Inference Compute-Optimal Video Vision Language Models

Peiqi Wang[1], ShengYun Peng[2], Xuewen Zhang[3], Hanchao Yu[3], Yibo Yang[3], Lifu Huang[4], Fujun Liu[3], Qifan Wang[3]

[1]MIT, [2]Georgia Tech, [3]Meta, [4]UC Davis

Email: wpq@mit.edu Code: github/vvlm_inference_scaling

Abstract

This work investigates the optimal allocation of inference compute across three key scaling factors in video vision language models: language model size, frame count, and the number of visual tokens per frame. While prior works typically focuses on optimizing model efficiency or improving performance without considering resource constraints, we instead identify optimal model configuration under fixed inference compute budgets. We conduct large-scale training sweeps and careful parametric modeling of task performance to identify the inference compute-optimal frontier. Our experiments reveal how task performance depends on scaling factors and finetuning data size, as well as how changes in data size shift the compute-optimal frontier. These findings translate to practical tips for selecting these scaling factors.

1 Introduction

Video vision-language models (VLMs) have become powerful tools for multimodal video understanding, achieving strong performance tasks such as captioning and question answering (Maaz et al., 2024a; Cheng et al., 2024; Li et al., 2024a). These models are increasingly used in large-scale industry applications, such as recommendation systems and content moderation, where they routinely process millions of videos each day.

In practice, these models are typically pretrained once, often by third parties (Grattafiori et al., 2024), and then finetuned on relatively modest datasets. In contrast, inference occurs continuously and at scale, often incurring orders of magnitude more compute cost than finetuning, as detailed in Appendix A.1. As a result, optimizing for inference efficiency is critical for real-world deployment.

Important design decisions x made before finetuning can significantly impact both the model's inference compute cost $c(x)$ and downstream task performance f. For video VLMs, the most important parameters include: (1) the language model (LM) size x_N, (2) the number of frames processed per video x_T, and (3) the number of tokens per frame x_V used to represent videos. Once set, these scaling factors remain fixed during finetuning and deployment.

Given the substantial computational challenges of deploying video VLMs at scale, our work addresses a key question:

> ***Q**: Given fixed finetuning data of size n and per-example inference compute budget c, how to select scaling factors x for the best model?*

We focus on three specific scaling factors $x \triangleq (x_N, x_T, x_V)$, although our framework can extend to any scaling factors that affects inference compute cost and performance.

Concurrent studies exploring scaling trade-offs in vision-language models have notable gaps (Li et al., 2024b; Du et al., 2024). For instance, Du et al. (2024) examines the trade-off between frame count x_T and tokens per frame x_V but overlooks other important scaling factors like language model size x_N. Moreover, it does not account for the computational cost of processing video frames through vision encoders, leading to overestimated benefits of increasing the number of frames x_T. Our work addresses these limitations.

Existing research on scaling inference-time compute (Li et al., 2024b; Wu et al., 2024b; Snell et al., 2024; Brown et al., 2024) don't investigate the possible interaction between scaling factors x and finetuning data size n. Scaling factors influence how effectively finetuning data is used: (1) larger models are more sample-efficient, achieving lower error with the same amount of data (Henighan et al., 2020), and (2) increasing the number of frames or tokens per frame enriches the dataset by providing more detailed visual information. In contrast, we conduct experiment to explore these interactions and derive insights into how finetuning data size n

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2345–2374
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

shifts the compute-optimal frontier.

In this paper, we tackle the inference compute allocation problem outlined above—determining the optimal scaling factors under a fixed inference compute budget—and propose a simple, effective recipe. Inspired by Hoffmann et al. (2022); Alabdulmohsin et al. (2023), we conduct efficient training sweeps to explore multiple scaling factors (e.g., x_N, x_T, x_V, n) while minimizing the number of training runs. To address the lack of consensus on appropriate parametric models for downstream task performance (Owen, 2024; Li et al., 2024b; Gadre et al., 2024), we perform model selection to identify suitable formulations and thoroughly validate the fitted model, accounting for challenges such as small sample sizes and the difficulty of predicting performance metrics (vs. next-token prediction loss). Finally, we determine the optimal scaling factors by solving a constrained discrete optimization problem numerically, providing a concrete answer to the posed question.

Our training sweeps reveals that: (1) scaling factors x and finetuning data size n exhibit diminishing returns on performance, (2) jointly scaling (x_N, x_T, x_V) is crucial for optimal performance, and (3) the utility of x varies significantly across tasks, indicating that no universal inference compute allocation strategy exists—allocation must be task-specific. We demonstrate that a power-law additive functional form with an interaction term effectively models average task performance. By solving a constrained optimization problem using this parametric model, we identify an inference compute-optimal frontier that provides guidance on scaling x optimally. Moreover, the finetuning data size n shapes this frontier; for instance, the trend suggests increasing x_T and x_V while decreasing x_N as more data becomes available.

In summary, our key contributions are:

- We tackle the important problem of allocating inference compute across scaling factors in video VLMs. Our approach combines efficient training sweeps, parametric modeling of task performance, and constrained discrete optimization to identify optimal trade-offs.
- We conduct large-scale training sweeps (e.g., ∼100k A100 hours) and derive qualitative observations of how video task performance varies with scaling factors and finetuning data size.
- We provide actionable insights into scaling inference compute for video VLMs, e.g., optimal allocation of factors such as LM size, frame count, and tokens per frame, as well as the impact of data size on the compute-optimal frontier.

Aspect	Training	Inference
Data	Abundant	Limited
Cost (FLOPs) †	$6x_N n_{\text{pt}}$	$2x_N x_T x_V$
Problem	$\min_{x_N, n_{\text{pt}}} f(x_N, n_{\text{pt}})$	$\min_x f(x, n)$
	$c(x_N, n_{\text{pt}}) \leq c_{\text{pt}}$	$c(x) \leq c$

Table 1: Comparison of optimizing training compute for LLMs versus inference compute for video VLMs. Notations: x_N (LM parameters), x_T (frames), x_V (tokens per frame), n_{pt} and n (training and finetuning data sizes), f (error), c_{pt} (training compute budget), and c (per-example inference compute budget). †denotes compute cost estimates for LMs only.

2 Relationship to Scaling Law Studies

In contrast to previous studies (Kaplan et al., 2020; Hoffmann et al., 2022) on training compute-optimal model that balances model size x_N and pretraining data size n_{pt}, we aim to optimize scaling factors for video VLMs such that the model is inference compute-optimal. While our motivation aligns with Sardana et al. (2024) in considering deployment costs, our focus is on finetuning rather than pretraining LMs. A notable distinction of our setup is that we finetune with all available data because finetuning datasets are limited in quantity, their compute cost is negligible, and more data always provide better performance without incurring inference compute cost. Table 1 contrasts the optimization of training compute for LMs and inference compute for video VLMs.

Motivated by existing finetuning scaling studies, our work investigates how finetuning data size n affects VLM performance. Unlike Hernandez et al. (2021), which focuses on pretraining-to-finetuning knowledge transfer, we explore how finetuning data size interacts with other scaling factors to influence inference efficiency and performance. While Zhang et al. (2023) examines similar interactions in the context of finetuning LMs for translation and summarization, we target video benchmarks and focus on allocating inference compute.

Our work can be viewed as scaling video VLMs' inference-time compute, e.g., by increasing the size of video representations. While previous research has explored scaling retrieval augmented generation (Yue et al., 2024) or test-time search strate-

gies (e.g., best-of-N or beam search) (Wu et al., 2024b; Snell et al., 2024; Brown et al., 2024), these approaches adjust inference compute for a fixed model. In contrast, we finetune models for any given scaling factors and use the same setup during inference.

3 Inference Computational Optimality

Notations Let $x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_K \triangleq \mathcal{X}$ denote a set of K factors that influence both inference compute and video VLM performance, where each $\mathcal{X}_k \subset \mathbb{N}$ is a subset of the natural numbers. In this work, we specialize x to (x_N, x_T, x_V), where x_N denotes the language model (LM) size, x_T the number of video frames, and x_V the number of visual tokens per frame, with $\mathcal{X}_N$ restricted to pretrained LM sizes and $\mathcal{X}_V$ to perfect squares. Model performance is quantified using a metric $f : \mathcal{X} \times \mathbb{N} \to \mathbb{R}$, where $f(x, n)$ represents the downstream task error (lower is better) from finetuning a model with scaling factors x on a dataset of size n. While f represents error, we refer to it interchangeably as performance throughout this paper.

Video VLM Our analysis of video VLM scaling is based on LLaVA-like architectures (Liu et al., 2023), which consist of two main components:

- *vision model* with x_M parameters (e.g., CLIP Radford et al., 2021) that processes x_T video frames independently. For each frame, it generates a grid of x_W visual features that are then projected and resampled into x_V visual tokens
- *language model* with x_N parameters (e.g., Llama-3) that consumes the sequence of visual token representations of length $x_T x_V$ to perform video understanding or reasoning tasks.

Inference Compute Cost We measure computational cost using floating-point operations (FLOPs), focusing on the inference cost of the vision and language model components. We assume each example consists of a single video, where the length of visual tokens dominate that of the input instructions or output generations. Thus, we disregard the compute cost of the latter. Using the standard approximation of $2x_N$ FLOPs per token for a transformer model with x_N parameters (Kaplan et al., 2020), the per-example inference compute cost is

$$c(x) = 2x_T \left(x_M x_W + x_N x_V\right). \tag{1}$$

For a video VLM using SoViT-400m/14 (Alabdulmohsin et al., 2023) as the vision model, this becomes $c(x) = 2x_T(0.43e9 \cdot 768 + x_N x_V)$.

Prior work often overlooks the vision model's compute cost, focusing solely on the language model (Li et al., 2024b; Du et al., 2024). In Appendix A.3, we demonstrate that the vision model's relative compute cost becomes significant as x_N and x_V decrease.

Parametric Model of Performance We model task error $f(x, n)$ using `add-interact` (Alabdulmohsin et al., 2023), a parametric function that defines an additive power-law relationship with interaction terms. This formulation is well-suited for scenarios where both inference compute $c(x)$ and data size n are constrained, as is often the case when deploying video VLMs. Moreover, it enables an exploration of the interactions between x and n.

For a specific scaling factor $x_k \in \mathcal{X}_k$ (e.g., the number of frames x_T) and finetuning data size n, we model the error as

$$f_k(x_k, n) = \alpha_k x_k^{-a_k} + \left(\beta_k x_k^{b_k} + \xi_k\right) n^{-d} + \varepsilon_k, \tag{2}$$

where $\alpha_k, \beta_k, \xi_k, \varepsilon_k \geq 0$ are coefficient parameters, and $a_k, b_k, c \in \mathbb{R}$ are exponent parameters to be estimated from empirical training results. The terms in this formulation are interpreted as follows:

- The coefficients α_k, β_k, ξ_k represent error components that can be reduced by increasing x_k or n, while ε_k accounts for irreducible error.
- The exponent a_k describes how the error scales with x_k in the data-unbounded regime, where $f_k \propto x_k^{-a_k}$ as $n \to \infty$.
- The data scaling exponent d quantifies how the error decreases with increasing data size n for fixed inference compute, i.e., $f_k \propto n^{-d}$
- The exponent b_k determines how x_k affects the reducible error $\beta_k x_k^{b_k} + \xi_k$ and, thereby the impact of increasing n. For $b_k > 0$, larger x_k implies: (1) requiring more finetuning data to achieve the same error and (2) faster error reduction per additional example. See Section D.2 for details.

For multiple factors x, we model the error as

$$f(x, n) = \sum_k \alpha_k x_k^{-a_k} + \sum_k \beta_k x_k^{b_k} n^{-d} + \xi n^{-d} + \varepsilon \tag{3}$$

where $\alpha_k, \beta_k, \xi, \varepsilon \geq 0$ and $a_k, b_k, d \in \mathbb{R}$. Consistent with prior work (Alabdulmohsin et al., 2023; Bahri et al., 2024), we assume d is independent of the scaling factors.

