NVILA demonstrates superior inference efficiency over the Qwen2-VL model [5] for both image and video understanding tasks. We benchmark NVILA-7B against Qwen2-VL-7B. Qwen2-VL-7B is served by vLLM [63] for W4A16 LLM quantization, while NVILA is quantized and deployed with our specialized inference engine. We ablate the efficiency gains achieved with different optimization techniques introduced in NVILA. NVILA demonstrates 1.6-2.2× faster prefilling and up to 2.8× higher decoding throughput compared to Qwen2-VL.

4. Conclusion

This paper introduces NVILA, a family of open VLMs designed to strike an optimal balance between efficiency and accuracy. By adopting the "scale-then-compress" paradigm, NVILA can efficiently process high-resolution images and long videos while maintaining high accuracy. We also systematically optimize its efficiency across the entire lifecycle, from training to fine-tuning to inference. NVILA delivers performance that matches or exceeds current leading VLMs, while being significantly more resource-efficient. We hope NVILA can empower researchers and developers to fully unlock its potential across a wide range of applications and research domains.

References

[1] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[2] Ji Lin, Hongxu Yin, Wei Ping, Yao Lu, Pavlo Molchanov, Andrew Tao, Huizi Mao, Jan Kautz, Mohammad Shoeybi, and Song Han. VILA: On Pre-training for Visual Language Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[3] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. InternVL: Scaling up Vision Foundation Models and Aligning for Generic Visual-Linguistic Tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[4] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy Visual Task Transfer. *arXiv:2408.03326*, 2024.

[5] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *arXiv:2409.12191*, 2024.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *Robotics: Science and Systems (RSS)*, 2022.

[7] Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and Wang He. NaVid: Video-based VLM Plans the Next Step for Vision-and-Language Navigation. In *Robotics: Science and Systems (RSS)*, 2024.

[8] An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Xueyan Zou, Jan Kautz, Erdem Biyik, Hongxu Yin, Sifei Liu, and Xiaolong Wang. NaVILA: Legged Robot Vision-Language-Action Model for Navigation. *arXiv:2412.04453*, 2024.

[9] Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. DriveVLM: The Convergence of Autonomous Driving and Large Vision-Language Models. In *Conference on Robot Learning (CoRL)*, 2024.

[10] Khaled Saab, Tao Tu, Wei-Hung Weng, Ryutaro Tanno, David Stutz, Ellery Wulczyn, Fan Zhang, Tim Strother, Chunjong Park, Elahe Vedadi, et al. Capabilities of Gemini Models in Medicine. *arXiv:2404.18416*, 2024.

[11] Vishwesh Nath, Wenqi Li, Dong Yang, Andriy Myronenko, Mingxin Zheng, Yao Lu, Zhijian Liu, Hongxu Yin, Yee Man Law, Yucheng Tang, Pengfei Guo, Can Zhao, Ziyue Xu, Yufan He, Greg Heinrich, Stephen Aylward, Marc Edgar, Michael Zephyr, Pavlo Molchanov, Baris Turkbey, Holger Roth, and Daguang Xu. VILA-M3: Enhancing Vision-Language Models with Medical Expert Knowledge. *arXiv:2411.12915*, 2024.

[12] OpenAI. GPT-4o, 2024.

[13] Anthropic. Claude 3.5, 2024.

[14] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid Loss for Language Image Pre-Training. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[15] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 Technical Report. *arXiv:2407.10671*, 2024.

[16] Baifeng Shi, Ziyang Wu, Maolin Mao, Xin Wang, and Trevor Darrell. When Do We Not Need Larger Vision Models? In *European Conference on Computer Vision (ECCV)*, 2024.

[17] Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, Ji Ma, Jiaqi Wang, Xiaoyi Dong, Hang Yan, Hewei Guo, Conghui He, Botian Shi, Zhenjiang Jin, Chao Xu, Bin Wang, Xingjian Wei, Wei Li, Wenjian Zhang, Bo Zhang, Pinlong Cai, Licheng Wen, Xiangchao Yan, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. How Far Are We to GPT-4V? Closing the Gap to Commercial Multimodal Models with Open-Source Suites. *arXiv:2404.16821*, 2024.

[18] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. MiniCPM-V: A GPT-4V Level MLLM on Your Phone. *arXiv:2408.01800*, 2024.

[19] Fuzhao Xue, Yukang Chen, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, Ethan He, Hongxu Yin, Pavlo Molchanov, Jan Kautz, Linxi Fan, Yuke Zhu, Yao Lu, and Song Han. LongVILA: Scaling Long-Context Visual Language Models for Long Videos. *arXiv:2408.10188*, 2024.

[20] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *European Conference on Computer Vision (ECCV)*, 2016.

[21] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Austin Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A Fully Open, Vision-Centric Exploration of Multimodal LLMs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[22] Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What Matters When Building Vision-Language Models? In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[23] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via Proxy: Efficient Data Selection for Deep Learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[24] Hu Xu, Saining Xie, Xiaoqing Ellen Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, and Christoph Feichtenhofer. Demystifying CLIP Data. In *International Conference on Learning Representations (ICLR)*, 2024.