While we primarily use `add-interact` defined in Equation 2, alternative formulations exist. These include its simplifications (e.g., $\texttt{add-interact}_s$), as well as commonly used additive power-law functions (`add`) and its multiplicative variants (`mult`). Section D.1 provides expressions of each parametric function we study.

Allocating Inference Compute We aim to optimize the allocation of inference compute across scaling factors (x_N, x_T, x_V) to maximize model performance under a given per-example inference compute budget c and finetuning data size n. This can be formulated as follows:

$$x^*(c; n) = \underset{x \in \mathcal{X}:\ c(x) \leq c}{\arg\min}\ f(x, n) \tag{4}$$

where x^* represents the optimal scaling factors. For simplicity, we omit c and n when their context is implicit. $x^*(c; n)$ is also referred to as the "(inference) compute-optimal frontier".

Unlike training compute optimization, inference compute optimization does not treat n as an optimization variable, as finetuning data size does not affect inference compute cost. However, for certain parametric functions (e.g., `add-interact`), n can still influence the solution x^*. For others (e.g., `add`), the solution $x^*(c)$ is independent of n. Table 7 summarizes which parametric functions exhibit this dependency.

For simple additive power-law parametric forms (e.g., `add` in Table 7) and when inference compute is limited to LM compute costs (i.e., $c(x) = x_N x_T x_V$), the problem admits an analytic solution if $\mathcal{X}$ is continuous. However, when vision model compute costs are included (e.g., $c(x)$ as defined in Equation 1), deriving x^* analytically as a function of c and n becomes intractable. This is further complicated by the discrete nature of $\mathcal{X}$ (e.g., pretrained LMs are only available in a fixed set of sizes). To address these challenges, we compute x^* using a brute-force search over all combinations in $\mathcal{X}$. In practice, this is computationally feasible in our setup due to the small size of $\mathcal{X}$: $|\mathcal{X}_N| \leq 5$, $|\mathcal{X}_T| \leq 128$, and $|\mathcal{X}_V| \leq 28$. Because the optimization problem is discrete, the optimal scaling factors x^* often lie within the feasible region rather than on the boundary.

4 Implementation Details

Video Instruction Dataset We use a comprehensive video instruction tuning dataset of ~ 2.2 million examples to investigate the impact of finetuning data size n on model performance. This dataset is compiled from a diverse range of video sources, e.g., LLaVA-Video-178K (Zhang et al., 2024b), and includes various types of instructions, e.g., chat, question-answering, and captioning. Appendix B.1 lists dataset composition and provides some explanations on how we assemble the dataset.

Model and Training We full-parameter finetune LLaVA-like architectures (Liu et al., 2023), which integrate a vision model, a projector, and a language model. The SoViT-400m/14 vision model (Alabdulmohsin et al., 2023) is chosen for its performance. The projector is a two-layer MLP that converts visual features into tokens, which are then processed by the Llama-3.2 series of LMs ($x_N \in \{1\text{B}, 3\text{B}, 8\text{B}\}$) (Grattafiori et al., 2024).

The finetuning process involves two stages: pretraining the projector on the LCS dataset (Liu et al., 2023) and subsequently finetuning the entire model on video instruction datasets. We use the codebase and default hyperparameters from Li et al. (2024a). Appendix B.2 provides additional details.

Video frames are uniformly sampled at a minimum rate of 1 frame per second (fps), repeating frames if the video duration is too short to meet this rate. To downsample a 2D grid of visual features x_W into a smaller grid x_V, we apply bilinear interpolation (Li et al., 2024a), implying values of x_V is constrained to be perfect squares.

Evaluation Tasks and Metrics To thoroughly assess the capabilities of video VLMs, we use a diverse set of eight video tasks, including various question-answering (QA) types, e.g., captioning, open-ended, and multiple-choice questions. Evaluations include Video Detailed Caption (VDC Zhang et al., 2024a) for detailed video descriptions, ActivityNet-QA (AQA Yu et al., 2019) for action-related QAs, VCGBench (VCG Maaz et al., 2024a) for video chat capabilities, LongVideoBench (LVB Wu et al., 2024a) for long video understanding, and PerceptionTest (PT Patraucean et al., 2023) for fine-grained perception. Additionally, MVBench (MV Li et al., 2024c), Video-MME (VMME Fu et al., 2024), and Next-QA (NQA Xiao et al., 2021) provide broad evaluations across diverse video tasks and domains. We measure performance using accuracy ("Acc") for multiple-choice QA tasks and gpt-4o-mini's ratings ("Score") for open-ended QA and captioning tasks. All metrics are standardized to a $(0, 100)$ scale to facilitate a balanced average of

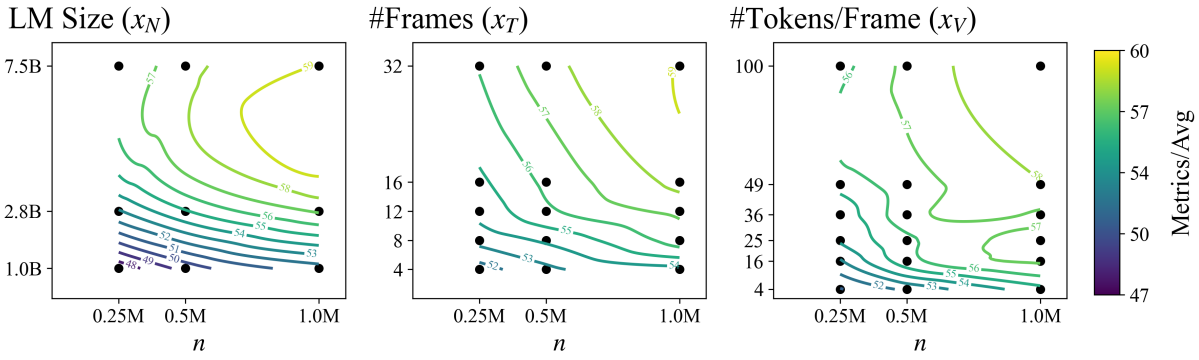

Figure 1: **IsoPerformance Contours.** Contours show average task performance as a function of a scaling factor (e.g., x_N, x_T, or x_V) and finetuning data size n, derived from the *star sweep*. As detailed in Section 5.1, we construct the star sweep by starting with a inference compute-intensive "center" $x^\star = (7.5\text{B}, 32, 196)$, varying one factor at a time while keeping the others fixed, and finetuning on different data sizes. For instance, in the left subfigure, each dot represents a x_N-parameter LM finetuned on n examples, with $x_T = 32$ and $x_V = 196$ fixed. **Performance improves as scaling factors x and n increase, albeit at a diminishing rate.** Irregularities in the contour lines, particularly near boundaries, arise from interpolation artifacts (via `matplotlib.pyplot.contourf`) and variability in benchmark scores across fine-tuning runs.

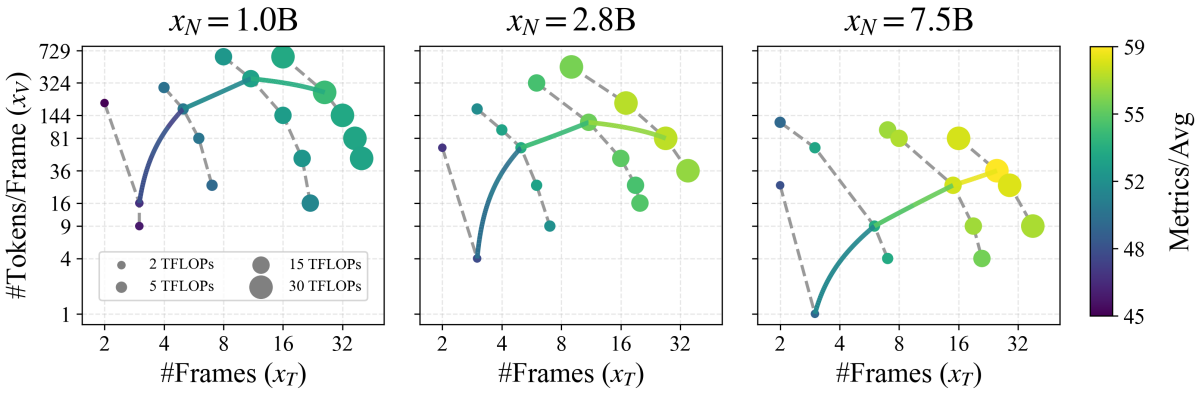

Figure 2: **IsoFLOP Curves and Compute-Optimal Frontier.** IsoFLOP curves (dotted lines) show task performance (color-coded) for models with fixed inference compute cost $c(x)$ across four TFLOP budgets: 2, 5, 15, and 30. The compute-optimal frontier (solid line) connects models with the best average task performance. Both are derived from the *isoFLOP sweep* described in Section 5.1. The compute-optimal frontier reveals that optimal performance requires scaling both x_T and x_V together. Moreover, at 30 TFLOPs, a model with $x_N = 7.5\text{B}$ outperforms one with $x_N = 1\text{B}$, as smaller LMs cannot effectively make use of higher compute budgets (e.g., increasing from 15 to 30 TFLOPs yields minimal gain), highlighting the bottleneck imposed by LM size. **These findings underscore the importance of jointly scaling x_N, x_T, and x_V to maximize performance.**

evaluation metrics ("Metrics/Avg"). Appendix B.3 provides additional details on evaluation.

5 Experiments

5.1 Training Sweeps

To study the behavior of video VLM w.r.t. scaling factors x and finetuning data size n, we perform two types of training sweeps:

- *Star sweep*: We start with an inference compute-intensive "star center" $(7.5\text{B}, 32, 196)$, vary one factor at a time while keeping the others constant, and finetune the model on three different data sizes n (in millions): 0.25, 0.5, and 1. The star sweep avoids an expensive grid search over (x_N, x_T, x_V, n) and instead focuses on characterizing how each scaling factor scales with n, leading to a more accurate estimate of scaling exponents (as shown in Alabdulmohsin et al., 2023). Appendix C provides more details on how our implementation differs slightly from the original.
- *IsoFLOP sweep*: We adjust the scaling factors $x = (x_N, x_T, x_V)$ to maintain a fixed inference compute cost $c(x)$ across four target TFLOPs: 2,

Form	Expressions for $f(x, n)$	Star+IsoFLOP CV(5-fold)			Star → IsoFLOP		
		MSE ↓	$E_{\%}$ ↓	R^2 ↑	MSE ↓	$E_{\%}$ ↓	R^2 ↑
`mult`	$\alpha(\prod_{k=1}^{K} x_k^{-a_k})n^{-d} + \epsilon$	1.21	1.62	0.88	6.73	3.55	0.45
`add`	$\sum_k \alpha_k x_k^{-a_k} + \xi n^{-d} + \epsilon$	0.56	1.11	0.94	2.04	2.15	0.83
`add-interact`$_s$	$\sum_k \alpha_k x_k^{-a_k} + \sum_k \beta_k x_k^{b_k} n^{-d} + \epsilon$	0.24	0.8	0.97	0.94	1.32	0.92
`add-interact`	$\sum_k \alpha_k x_k^{-a_k} + \sum_k \beta_k x_k^{b_k} n^{-d} + \xi n^{-d} + \epsilon$	0.2	0.77	0.98	0.95	1.33	0.92

Table 2: **Comparison of Parametric Models of Task Performance.** Evaluation of different parametric functions for modeling the average task performance under two setups: (1) *in-distribution* performance using 5-fold cross-validation (CV) on combined star and isoFLOP data, and (2) *extrapolation* on isoFLOP data after training on star data. Metrics include mean squared error (MSE), average relative error ($E_{\%}$), and coefficient of determination (R^2). **The `add-interact` model and its simpler variant `add-interact`$_s$ incorporate additive power laws with interaction terms and achieve the best performance.** These functional forms outperform simpler additive (`add`) and multiplicative (`mult`) power law models, emphasizing the importance of appropriately modeling the interactions between scaling factors x and finetuning data size n.