[25] Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. SemDeDup: Data-Efficient Learning at Web-Scale through Semantic Deduplication. *arXiv:2303.09540*, 2023.

[26] Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos. D4: Improving LLM Pretraining via Document De-Duplication and Diversification. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[27] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting Influential Data for Targeted Instruction Tuning. In *International Conference on Machine Learning (ICML)*, 2024.

[28] Yuxian Gu, Li Dong, Hongning Wang, Yaru Hao, Qingxiu Dong, Furu Wei, and Minlie Huang. Data Selection via Optimal Control for Language Models. *arXiv:2410.07064*, 2024.

[29] Yuxian Gu, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. MiniPLM: Knowledge Distillation for Pre-Training Language Models. *arXiv:2410.17215*, 2024.

[30] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed Precision Training. In *International Conference on Learning Representations (ICLR)*, 2018.

[31] Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A Study of BFLOAT16 for Deep Learning Training. *arXiv:1905.12322*, 2019.

[32] Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, Ruihang Li, Miaosen Zhang, Chen Li, Jia Ning, Ruizhe Wang, Zheng Zhang, Shuguang Liu, Joe Chau, Han Hu, and Peng Cheng. FP8-LM: Training FP8 Large Language Models. *arXiv:2310.18313*, 2023.

[33] Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. COAT: Compressing Optimizer States and Activation for Memory-Efficient FP8 Training. *arXiv:2410.19313*, 2024.

[34] Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, Siyu Zhu, Steven Shimizu, Shivam Sahni, Haowen Ning, and Yanning Chen. Liger Kernel: Efficient Triton Kernels for LLM Training. *arXiv:2410.10989*, 2024.

[35] Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the Zoo: Chain-of-Action-Thought for GUI Agents. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

[36] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[37] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[38] Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. PathVQA: 30000+ Questions for Medical Visual Question Answering. *arXiv:2003.10286*, 2020.

[39] Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget Captioning: Generating Natural Language Description for Mobile User Interface Elements. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[40] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-Aware Weight Quantization for On-Device LLM Compression and Acceleration. In *Conference on Machine Learning and Systems (MLSys)*, 2024.

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[42] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark-Albert Saroufim, Marcos Yukio, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2024.

[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-Art Natural Language Processing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[44] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2020.

[45] Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[46] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A Diagram is Worth a Dozen Images. In *European Conference on Computer Vision (ECCV)*, 2016.

[47] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

[48] Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. DocVQA: A Dataset for VQA on Document Images. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[49] Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. InfographicVQA. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[50] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts. In *International Conference on Learning Representations (ICLR)*, 2024.

[51] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[52] xAI. Grok-1.5, 2024.

[53] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. SEED-Bench: Benchmarking Multimodal LLMs with Generative Comprehension. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[54] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA Models That Can Read. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[55] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[56] Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. ActivityNet-QA: A Dataset for Understanding Complex Web Videos via Question Answering. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[57] Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Shitao Xiao, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. MLVU: A Comprehensive Benchmark for Multi-Task Long Video Understanding. *arXiv:2406.04264*, 2024.

[58] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. MVBench: A Comprehensive Multi-modal Video Understanding Benchmark. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2024.

[59] Chaoyou Fu, Yuhan Dai, Yondong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Rongrong Ji, and Xing Sun. Video-MME: The First-Ever Comprehensive Evaluation Benchmark of Multi-Modal LLMs in Video Analysis. *arXiv:2405.21075*, 2024.

[60] Yan Shu, Peitian Zhang, Zheng Liu, Minghao Qin, Junjie Zhou, Tiejun Huang, and Bo Zhao. Video-XL: Extra-Long Vision Language Model for Hour-Scale Video Understanding. *arXiv:2409.14485*, 2024.

[61] Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, Zhuang Liu, Hu Xu,

Hyunwoo J Kim, Bilge Soran, Raghuraman Krishnamoorthi, Mohamed Elhoseiny, and Vikas Chandra. LongVU: Spatiotemporal Adaptive Compression for Long Video-Language Understanding. *arXiv:2410.17434*, 2024.

[62] Zuyan Liu, Yuhao Dong, Ziwei Liu, Winston Hu, Jiwen Lu, and Yongming Rao. Oryx MLLM: On-Demand Spatial-Temporal Understanding at Arbitrary Resolution. *arXiv:2409.12961*, 2024.

[63] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

[64] OpenAI. GPT-4V, 2023.

[65] Google. Gemini 1.5: Unlocking Multimodal Understanding Across Millions of Tokens of Context. *arXiv:2403.05530*, 2024.

[66] Google. Gemini: A Family of Highly Capable Multimodal Models. *arXiv:2312.11805*, 2023.

[67] Anthropic. Claude 3, 2024.