5, 15, and 30, and finetune the model on $n = 2$ million examples. The isoFLOP sweep is designed to identify the optimal scaling factors for a given inference FLOP budget and to evaluate the effects of jointly scaling multiple factors. Additionally, it serves as a held-out set for model validation and selection.

Appendix C provides a detailed description of the experimental setup and execution of these sweeps.

> Scaling factors x and finetuning data size n yields diminishing return on performance.

Figure 1 illustrates that average task performance improves with increasing scaling factors x and data size n. However, the rate of improvement (1) diminishes as x and n grows larger (2) varies across x, n. These trends suggest that task performance can be effectively modeled using power-law relationships with factor-specific exponents.

> Jointly scale (x_N, x_T, x_V) is crucial.

Figure 2 presents the isoFLOP curves, which represent models with varying x and fixed inference compute costs, and the compute-optimal frontier that connects the best-performing models across different inference FLOPs budget. The compute-optimal frontier underscore the importance of increasing x_N, x_T, and x_V in tandem to maximize performance. For instance, doubling the inference compute from 15 to 30 TFLOPs (by varying x_T and x_V) yields marginal performance gains for a smaller x_N = 1B LM, while the same increase in inference compute provides significant improvement for a larger x_N = 7.5B LM, highlighting the bottleneck imposed by LM size. Similar bottleneck effects are observed for x_T and x_V respectively. The per-task compute-optimal frontier in Figure 9 reaffirms the importance of jointly scaling x for each task. However, this trend is not perfectly consistent due to limited robustness of the finetuning and evaluation process, as well as the coarse granularity of the isoFLOP sweep conducted under computational constraints.

> Utility of (x_N, x_T, x_V) vary by task → so does optimal allocation of inference compute.

Figure 8 demonstrates that the utility of scaling factors x differs significantly across downstream tasks. For example, increasing the number of frames x_T yields greater benefits than scaling n for long video understanding tasks (e.g., LongVideoBench). In contrast, x_T offers minimal gains compared to n for fine-grained perception tasks (e.g., PerceptionTest). These variations in utility shape the compute-optimal frontier, as shown in Figure 8. For example, in PerceptionTest, the frontier prioritizes x_V (with higher marginal benefits) over x_T (with lower marginal benefits). This highlights that optimal inference compute allocation strategies should adapt to the task of interest, focusing on factors that yield the highest marginal returns on performance.

5.2 Modeling & Fitting of Task Performance

Figure 2 illustrates that the training runs sparsely cover the space of (x, n), making accurate interpolation or extrapolation of the compute-optimal frontier challenging. To address this, we model task performance using simple parametric functions and fit them to the training results from the star and isoFLOP sweeps. This modeling step is crucial for solving the optimization problem in Equation 4.

We fit the parametric model by minimizing the mean squared error loss between the predicted and

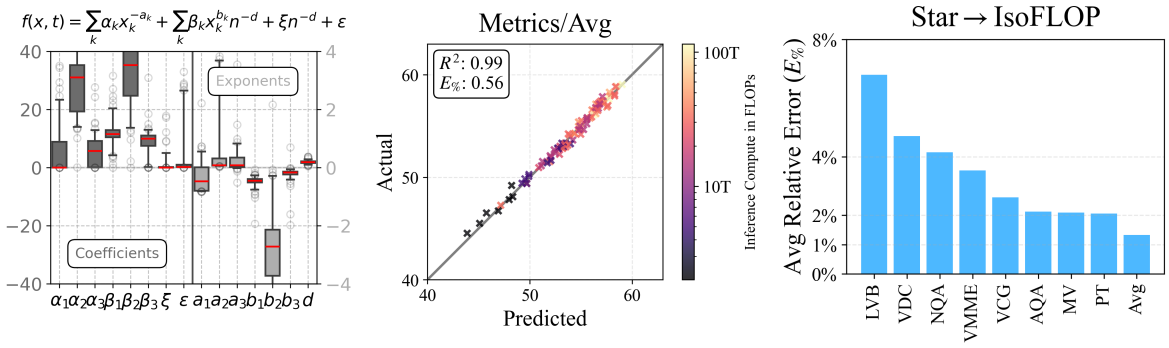

Figure 3: **Parametric Fitting of Task Performance.** (Left) Box plot of bootstrap-resampled parameter estimates (100 resamples) for the `add-interact` model (defined in Equation 3) highlights the challenge with fitting a model with just ∼100 examples. (Center) Scatter plot comparing the predicted average task performance ("Metrics/Avg") with the actual performance for each run in the star and isoFLOP sweeps. `add-interact` achieves a strong fit to data. (Right) Bar plot illustrating `add-interact`'s extrapolation performance on isoFLOP data after being trained on star data across various video tasks. While it achieves good performance for Metrics/Avg (Avg), it struggles to extrapolate effectively for tasks such as LongVideoBench (LVB) and Next-QA (NQA). **Overall, the bagged `add-interact` model provides a reasonable fit for predicting downstream task performance.**

observed performance metrics in log space. Appendix D.3 provides a detailed ablation study of the parameter estimation procedure, demonstrating the importance of a carefully designed setup for achieving accurate task performance modeling.

We assess the quality of the fit under two scenarios: (1) *in-distribution* performance using 5-fold cross-validation (CV) on the star and isoFLOP data, and (2) *extrapolation* performance on the isoFLOP data after estimating parameters on the star data. We report: mean squared error or MSE, average relative error $E_{\%} = |\hat{f} - f|/f$ in percentage, and the coefficient of determination R^2.

> Bagged `add-interact` is a good fit to predict the performance of (some) downstream tasks.

Table 2 compares several parametric functions (details in Appendix D.1) for modeling average task performance. Functions that include an interaction term between each scaling factor (e.g., x_N, x_T, or x_V) and finetuning data size n (e.g., `add-interact` and `add-interact`$_s$) outperforms those without interaction terms (e.g., `add`) or with incorrect interaction model (e.g., `mult`).

Left subfigure in Figure 3 shows a box plot of bootstrap-resampled parameter estimates for the `add-interact` model. Some parameters exhibit high variability, reflecting the challenge of robustly fitting the model with ∼100 examples. To alleviate this issue, we use bootstrap aggregation (bagging) to improve stability and accuracy. Ablation studies in Appendix D.4 demonstrate the effectiveness of using median aggregation over 100 base models.

Based on these results, we adopt `add-interact` with bootstrap aggregation for all subsequent analyses, as it's more stable and provides the best fit.

The middle subfigure in Figure 3 demonstrates that `add-interact` achieves an excellent fit to the star & isoFLOP data, covering roughly 2 orders of magnitude in inference compute. In contrast, the right subfigure in Figure 3 reveals considerable variability in the model's extrapolation performance across tasks. While average task performance is easier to model, `add-interact` struggles with tasks like LongVideoBench (LVB) and Next-QA (NQA). For these tasks, $E_{\%} \geq 5\%$ corresponds to an average deviation exceeding 3 points (on a 0-100 scale), representing substantial error.

5.3 Optimal Allocation of Inference Compute

In this section, we address the question posed in Section 1. Using the `add-interact` model of task performance (described in Section 3) that we fit in Section 5.2, we compute the inference compute-optimal frontier $x^*(c; n)$ for any fixed c, n by solving Equation 4. This is done via a brute-force search over $x \in \mathcal{X}$ to minimize the error $f(x, n)$.

> The frontier $x^*(c)$ scales jointly at varying rates and is not monotonic non-decreasing.

Figure 4 illustrates the predicted compute-optimal frontier $x^*(c; n)$ for key scaling factors x of video VLMs, across different data sizes n. This frontier reflects the joint scaling of (x_N, x_T, x_V) at different rates, consistent with empirical trends observed from training sweeps in Figure 2. The domain of x

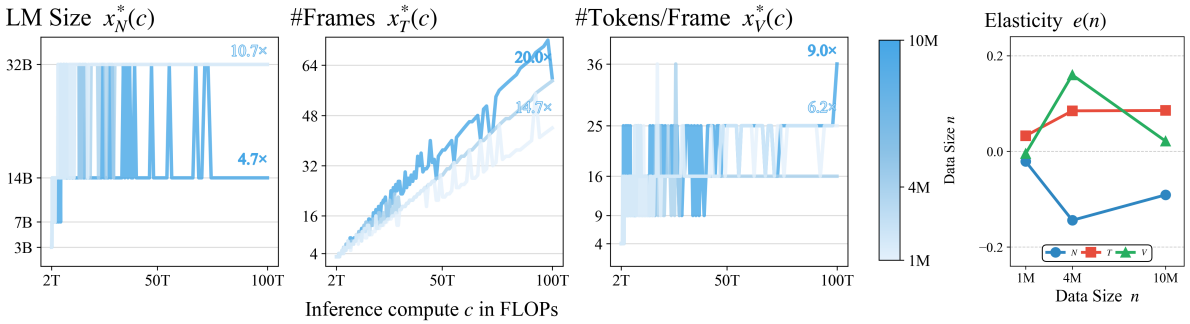

Figure 4: **Predicted Compute-Optimal Frontier for Video VLMs.** The left three subplots show the predicted inference compute-optimal frontier $x^*(c; n)$ for key scaling factors x of video VLMs, across varying fine-tuning data sizes n (shades of blue). The blue text indicates the increase in x^* as inference compute grows from 2T to 100T FLOPs. Task performance $f(x, n)$ is modeled using the bagged `add-interact` model, which identifies **an efficiency frontier that requires joint scaling of** (x_N, x_T, x_V) **at varying rates and is non-monotonic due to the discrete domain** $\mathcal{X}$ **of** x. The rightmost subplot depicts the elasticity (defined in Equation 10) for each factor $k \in \{N, T, V\}$, quantifying the sensitivity of x_k^* to changes in n. For instance, as n increases, the frontier $x_T^*(c)$ shifts upward (in darker blue), corresponding to a positive $e_T(n)$ (red curve) in the elasticity plot.

is a discrete set $\mathcal{X}$ with coarse increments (e.g., for x_N and x_V), which shapes the compute-optimal frontier in two key ways: (1) it appears staggered, and (2) it is not monotonically non-decreasing, even though the predicted performance is.

Figure 10 shows that the predicted optimal allocation of inference compute varies notably across tasks, consistent with trends observed in Figure 9.

> Finetuning data size n influence the shape of the predicted compute-optimal frontier $x^*(c)$.

We quantify how changes in finetuning data size n affect the compute-optimal scaling factor $x_k^*(c; n)$ using elasticity, defined as:

$$e_k(c, n) = \frac{\partial x_k^*(c; n)}{\partial n} \cdot \frac{n}{x_k^*(c; n)}. \tag{5}$$

Commonly used in economics, elasticity expresses sensitivity in percentage terms, enabling intuitive interpretation and comparison across variables of different scales. For instance, $e_T = 0.1$ indicates that a 1% increase in n results in a 0.1% increase in x_T^*. We use forward differences to compute elasticity and aggregate across compute budgets and data sizes to capture overall trends. Appendix E provides detailed explanations of its definition, numerical approximations, and ablation studies.

Figure 4 illustrates the effect of n on the predicted compute-optimal frontier $x^*(c; n)$. As n increases, x_N^* shifts downward (negative elasticity), while x_T^* and x_V^* shifts upward (positive elasticity). This trend is consistent across video tasks, as shown in Figure 5, though with task-specific variations. The average trend suggests decreasing x_N and increasing x_T, x_V as data size n grows.

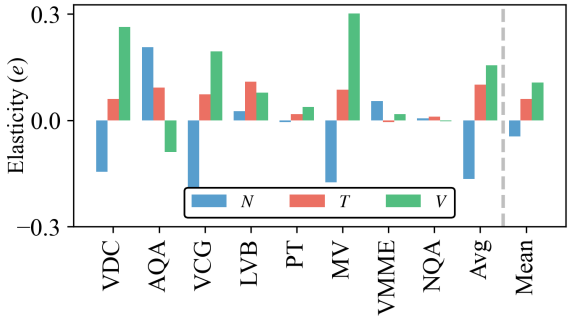

Figure 5: **Elasticity Across Tasks.** Bar plot showing the elasticity (defined in Equation 11) for scaling factors $k \in \{N, T, V\}$ across video tasks. This measures the sensitivity of optimal scaling factors x_k^* to changes in data size n. **While there is significant task-specific variation, the general trend suggests decreasing** x_N **and increasing** x_T, x_V **as data size** n **grows.**

6 Discussions

Joint Scaling in Video VLMs Efficient use of compute through joint scaling of factors is a recurring theme in scaling law studies. EfficientNet (Tan and Le, 2019) demonstrated that scaling a single factor (e.g., resolution) while keeping others (e.g., width and depth) fixed is suboptimal, as bottlenecked factors limit performance gains. Instead, it introduced compound scaling, which jointly scales multiple architectural factors to achieve better results. This principle has been applied broadly, including to ViTs (Alabdulmohsin et al., 2023) and

LM pretraining (Hoffmann et al., 2022). We empirically confirm that this principle extends to video VLMs. Specifically, jointly scaling LM size, frame count x_T, and visual tokens per frame x_V improves performance, as shown in Figure 2. Table 2 further supports this, where the best-performing model, `add-interact`, identifies a compute-optimal frontier that scales all three factors together. These findings provide stronger evidence than concurrent studies, e.g., Du et al. (2024), which assumes a parametric model without directly observing trends from training sweeps.

Practical Implications of Joint Scaling Despite the benefits of joint scaling, many existing works, e.g., MM1 (McKinzie et al., 2024), Idefics3 (Laurençon et al., 2024), and LLaVA-OneVision (Li et al., 2024a), rely on ablation studies that vary one factor at a time to identify good training setups. While this reduces the number of training runs, our findings suggest that, after establishing a reasonable baseline, compute resources should instead be allocated to exploring how key factors can be scaled jointly for better performance. Furthermore, many video VLMs, e.g., LLaVA-Video (Zhang et al., 2024b) and Qwen-VL (Wang et al., 2024), fix the size of visual representation of videos while varying LM size. Our results show this approach is suboptimal for inference compute, highlighting significant opportunities for improvement.

Varied Scaling Rates Our experiments reveal that frame count x_T and tokens per frame x_V have distinct effects on task performance. As shown in Figure 4, scaling x_T yields larger performance gains than scaling x_V, suggesting that improving the vision model's efficiency is more impactful than reducing the LM's cost of processing tokens for each frame. By enabling more frames to be processed, x_T scaling provides greater overall improvements. This observation also has implications for comparing models. For instance, AuroraCap (Chai et al., 2024) compares its model to baselines with an equal number of visual tokens $x_T x_V$ per video but varies x_V while fixing x_T for its method and does the reverse for baselines. This inconsistency inflates the perceived performance gains of its approach.

Effect of Finetuning Data Size Our findings provide evidence on how finetuning data size affects the compute-optimal frontier. Figure 4 and 5 suggest that as more data becomes available, it is optimal to allocate less compute to LM size x_N and more to video visual representations x_T and x_V. We hypothesize this is because detailed visual representations are more complex to learn and require more data to achieve comparable performance. However, as data size grows, the richer information in each visual example contributes more to overall performance, outweighing the added learning complexity. This trend mirrors findings in structured vision tasks such as table recognition (Peng et al., 2023, 2024). Further research is needed to validate this hypothesis.

7 Conclusions

In this work, we tackle the problem of optimally allocating inference compute across scaling factors that impact both compute cost and downstream task performance in video VLMs. Our approach involves: (1) conducting training sweeps to gather performance data, (2) fitting parametric models to predict task performance, and (3) solving constrained optimization problems to identify optimal trade-offs. Our work provides deeper insights into task performance trends for video VLMs, highlights effective strategies for investigating and understanding scaling behaviors, and offers practical guidance for deploying these models at scale.

Limitations

Inference Compute Estimation To accurately measure inference compute for video VLMs, we account for the compute cost of the vision model. However, theoretical FLOPs alone are insufficient for real-world deployments due to several overlooked factors. (1) Hardware utilization during inference often falls short of theoretical peaks, varying across implementations and hardwares. (2) Inference compute is split into two stages: prefilling, which is typically compute-bound, and decoding, which is often constrained by memory bandwidth. Our current analysis only considers the compute cost of the prefilling stage. (3) Modern efficient inference techniques, e.g., quantization (Dettmers et al., 2022) and speculative decoding (Leviathan et al., 2023), add further complexity to accurately estimating inference compute.

Components Not Accounted For While we aim to include key scaling factors affecting both inference compute and task performance, some components were omitted. For instance, we did not explore scaling the size of the vision model, as most experiments were conducted using SoViT, which lacks a series of models with varying sizes. Prior work has shown that jointly scaling language and vision models is important (Chen et al., 2024). Additionally, we hold video instruction dataset, the language model family (e.g., Llama-3.2), and downsampling methods (e.g., bilinear interpolation) fixed. While exploring the impact of these factors could provide valuable insights, we prioritized a simple, broadly applicable setup to make use the limited computational resources.

Parametric Model Validation The accuracy of our parametric models of performance is affected by experimental design choices, which may introduce implicit biases despite careful attention to implementations, training sweep design, and model fitting. For instance, we selected the Llama-3.2 model family early in the project, which lacks pretrained LM sizes between 8B and 70B. As a result, our experiments include only three LM sizes (1B, 3B, 8B), potentially limiting our ability to fully capture the relationship between LM size and task performance. While we validated the models' extrapolation performance using the isoFLOP sweep (which includes runs with lower inference compute than the star sweep), it remains uncertain how well the predicted compute-optimal frontier generalizes to significantly higher inference compute, such as 10x the current FLOPs. Addressing these issues is challenging due to the limited computational resources allocated to this project.

References

Ibrahim Alabdulmohsin, Xiaohua Zhai, Alexander Kolesnikov, and Lucas Beyer. 2023. Getting ViT in Shape: Scaling Laws for Compute-Optimal Model Design. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. 2024. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*.

Max Bain, Arsha Nagrani, Gul Varol, and Andrew Zisserman. 2021. Frozen in Time: A Joint Video and Image Encoder for End-to-End Retrieval. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. *Preprint*, arXiv:2407.21787.

Wenhao Chai, Enxin Song, Yilun Du, Chenlin Meng, Vashisht Madhavan, Omer Bar-Tal, Jenq-Neng Hwang, Saining Xie, and Christopher D. Manning. 2024. AuroraCap: Efficient, Performant Video Detailed Captioning and a New Benchmark. In *The Thirteenth International Conference on Learning Representations*.

Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, Siamak Shakeri, Mostafa Dehghani, Daniel Salz, Mario Lucic, Michael Tschannen, Arsha Nagrani, Hexiang Hu, Mandar Joshi, Bo Pang, Ceslee Montgomery, Paulina Pietrzyk, Marvin Ritter, Aj Piergiovanni, Matthias Minderer, Filip Pavetic, Austin Waters, Gang Li, Ibrahim Alabdulmohsin, Lucas Beyer, Julien Amelot, Kenton Lee, Andreas Peter Steiner, Yang Li, Daniel Keysers, Anurag Arnab, Yuanzhong Xu, Keran Rong, Alexander Kolesnikov, Mojtaba Seyedhosseini, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. 2024. On Scaling Up a Multilingual Vision and Language Model. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, and Lidong Bing. 2024. VideoLLaMA 2: Advancing Spatial-Temporal Modeling and Audio Understanding in Video-LLMs. *Preprint*, arXiv:2406.07476.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale. In *Advances in Neural Information Processing Systems*.

Yifan Du, Yuqi Huo, Kun Zhou, Zijia Zhao, Haoyu Lu, Han Huang, Xin Zhao, Bingning Wang, Weipeng Chen, and Ji-Rong Wen. 2024. Exploring the Design Space of Visual Context Representation in Video MLLMs. In *The Thirteenth International Conference on Learning Representations*.

Chaoyou Fu, Yuhan Dai, Yongdong 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. 2024. Video-MME: The First-Ever Comprehensive Evaluation Benchmark of Multi-modal LLMs in Video Analysis. *Preprint*, arXiv:2405.21075.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, Rui Xin, Marianna Nezhurina, Igor Vasiljevic, Jenia Jitsev, Luca Soldaini, Alexandros G. Dimakis, Gabriel Ilharco, Pang Wei Koh, Shuran Song, Thomas Kollar, Yair Carmon, Achal Dave, Reinhard Heckel, Niklas Muennighoff, and Ludwig Schmidt. 2024. Language models scale reliably with over-training and on downstream tasks. *Preprint*, arXiv:2403.08540.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. 2017. The "Something Something" Video Database for Learning and Evaluating Visual Common Sense. In *2017 IEEE International Conference on Computer Vision (ICCV)*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and et al. 2024. The Llama 3 Herd of Models. *Preprint*, arXiv:2407.21783.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. 2020. Scaling Laws for Autoregressive Generative Modeling. *Preprint*, arXiv:2010.14701.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. 2021. Scaling Laws for Transfer. *Preprint*, arXiv:2102.01293.

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, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. 2022. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

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. *Preprint*, arXiv:2001.08361.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. 2017. The Kinetics Human Action Video Dataset. *Preprint*, arXiv:1705.06950.

Li Kunchang, He Yinan, Wang Yi, Li Yizhuo, Wang Wenhai, Luo Ping, Wang Yali, Wang Limin, and Qiao Yu. 2025. VideoChat: Chat-Centric Video Understanding. *SCIENCE CHINA Information Sciences*.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. 2024. What matters when building vision-language models? *Advances in Neural Information Processing Systems*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast Inference from Transformers via Speculative Decoding. In *Proceedings of the 40th International Conference on Machine Learning*.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. 2024a. LLaVA-OneVision: Easy Visual Task Transfer. *Transactions on Machine Learning Research*.

Kevin Li, Sachin Goyal, João D. Semedo, and J. Zico Kolter. 2024b. Inference Optimal VLMs Need Fewer Visual Tokens and More Parameters. In *The Thirteenth International Conference on Learning Representations*.

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. 2024c. MVBench: A Comprehensive Multi-modal Video Understanding Benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual Instruction Tuning. *Advances in Neural Information Processing Systems*.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. 2024a. Video-ChatGPT: Towards Detailed Video Understanding via Large Vision and Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. 2024b. VideoGPT+: Integrating Image and Video Encoders for Enhanced Video Understanding. *Preprint*, arXiv:2406.09418.

Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Anton Belyi, Haotian Zhang, Karanjeet Singh, Doug Kang, Hongyu Hè, Max Schwarzer, Tom Gunter, Xiang Kong, Aonan Zhang, Jianyu Wang, Chong Wang, Nan Du, Tao Lei, Sam Wiseman, Mark Lee, Zirui Wang, Ruoming Pang, Peter Grasch, Alexander Toshev, and Yinfei Yang. 2024. MM1: Methods, Analysis and Insights from Multimodal LLM Pre-training. In *European Conference on Computer Vision*.

David Owen. 2024. How predictable is language model benchmark performance? *Preprint*, arXiv:2401.04757.