[68] Haotian Zhang, Mingfei Gao, Zhe Gan, Philipp Dufter, Nina Wenzel, Forrest Huang, Dhruti Shah, Xianzhi Du, Bowen Zhang, Yanghao Li, Sam Dodge, Keen You, Zhen Yang, Aleksei Timofeev, Mingze Xu, Hong-You Chen, Jean-Philippe Fauconnier, Zhengfeng Lai, Haoxuan You, Zirui Wang, Afshin Dehghan, Peter Grasch, and Yinfei Yang. MM1.5: Methods, Analysis & Insights from Multimodal LLM Fine-Tuning. *arXiv:2409.20566*, 2024.

[69] xAI. Grok-2, 2024.

[70] Oleksii Kuchaiev, Jason Li, Huyen Nguyen, Oleksii Hrinchuk, Ryan Leary, Boris Ginsburg, Samuel Kriman, Stanislav Beliaev, Vitaly Lavrukhin, Jack Cook, Patrice Castonguay, Mariya Popova, Jocelyn Huang, and Jonathan M Cohen. NeMo: A Toolkit for Building AI Applications using Neural Modules. *arXiv:1909.09577*, 2019.

[71] Yunhao Fang, Ligeng Zhu, Yao Lu, Yan Wang, Pavlo Molchanov, Jan Kautz, Jang Hyun Cho, Marco Pavone, Song Han, and Hongxu Yin. VILA^2: VILA Augmented VILA. *arXiv:2407.17453*, 2024.

[72] Min Shi, Fuxiao Liu, Shihao Wang, Shijia Liao, Subhashree Radhakrishnan, De-An Huang, Hongxu Yin, Karan Sapra, Yaser Yacoob, Humphrey Shi, Bryan Catanzaro, Andrew Tao, Jan Kautz, Zhiding Yu, and Guilin Liu. Eagle: Exploring The Design Space for Multimodal LLMs with Mixture of Encoders. *arXiv:2408.15998*, 2024.

[73] Wenliang Dai, Nayeon Lee, Boxin Wang, Zhuoling Yang, Zihan Liu, Jon Barker, Tuomas Rintamaki, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. NVLM: Open Frontier-Class Multimodal LLMs. *arXiv:2409.11402*, 2024.

[74] Meta. Llama 3, 2024.

[75] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, Yen-Sung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Jen Dumas, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and PixMo: Open Weights and Open Data for State-of-the-Art Multimodal Models. *arXiv:2409.17146*, 2024.

[76] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token Merging: Your ViT But Faster. In *International Conference on Learning Representations (ICLR)*, 2023.

[77] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 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)*, 2024.

[78] Yizhe Xiong, Hui Chen, Tianxiang Hao, Zijia Lin, Jungong Han, Yuesong Zhang, Guoxin Wang, Yongjun Bao, and Guiguang Ding. PYRA: Parallel Yielding Re-Activation for Training-Inference Efficient Task Adaptation. In *European Conference on Computer Vision (ECCV)*, 2024.

[79] Joonmyung Choi, Sanghyeok Lee, Jaewon Chu, Minhyuk Choi, and Hyunwoo J Kim. vid-TLDR: Training Free Token Merging for Light-Weight Video Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[80] Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. Chat-UniVi: Unified Visual Representation Empowers Large Language Models with Image and Video Understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[81] Mingze Xu, Mingfei Gao, Zhe Gan, Hong-You Chen, Zhengfeng Lai, Haiming Gang, Kai Kang, and Afshin Dehghan. SlowFast-LLaVA: A Strong Training-Free Baseline for Video Large Language Models. *arXiv:2407.15841*, 2024.

[82] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. SpatialRGPT: Grounded Spatial Reasoning in Vision Language Models. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[83] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[84] Qianlong Du, Chengqing Zong, and Jiajun Zhang. MoDS: Model-Oriented Data Selection for Instruction Tuning. *arXiv:2311.15653*, 2023.

[85] Maxim Fishman, Brian Chmiel, Ron Banner, and Daniel Soudry. Scaling FP8 Training to Trillion-Token LLMs. *arXiv:2409.12517*, 2024.

[86] Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. FP8 Formats for Deep Learning. *arXiv:2209.05433*, 2022.

[87] Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Bhuminand Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact Language Models via Pruning and Knowledge Distillation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[88] Lucio Dery, Steven Kolawole, Jean-François Kagy, Virginia Smith, Graham Neubig, and Ameet Talwalkar. Everybody Prune Now: Structured Pruning of LLMs with only Forward Passes. *arXiv:2402.05406*, 2024.

[89] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-Trained Transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

[90] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2021.

[91] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-Decomposed Low-Rank Adaptation. In *International Conference on Machine Learning (ICML)*, 2024.

[92] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[93] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. In *International Conference on Machine Learning (ICML)*, 2024.

[94] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments. In *European Conference on Computer Vision (ECCV)*, 2020.

[95] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018.