Viorica Patraucean, Lucas Smaira, Ankush Gupta, Adria Recasens Continente, Larisa Markeeva, Dylan Sunil Banarse, Skanda Koppula, Joseph Heyward, Mateusz Malinowski, Yi Yang, Carl Doersch, Tatiana Matejovicova, Yury Sulsky, Antoine Miech, Alexandre Fréchette, Hanna Klimczak, Raphael Koster, Junlin Zhang, Stephanie Winkler, Yusuf Aytar, Simon Osindero, Dima Damen, Andrew Zisserman, and Joao Carreira. 2023. Perception Test: A Diagnostic Benchmark for Multimodal Video Models. In *Thirty-Seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

ShengYun Peng, Seongmin Lee, Xiaojing Wang, Rajarajeswari Balasubramaniyan, and Duen Horng Chau. 2023. High-performance transformers for table structure recognition need early convolutions. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

ShengYun Peng, Seongmin Lee, Xiaojing Wang, Rajarajeswari Balasubramaniyan, and Duen Horng Chau. 2024. Unitable: Towards a unified framework for table structure recognition via self-supervised pretraining. *Preprint*, arXiv:2403.04822.

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. In *Proceedings of the 38th International Conference on Machine Learning*.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. 2024. Beyond Chinchilla-optimal: Accounting for inference in language model scaling laws. In *Proceedings of the 41st International Conference on Machine Learning*.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. In *The Thirteenth International Conference on Learning Representations*.

Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*.

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. 2024. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *Preprint*, arXiv:2409.12191.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. 2024a. LongVideoBench: A Benchmark for Long-context Interleaved Video-Language Understanding. In *38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks*.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024b. Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference for LLM Problem-Solving. In *The Thirteenth International Conference on Learning Representations*.

Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. 2021. NExT-QA: Next Phase of Question-Answering to Explaining Temporal Actions. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kexin Yi*, Chuang Gan*, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. 2019. CLEVRER: Collision Events for Video Representation and Reasoning. In *International Conference on Learning Representations*.

Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. 2019. ActivityNet-QA: A Dataset for Understanding Complex Web Videos via Question Answering. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. 2024. Inference Scaling for Long-Context Retrieval Augmented Generation. In *The Thirteenth International Conference on Learning Representations*.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2023. When Scaling Meets LLM Finetuning: The Effect of Data, Model and Finetuning Method. In *The Twelfth International Conference on Learning Representations*.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2024a. LMMs-Eval: Reality Check on the Evaluation of Large Multimodal Models. *Preprint*, arXiv:2407.12772.

Ruohong Zhang, Liangke Gui, Zhiqing Sun, Yihao Feng, Keyang Xu, Yuanhan Zhang, Di Fu, Chunyuan Li, Alexander G Hauptmann, Yonatan Bisk, and Yiming Yang. 2025. Direct Preference Optimization of Video Large Multimodal Models from Language Model Reward. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. 2024b. Video Instruction Tuning With Synthetic Data. *Preprint*, arXiv:2410.02713.

A Frequently Asked Questions (FAQs)

We address common questions with additional detail that may be missing from the main text's flow.

A.1 Why finetuning compute cost is negligible compared to inference compute under heavy demand?

To compare the costs of finetuning a video vision-language model (VLM) and running inference, consider the following. The cost of finetuning scales as $6x_N n$, where x_N is the model size (number of parameters) and n is the number of training tokens. In contrast, the cost of inference scales as $2x_N n_{\text{inf}}$, where n_{inf} is the total number of tokens processed during inference. The ratio of inference cost to training cost is therefore given by:

$$\text{Cost ratio} = \frac{2x_N n_{\text{inf}}}{6x_N n} = \frac{n_{\text{inf}}}{3n}.$$

Next, assume that both training and inference costs are dominated by the tokens used to represent a video data. If each video contributes roughly the same number of tokens during training and inference, n and n_{inf} can be treated as the number of videos instead of the number of tokens.

As an example, suppose finetuning is performed on $n = 1$ million videos. During deployment, the model processes $n_{\text{inf}} = 34$ million videos per day (daily upload to TikTok in 2024). Over the course of a month (30 days), the total number of videos processed for inference becomes $n_{\text{inf}} = 34 \times 30 = 1020$ million videos. Substituting these values into the cost ratio gives:

$$\text{Cost ratio} = \frac{n_{\text{inf}}}{3n} = \frac{1020}{3 \times 1} = 340.$$

Thus, in this scenario, the inference cost is approximately 340 times higher than the finetuning cost over a one-month deployment period. This illustrates the significant computational demands of large-scale inference compared to finetuning, especially for applications involving high volumes of video data.

Although one might consider increasing the finetuning data size n to make its compute cost comparable to inference, this is rarely practical nor necessary. Collecting high-quality annotations for large-scale video data is costly and labor-intensive. Moreover, as shown in Figure 10, scaling data alone yields diminishing returns, e.g., doubling the number of training videos leads to only marginal performance gains. Therefore, it is typically more effective to improve other factors, such as parameter count or number of tokens to represent a video, rather than dramatically increasing n. This supports our assumption that the finetuning compute cost is negligible compared to inference compute in real-world deployments.

A.2 Why "inference-compute optimal" in the paper title if we perform many finetuning runs?

Our focus is on optimizing inference compute costs—the dominant cost in real-world video VLM deployments—not finetuning. As shown in Equation 4, our optimization constrains inference compute while treating finetuning costs as negligible. Section 2 further contrasts our work with training or finetuning compute-optimal scaling studies.

This emphasis reflects practical deployment patterns: a VLM is finetuned once or infrequently, but serves inference requests continuously at scale. As detailed in Section A.1, monthly inference costs can far exceed the one-time finetuning cost—often by orders of magnitude.

Why, then, do we run multiple finetuning experiments? Because inference-time scaling factors (e.g., number of frames, tokens per frame) must match those used during finetuning for optimal performance. A model finetuned on 2-frame inputs performs poorly if asked to handle 64 frames at inference—it was simply never trained to process that much context. Our study jointly identifies optimal finetuning and serving configurations under a fixed inference compute budget. We believe that concurrent inference time scaling studies could also benefit from taking into account the post-training stage, e.g., if we scale inference compute by increasing the length of chain-of-thought reasoning traces, it's beneficial to adapt the pretrained model to generate longer answers in the first place.

A.3 Does vision model's compute cost matter for video VLM?

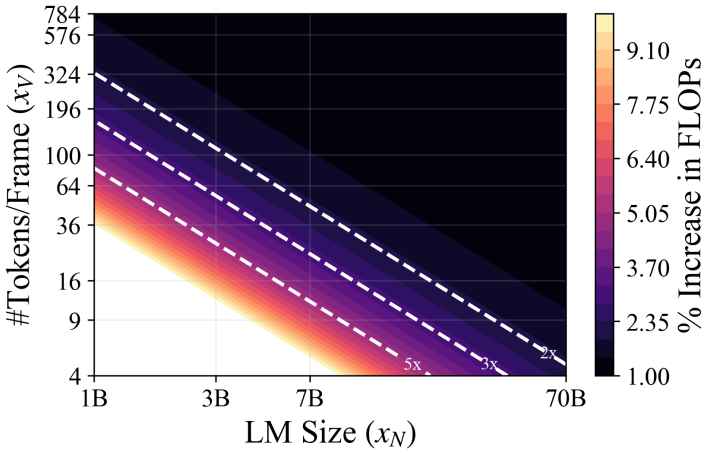

Figure 6: **% FLOPs Increase From Adding Vision Model Compute.** Contours illustrates the percentage increase in inference FLOPs, when accounting for the compute cost of the vision model, as a function of language model size x_N and the number of tokens per frame x_V. The vision encoder's compute cost significantly impacts the inference FLOPs of video VLMs. For example, for a 7B video VLM with $x_V \approx 50$, the vision model accounts for approximately half of the inference compute cost. This contribution becomes more significant for smaller x_N, x_V.

Previous research often considers only the compute cost of the language model, neglecting the vision encoder's compute cost (Li et al., 2024b; Du et al., 2024). Here, we emphasize the importance of accounting for the vision model's compute cost in video VLMs.

Using the standard approximation of $2x_N$ FLOPs per token for a transformer model with x_N parameters (Kaplan et al., 2020), the per-example inference compute costs for both the vision and language components of the video VLM are:

$$c_{\text{ViT}} = 2x_M x_T x_W, \quad c_{\text{LM}} = 2x_N x_T x_V,$$

where x_M is the number of vision model parameters, x_T the number of video frames, x_W the visual features output by the vision model, x_N the number of language model parameters, and x_V the visual tokens per frame after projection and resampling.

Including the vision model's compute cost, the percentage increase in total FLOPs is:

$$\%\ \text{Increase in FLOPs} = \frac{c_{\text{ViT}} + c_{\text{LM}}}{c_{\text{LM}}} = 1 + \frac{x_M x_W}{x_N x_V}.$$

For a 430M-parameter SoViT-400m/14 (Alabdulmohsin et al., 2023) outputting $x_W = 768$ visual features, this becomes:

$$\%\ \text{Increase in FLOPs} = 1 + \frac{0.43 \cdot 10^9 \cdot 768}{x_N x_V}.$$

The increase in compute is independent of x_T, as both vision and language model costs scale linearly with the number of frames. Instead, it depends on x_N and x_V, with smaller values amplifying the vision model's relative contribution. Omitting the vision model's compute in such cases underestimates the total inference cost for video vision-language models.

Figure 6 illustrates the percentage increase in inference FLOPs as a function of language model size x_N and the number of tokens per frame x_V. The contours highlight the substantial impact of the vision encoder's compute cost on the overall inference FLOPs of video VLMs. Notably, for a 7B video VLM with $x_V \approx 50$, the vision model contributes to approximately half of the total inference compute cost. This influence is even more pronounced in smaller video VLMs, i.e., smaller x_N, where the vision model's compute cost becomes a larger fraction of the total.

B Implementation Details

B.1 Video Instruction Tuning Dataset

Citation	Data Subset	Size	Fraction (%)
(Zhang et al., 2025)	LLaVA-Hound	255,000	11.6
(Zhang et al., 2024b)	LLaVA-Video-178K	1,335,500	60.9
(Yu et al., 2019)	ActivityNet-QA	23,530	1.1
(Xiao et al., 2021)	Next-QA	34,114	1.6
(Patraucean et al., 2023)	PerceptionTest	2,403	0.1
(Maaz et al., 2024a)	VideoInstruct100K	100,010	4.6
(Maaz et al., 2024b)	VCG+112K	112,716	5.1
(Maaz et al., 2024a)	VideoChatGPT Human Anno.	25,803	1.2
(Kunchang et al., 2025)	VideoChat	40,807	1.9
(Kay et al., 2017)	Kinetics-710	39,949	1.8
(Goyal et al., 2017)	Something-Something-v2	40,000	1.8
(Yi* et al., 2019)	CLEVRER	82,620	3.8
(Bain et al., 2021)	WebVid	99,922	4.6
	Total	2,192,374	100.0

Table 3: Video Instruction Tuning Dataset Composition

We compile a comprehensive video instruction-tuning dataset designed to explore the scaling behavior of finetuning data size and to achieve a performance level that remains relevant to both researchers and practitioners. This dataset, consisting of approximately 2.2 million examples, integrates data mixture used from two prior works: VideoGPT+ (Maaz et al., 2024b) and LLaVA-Video (Zhang et al., 2024b).

The dataset is sourced from a diverse array of video sources, e.g., LLaVA-Video-178K, and includes various types of instructions. For example, VideoInstruct100K, VCG+112K, and VideoChat datasets contain conversational data, while other subsets primarily focus on question-answering (QA). A smaller portion is dedicated to captioning tasks. Table 3 provides additional details.

To assess the dataset's effectiveness, we conduct a coarse ablation study by removing subsets such as LLaVA-Hound, the VideoChatGPT suite, and the QA datasets. Although some tasks show improved performance in these ablated setups, the complete dataset demonstrates the best average performance across downstream tasks, underscoring its overall efficacy.

B.2 Model & Training

Hyperparameter	Pretraining	Finetuning
Trainable components	Projector	Vision model, projector, and language model
Batch size	512	256
Learning rate	1e-3 (projector)	1e-5 (LM & projector), 2e-6 (vision tower)
Learning rate schedule	Cosine (3% warmup)	Cosine (3% warmup)
Weight decay	0	0
Optimizer	AdamW	AdamW
Epochs	1	1

Table 4: Training hyperparameters for pretraining (the projector) and video instruction tuning.

In this paper, we explore LLaVA-like architectures (Liu et al., 2023), which comprise three key components: a vision model, a projector, and a language model. The vision model, SoViT-400m/14 (Alabdulmohsin et al., 2023), is chosen for its high performance with minimal inference FLOPs and encodes each frame independently. The projector is a two-layer MLP that transforms visual features into tokens. For the language model, we use the Llama-3.2 series, available in sizes of 1B, 2.8B, and 7.5B parameters, to process these visual token representations for video-related tasks.

The finetuning process consists of two stages: pretraining the projector and video instruction tuning. Initially, we pretrain the MLP projector using the 558k LCS dataset (Liu et al., 2023), and this pretrained weight is employed in all subsequent finetuning experiments. We then finetune the entire model on video instruction tuning datasets.

To isolate the effects of video instruction tuning, we deliberately avoid using checkpoints that have been further fine-tuned on datasets specifically designed to enhance knowledge understanding or improve instruction following for images or multi-image inputs. In contrast, models like LLaVA-OneVision (Li et al., 2024a) and LLaVA-Video (Zhang et al., 2024b) incorporate such additional fine-tuning stages to boost performance. While this allows us to focus solely on evaluating the impact of video instruction tuning, it likely limit our model's performance compared to these approaches.

We adopt the codebase and default hyperparameters from Li et al. (2024a). Table 4 details the specific hyperparameters used during both pretraining and video instruction tuning.

B.3 Evaluation

Evaluation	Abbr.	QA Type	Long Vid.	LLM Judge	LLMs-Eval Task Name	Metrics
VideoDetailedDescription	VDC	caption		gpt-4o-mini	`video_dc499`	score
ActivityNet-QA	AQA	open-ended		gpt-4o-mini	`activitynetqa`	accuracy
VCGBench	VCG	open-ended		gpt-4o-mini	`videochatgpt`	score
LongVideoBench	LVB	MC	✓		`longvideobench_val_v`	accuracy
PerceptionTest	PT	MC			`perceptiontest_val_mc`	accuracy
MVBench	MV	MC			`mvbench`	accuracy
Video-MME	VMME	MC	✓		`videomme`	accuracy
Next-QA	NQA	MC			`nextqa_mc_test`	accuracy

Table 5: **Video Evaluation Benchmarks.** This table summarizes the evaluation benchmarks used to assess the capabilities of video VLMs. The benchmarks cover various question-answering (QA) types, including captioning, open-ended questions, and multiple-choice (MC) questions. Some benchmarks are designed for specific capabilities, e.g., long video understanding (under "Long Vid."), while others evaluate a whole suite of capabilities, e.g., Next-QA and Video-MME. The table also specifies whether a language model judge (e.g., gpt-4o-mini) is involved, the corresponding task name in LMMs-Eval (Zhang et al., 2024a), and the metrics used for assessment.

In this paper, we evaluate the video VLM on a diverse set of 8 downstream video tasks to ensure comprehensive assessment across various tasks and domains. For reproducibility, we use LMMs-Eval (Zhang et al., 2024a). The evaluation includes: (1) VideoDetailedCaption (VDC Zhang et al., 2024a) for detailed video descriptions, (2) ActivityNet-QA (AQA Yu et al., 2019) for action-related QA, (3) VCGBench (VCG Maaz et al., 2024a) for assessing video chat capabilities, (4) LongVideoBench (LVB Wu et al., 2024a) for long video understanding, (5) PerceptionTest (PT Patraucean et al., 2023) for fine-grained perception evaluation, and (6) MVBench (MV Li et al., 2024c), Video-MME (VMME Fu et al., 2024), and Next-QA (NQA Xiao et al., 2021) for broad evaluation across diverse video tasks and domains. This combination of benchmarks enables a thorough and reproducible analysis of the model's performance.

Table 5 offers a detailed summary of the video evaluation tasks employed to evaluate the capabilities of video VLMs. These tasks are categorized by the type of question-answering (QA) they address, including captioning, open-ended, and multiple-choice (MC) questions. The table highlights key features of each task, such as their emphasis on long video understanding (under "Long Vid."). It also specifies whether a language model judge, like gpt-4o-mini, is involved in the evaluation process. Prior work rely on gpt-3.5-turbo-0613 as the LLM judge that has since been deprecated, we transition to use gpt-4o-mini-2024-07-18 instead. Each task is linked to a specific task name within the LMMs-Eval framework (Zhang et al., 2024a), and the metrics used for assessment, such as scores generated by the LLM judge or multiple-choice accuracy, are also provided. To ensure consistency, we convert both accuracy and the LLM judge's scores to $(0, 100)$, allowing for a balanced average across evaluation metrics. We use "Metrics/Avg" to denote the average task performance.

C Training Sweeps

Sweep	n (in million)	x_N (in billion)	x_T	x_V	Comment
Star	{0.25, 0.5, 1}	{1, 2.8, 7.5}	32	196	Vary x_N
	{0.25, 0.5, 1}	7.5	{4, 8, 12, 16, 32}	196	Vary x_T
	{0.25, 0.5, 1}	7.5	32	{4, 16, 25, 36, 49, 100, 196}	Vary x_V
IsoFLOP	2	1	{(2, 196), (3, 9), (3, 16)}		$c(x) \approx 2$ TFLOPs
		2.8	{(2, 64), (3, 4)}		
		7.5	{(2, 25), (3, 1)}		
		1	{(4, 289), (5, 169), (6, 81), (7, 25)}		$c(x) \approx 5$ TFLOPs
		2.8	{(3, 169), (4, 100), (5, 64), (6, 25), (7, 9)}		
		7.5	{(2, 121), (3, 64), (6, 9), (7, 4)}		
		1	{(8, 625), (11, 361), (16, 144), (20, 49), (22, 16)}		$c(x) \approx 15$ TFLOPs
		2.8	{(6, 324), (11, 121), (16, 49), (19, 25), (20, 16)}		
		7.5	{(7, 100), (8, 81), (15, 25), (19, 9), (21, 4)}		
		1	{(16, 625), (26, 256), (32, 144), (37, 81), (40, 49)}		$c(x) \approx 30$ TFLOPs
		2.8	{(9, 484), (17, 196), (22, 121), (27, 81), (35, 36)}		
		7.5	{(16, 81), (25, 36), (29, 25), (38, 9)}		

Table 6: **Star and IsoFLOP Sweep Setup**. Scaling factors x and finetuning data sizes n used in the experiments.

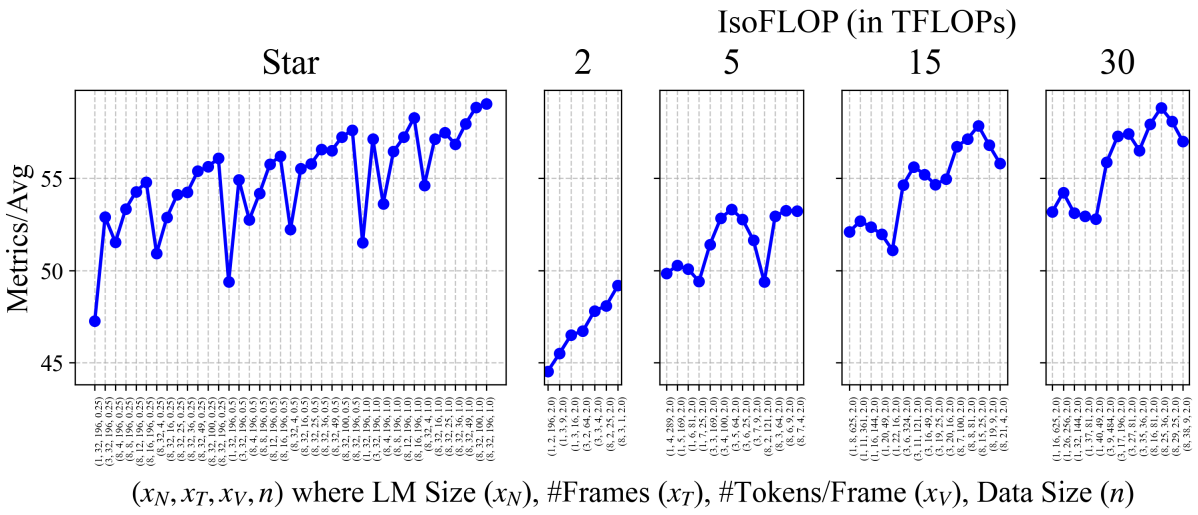

Figure 7: **Empirical Data from Star and IsoFLOP Sweeps**. We show the average task performance across different runs of the star and isoFLOP sweeps. The data gathered from sweeps is used for visualization and parametric fitting of scaling curves.

In this paper, we conduct two types of sweeps on scaling factors x and finetuning data size n: (1) the star sweep, proposed in SoViT (Alabdulmohsin et al., 2023), and (2) the isoFLOP sweep, used by Chinchilla (Hoffmann et al., 2022). We finetune a video VLM using scaling factor x on datasets of size n, and evaluate the model to obtain data points $((x, n), f(x, n))$. These data points are used for visualization and parametric fitting. Table 6 lists the scaling factors x and finetuning data sizes n used in our experiments. Figure 7 illustrates the average task performance for each run of the star and isoFLOP sweeps.

Star Sweep We start with an inference compute-intensive set of scaling factors, the "star center" $x^\star = (7.5\text{B}, 32, 196)$. We vary one factor at a time while keeping the others fixed, and finetune the video VLM on three data sizes: $\{0.25\text{M}, 0.5\text{M}, 1\text{M}\}$. This approach allows for more accurate estimation of the scaling exponent for each factor as the finetuning data size increases. Additionally, it enables exploration on how individual scaling factor interact with finetuning data size.

The star sweep approach avoids a brute-force grid search over (x_N, x_T, x_V, n) to estimate scaling parameters across all dimensions. In Alabdulmohsin et al. (2023), the star center $x^\bigstar$ is set larger than the scaling factors used in the sweep to prevent bottlenecks. For example, with a star center having `MLP dim` of 6144, the sweep explores values for `MLP dim` in a grid $(1088, 1360, 1728, 2160, 2592, 3072)$, all much smaller than 6144. In contrast, our sweep matches the largest scaling factor to the star center, with the next largest about half its value, followed by more densely sampled smaller values. For instance, with $x_T^\bigstar = 32$, we use the grid $(4, 8, 12, 16, 32)$ for the sweep. This provides more data points where scaling factors are smaller than the star center, and fitting a parametric function with equal weighting will prioritize these smaller values. Due to computational constraints, we set a smaller star center than the ideal, e.g., $(70\text{B}, 64, 512)$, resulting in a minor deviation from the prescribed strategy, which we consider reasonable given the trade-offs.

IsoFLOP Sweep We vary scaling factors $x = (x_N, x_T, x_V)$ to maintain a constant inference compute cost $c(x)$ (defined in Equation 1) across four target FLOPs: 2, 5, 15, 30. Each model is finetuned on approximately 2 million examples. For each target FLOP, we exhaustively search for scaling factor combinations within $\epsilon = 0.03$ of the target and select a well-spaced subset to ensure sufficient coverage while minimizing the number of runs. Additional points are added post-sweep to ensure the empirical compute-optimal frontier for average downstream task performance lies within each isoFLOP curve, as shown in Figure 7. The isoFLOP sweep addresses key questions, such as identifying the optimal set of scaling factors for a given inference FLOP budget and assessing the impact of jointly scaling multiple factors. It also serves as a held-out validation set for model evaluation and selection.

D Modeling and Fitting of Task Performance

D.1 Parametric Functions

Form	Expressions for $f(x, n)$		$x^*(c)$ ind. n
	Multiple Factors ($K > 1$)	Single Factor ($K = 1$)	
`mult`	$\alpha(\prod_k x_k^{-a_k})n^{-d} + \varepsilon$	$\alpha x^{-a} n^{-d} + \varepsilon$	✓
`add`	$\sum_k \alpha_k x_k^{-a_k} + \xi n^{-d} + \varepsilon$	$\alpha x^{-a} + \xi n^{-d} + \varepsilon$	✓
`add-interact`$_s$	$\sum_k \alpha_k x_k^{-a_k} + \sum_k \beta_k x_k^{b_k} n^{-d} + \varepsilon$	$\alpha x^{-a} + \beta x^{b} n^{-d} + \varepsilon$	×
`add-interact`	$\sum_k \alpha_k x_k^{-a_k} + \sum_k \beta_k x_k^{b_k} n^{-d} + \xi n^{-d} + \varepsilon$	$\alpha x^{-a} + \beta x^{b} n^{-d} + \xi n^{-d} + \varepsilon$	×

Table 7: **Parametric Models of Task Performance.** This table summarizes the parametric functional forms used to model task performance, distinguishing between cases where all scaling factors x are modeled jointly with the finetuning data size n ($K > 1$) and cases where each scaling factor is modeled individually with n ($K = 1$). The `add-interact` form combines additive power laws with interaction terms between x_k and n. `add-interact`$_s$ simplifies this by removing the standalone term dependent on n. `add` represents a standard additive power law, while `mult` corresponds to a multiplicative power law. Coefficients (e.g., $\alpha, \beta, \xi, \varepsilon$) and scaling exponents (e.g., a, b, d) are denoted by Greek and English letters, respectively. The final column indicates whether varying n affects the optimal scaling factors that is the solution to the optimization problem defined in Equation 4.

We investigate several parametric functions to model task error in video VLMs as a function of scaling factors x and finetuning data size n. These functions capture the diminishing returns observed in scaling laws, where increasing x or n yields progressively smaller performance gains. For example, adding more frames or increasing the finetuning data size improves performance initially, but the benefit diminishes as the model saturates in available information or data. This behavior parallels the concept of diminishing marginal utility in economics.

The primary function we use is the `add-interact` form, which combines additive power-law terms with interaction terms between each scaling factor x_k and n. This formulation is particularly effective for analyzing the joint influence of scaling factors and data size on performance under constraints on inference compute $c(x)$ and data size n. This approach was first proposed to study the relationship between architecture factors (e.g., width, depth) and pretraining data size when scaling ViT models (Alabdulmohsin et al., 2023). A simplified variant, `add-interact`$_s$, removes the standalone term dependent on n. We also evaluate the `add` model, a special case of `add-interact` that assumes independent contributions from x and n. This additive power-law model is widely used in pretraining scaling law studies (Kaplan et al., 2020; Hoffmann et al., 2022) and has been adapted for finetuning scenarios (Du et al., 2024). Additionally, we consider the `mult` model, a multiplicative power-law formulation that is sometimes preferred in finetuning scaling law studies (Wei et al., 2022; Li et al., 2024b).

Table 7 provides the expressions for each parametric function. By comparing these formulations, we aim to identify the most effective model for capturing the scaling behavior of video VLMs.

D.2 What is the effect of exponent parameter b in `add-interact` parametric function?

We analyze the role of the exponent parameter $b \in \mathbb{R}$ in the `add-interact` parametric form. For simplicity, we redefine the function for a scalar x as

$$f(x, n) = \alpha x^{-a} + (\beta x^b + \xi) n^{-d} + \varepsilon. \tag{6}$$

The parameter b controls how the scaling factor x influences the coefficient $(\beta x^b + \xi)$, which represents the reducible error associated with scaling the finetuning data size n. A positive $b > 0$ has two key effects:

1. **Larger x Requires More Data to Match the Same Error:** When $b > 0$, the coefficient $(\beta x^b + \xi)$ increases with x. For two values $x_{\mathrm{S}} < x_{\mathrm{L}}$, we have $(\beta x_{\mathrm{S}}^b + \xi) < (\beta x_{\mathrm{L}}^b + \xi)$. To achieve the same error $f(x, n)$, a larger x_{L} requires a larger finetuning data size n_{L} compared to n_{S}. Specifically, the equality $(\beta x_{\mathrm{S}}^b + \xi) n_{\mathrm{S}}^{-d} = (\beta x_{\mathrm{L}}^b + \xi) n_{\mathrm{L}}^{-d}$ holds only if $n_{\mathrm{S}} < n_{\mathrm{L}}$. Thus, $b > 0$ increases the data required for larger x to match the performance of smaller x.

2. **Larger x Reduces Error Faster Per Example:** A positive b also increases the marginal benefit of finetuning data for larger x. The derivative of the reducible error term with respect to n is $\frac{\partial}{\partial n}(\beta x^b + \xi) n^{-d} = -d(\beta x^b + \xi) n^{-d-1}$. For larger x, the term $(\beta x^b + \xi)$ is larger, making the derivative more negative. This implies that the absolute rate of error reduction (i.e., the decrease in error per additional finetuning example) is greater for larger x when $b > 0$. As a result, $b > 0$ improves the efficiency of finetuning for larger x.

In contrast, when $b < 0$, the effects are reversed: larger x requires less data to achieve the same error, but the marginal benefit of each additional finetuning example decreases.

D.3 Fitting the Parametric Model

loss	Star+IsoFLOP CV(5-fold)			Star → IsoFLOP		
	MSE ↓	$E_{\%}$ ↓	R^2 ↑	MSE ↓	$E_{\%}$ ↓	R^2 ↑
Huber	0.44	0.97	0.95	1.11	1.38	0.91
MSE	0.2	0.77	0.98	0.95	1.33	0.92

init	Star+IsoFLOP CV(5-fold)			Star → IsoFLOP		
	MSE ↓	$E_{\%}$ ↓	R^2 ↑	MSE ↓	$E_{\%}$ ↓	R^2 ↑
`zero`	0.43	0.99	0.95	1.36	1.63	0.89
`r(100)`	0.34	0.88	0.96	0.96	1.36	0.92
`r(500)`	0.2	0.77	0.98	0.95	1.33	0.92

bound	Star+IsoFLOP CV(5-fold)			Star → IsoFLOP		
	MSE ↓	$E_{\%}$ ↓	R^2 ↑	MSE ↓	$E_{\%}$ ↓	R^2 ↑
✓	0.57	1.11	0.94	2.45	2.42	0.8
×	0.2	0.77	0.98	0.95	1.33	0.92

log space	Star+IsoFLOP CV(5-fold)			Star → IsoFLOP		
	MSE ↓	$E_{\%}$ ↓	R^2 ↑	MSE ↓	$E_{\%}$ ↓	R^2 ↑
×	0.29	0.88	0.97	19.72	2.93	-0.62
✓	0.2	0.77	0.98	0.95	1.33	0.92

Table 8: **Ablation Study on Fitting Procedures.** We analyze the impact of key design choices for fitting `add-interact` to predict average task performance across two setups: (1) *in-distribution* evaluation using 5-fold cross-validation (CV) on combined star and isoFLOP data, and (2) *extrapolation* on isoFLOP data after training on star data. Metrics include mean squared error (MSE), average relative error ($E_{\%}$), and coefficient of determination (R^2). The ablations include: (1) *loss function* (top left): MSE provides better fit than Huber loss, (2) *parameter initialization* (top right): e.g., `r(500)` initializes parameters randomly within a range and selects the best model from 500 runs, while `zero` sets all parameters to zero, (3) *positivity constraint* (bottom left): enforcing positivity for the exponents degrades performance, and (4) *log-space computation* (bottom right): transforming the scaling function into log space improves numerical stability. Results highlight the importance of careful parameter initialization and log-space computation, while positivity constraints are detrimental, and both loss functions perform comparably.

To estimate the parameters θ of the parametric model of task performance $f(x, n)$, we minimize the relative error between predicted and observed performance in log space:

$$\min_{\theta} \sum_{\text{Run } i} \left(\log f(x^{(i)}, n^{(i)}; \theta) - \log f^{(i)} \right)^2 . \tag{7}$$

We solve the above optimization problem using `scipy`'s implementation of the L-BFGS algorithm. Table 8 presents an ablation study on key design choices for fitting the parametric model. Below, we summarize the findings:

- **Loss Function**: We compare mean squared error (MSE) and Huber loss. MSE consistently outperforms Huber loss and is therefore used as the objective.
- **Parameter Initialization**: The optimization problem is nonconvex and nonlinear, so we mitigate the risk of poor local minima by running 500 trials with random initializations and selecting the best fit. Coefficient parameters (e.g., $\alpha, \beta, \xi, \varepsilon$) are sampled uniformly from $(0, 30)$, while exponent parameters (e.g., a, b, d) are sampled from $(-1, 1)$.
- **Positivity Constraints**: Unlike Alabdulmohsin et al. (2023), we do not enforce positivity constraints on exponent parameters. Ablation results show that such constraints degrade performance.
- **Log-Space Computation**: Following Hoffmann et al. (2022), we compute $f(x, n)$ in log space. This improves numerical stability and yields better performance across all metrics.

In summary, we fit the parametric models by minimizing relative error in log space using MSE loss. Parameters are initialized randomly, and the best fit is selected from 500 runs. We avoid positivity constraints on exponents and leverage log-space computation for improved stability and accuracy.

D.4 Bootstrap Aggregation

# Resamples	Aggregation	Train			Star → IsoFLOP		
		MSE ↓	$E_{\%}$ ↓	R^2 ↑	MSE ↓	$E_{\%}$ ↓	R^2 ↑
100	mean	0.15	0.58	0.99	4418	28	-362
1	median	0.17	0.62	0.98	1.24	1.55	0.9
50	median	0.14	0.57	0.99	0.96	1.35	0.92
100	median	0.14	0.56	0.99	0.95	1.33	0.92

Table 9: **Impact of Bootstrap Aggregation Design Choices.** This table evaluates the effect of bootstrap aggregation strategies when using `add-interact` to predict average task performance on training results from the isoFLOP sweep when trained on that is the star sweep. Metrics include mean squared error (MSE), average relative error ($E_{\%}$), and coefficient of determination (R^2). Results show that median aggregation consistently outperforms mean aggregation. Performance improves with more bootstrap resamples (or fitted base models), with the best results achieved using median aggregation with 100 resamples that we adopt.

Figure 3 (left) highlights the high variance in parameter estimates for the parametric model of task performance when trained on approximately 100 examples. This variance poses a significant challenge due to the limited data. To mitigate this, we apply bootstrap aggregation (bagging), where multiple base models are trained on bootstrap-resampled datasets, and their predictions are aggregated.

Table 9 summarizes the ablation study on the bagging setup, analyzing the effects of the number of resamples and the aggregation method on model performance. The results demonstrate that median aggregation consistently outperforms mean aggregation across all metrics. Furthermore, increasing the number of resamples improves performance, with the best results obtained using 100 resamples and median aggregation. This setup significantly improves goodness-of-fit, and it is adopted in our work.

E Measuring Sensitivity of Compute-Optimal Frontier $x^*(c, n)$ to Data Size n

$\lvert\mathsf{C}\rvert$	e_N	e_T	e_V
1	0.0	0.0	0.0
10	-0.19	0.17	0.58
50	-0.23	0.16	0.89
100	-0.23	0.16	0.88
200	-0.22	0.17	0.78
300	-0.22	0.17	0.79

$\lvert\mathsf{N}\rvert$	e_N	e_T	e_V
1	-0.04	0.05	0.12
10	-0.22	0.17	0.77
20	-0.22	0.17	0.78
50	-0.22	0.17	0.79
100	-0.22	0.17	0.79

$\max(\mathsf{N})$	e_N	e_T	e_V
1	-0.04	0.05	0.12
2	-0.06	0.08	0.19
4	-0.12	0.12	0.42
6	-0.16	0.13	0.58
8	-0.19	0.15	0.7
10	-0.22	0.17	0.79

$\triangle n$	e_N	e_T	e_V
0.1	0.32	0.2	1.13
0.5	-0.08	0.2	0.99
1.0	-0.15	0.2	0.96
2.0	-0.2	0.19	0.89
3.0	-0.23	0.18	0.83
4.0	-0.23	0.17	0.8
5.0	-0.22	0.17	0.79
6.0	-0.22	0.17	0.78
7.0	-0.22	0.16	0.75
8.0	-0.21	0.16	0.72
9.0	-0.2	0.15	0.68
10.0	-0.19	0.15	0.64

Table 10: **Ablation Study on Elasticity Computation.**This table evaluates the impact of key parameters on the computation of elasticity, including: (1) the number of inference compute budgets $\lvert\mathsf{C}\rvert$ (measured in TFLOPs) used for averaging, (2) the number of finetuning data sizes $\lvert\mathsf{N}\rvert$ (measured in millions) used for averaging, (3) the maximum finetuning data size $\max(\mathsf{N})$ (upper bound of data sizes in N), and (4) the step size $\triangle n$ (in millions) used for the forward difference approximation of the derivative. Results show that elasticity values stabilize with larger $\lvert\mathsf{C}\rvert$ and $\lvert\mathsf{N}\rvert$, so we set $\lvert\mathsf{C}\rvert = 300$ and $\lvert\mathsf{N}\rvert = 100$ to minimize variability. While $\max(\mathsf{N})$ affects the magnitude of elasticity, it preserves consistent trends across $k \in \{N, T, V\}$. We set $\max(\mathsf{N}) = 10$M to ensure broad coverage of data sizes. The step size $\triangle n$ affects the stability and precision of the derivative approximation; we select $\triangle n = 5$ to balance this trade-off. These choices ensure reliable and interpretable elasticity estimates.

Figure 4 shows that the inference compute-optimal frontier $x_k^*(c)$, shifts up or down depending on the finetuning data size n. To quantify both the direction and magnitude of this shift, we define "elasticity"

$$e_k(c, n) = \frac{\partial x_k^*(c; n)}{\partial n} \cdot \frac{n}{x_k^*(c; n)} \tag{8}$$

to measure the relative sensitivity of x_k^* to variations in n for a given inference compute budget c.

We use elasticity instead of simple partial derivatives, e.g., $\partial x_k^*(c; n)/\partial n$, because partial derivatives are highly sensitive to the scale of the variables involved. In our case, the scaling factors x and the data size n differ significantly in magnitude: for example, LM size x_N is on the order of billions, x_T & x_V are in the tens or hundreds, and n is in the millions. Elasticity allows us to express relationships in percentage terms, making comparisons more intuitive. For instance, an elasticity value of $e_T = 0.1$ means that a 1% increase in n results in a 0.1% increase in x_T^*. This approach simplifies the interpretation of results and enables easier comparison of the sensitivity of x^* to changes in n.

Elasticity is commonly used in economics to measure how changes in one variable affect another, e.g., price elasticity of demand quantifies how price changes influence the quantity of goods demanded. This quantity is derived by solving a utility maximization problem under (monetary) budget constraints, often yielding closed-form solutions that allow direct computation of elasticity. In contrast, our approach solves the inference compute allocation problem in Equation 4 using discrete search, which does not yield a closed-form expression for elasticity. Instead, we approximate elasticity using forward differences:

$$e_k(c, n) \approx \frac{x_k^*(c; n + \triangle n)}{\triangle n} \cdot \frac{n}{x_k^*(c; n)} \tag{9}$$

where $\triangle n$ is the step size. To evaluate the overall impact of n on x_k^*, we calculate the average elasticity across compute budgets as

$$e_k(n) = \frac{1}{\lvert\mathsf{C}\rvert} \sum_{c \in \mathsf{C}} e_k(c, n), \tag{10}$$

where C represents the set of inference budgets of interest. We aggregate elasticity over both compute budgets and data sizes as

$$e_k = \frac{1}{\lvert\mathsf{C}\rvert\lvert\mathsf{N}\rvert} \sum_{c \in \mathsf{C}} \sum_{n \in \mathsf{N}} e_k(c, n), \tag{11}$$

with N denoting the set of finetuning data sizes of interest. These aggregated metrics provide a concise summary of how data size influences optimal scaling factors.

Table 10 presents an ablation study analyzing key design choices for computing elasticity. The findings are summarized as follows:

- **Size of C:** We vary $|\mathsf{C}|$ (evenly spaced between 1 TFLOP and 100 TFLOPs) and observe that elasticity values stabilize as $|\mathsf{C}|$ increases. To minimize sensitivity to $|\mathsf{C}|$, we set $|\mathsf{C}| = 300$.
- **Size of N:** We vary $|\mathsf{N}|$ (evenly spaced between 1M and 10M) and find that elasticity values stabilize with larger $|\mathsf{N}|$. To reduce variability, we set $|\mathsf{N}| = 100$.
- **Maximum Data Size $\max(\mathsf{N})$:** Ablating $\max(\mathsf{N})$ (with N sampled evenly starting from 1M examples) reveals that elasticity magnitudes increase consistently with scaling. We set $\max(\mathsf{N}) = 10$M to ensure a broad range of data sizes is covered.
- **Step Size $\triangle n$:** The step size $\triangle n$ significantly affects elasticity. Small $\triangle n$ (e.g., adding one example) leads to near-zero elasticity except at points of sudden jumps, making numerical approximations unstable and sensitive to C and N. Larger $\triangle n$ reduces accuracy of the forward difference approximation of the derivative. We select $\triangle n = 5$ to balance stability and precision.

The resulting setup ensure reliable elasticity estimates.

F Additional Results

F.1 Task-Specific Interpretation of Training Run Results

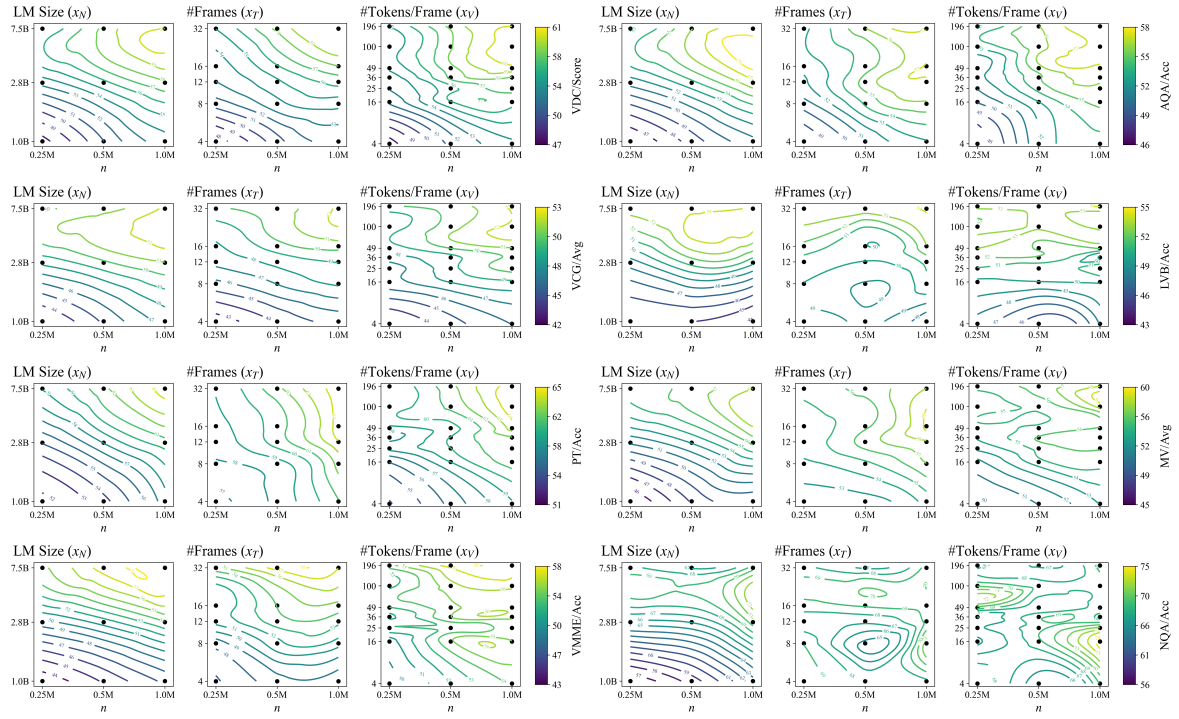

Figure 8: **Task-Specific IsoPerformance Contours.**

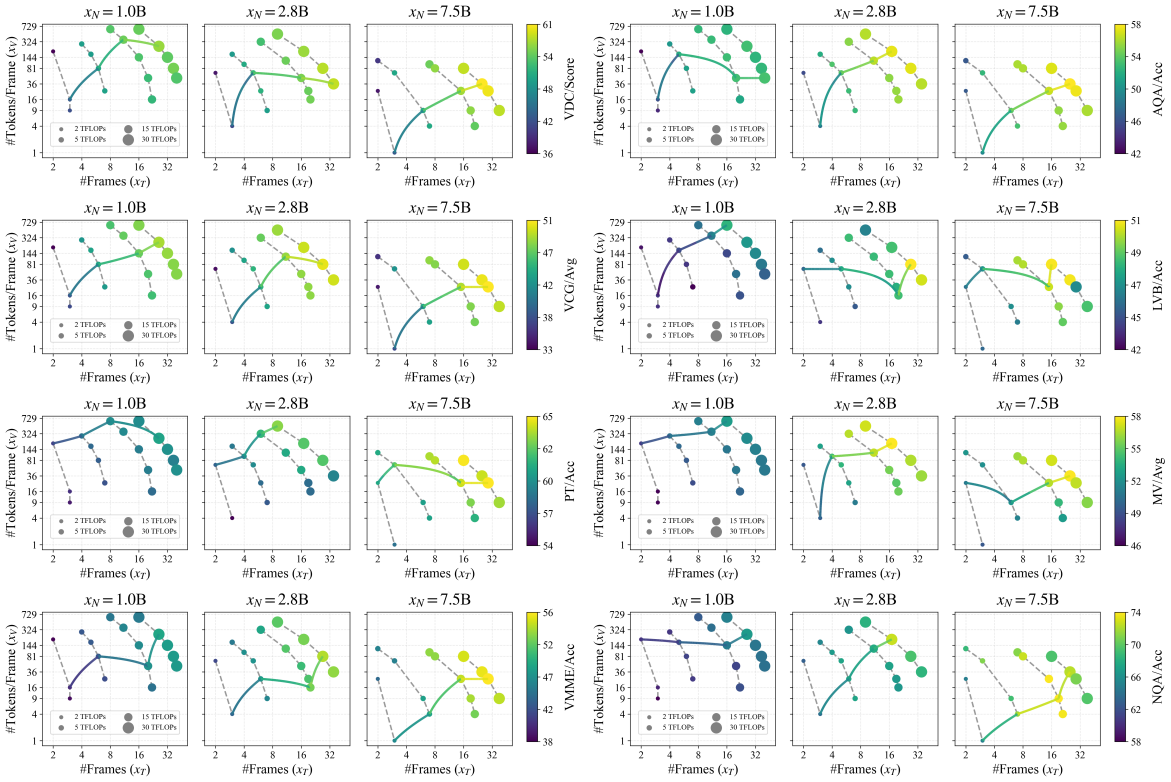

Figure 9: **Task-Specific isoFLOPs Curves and Compute-Optimal Frontier**.

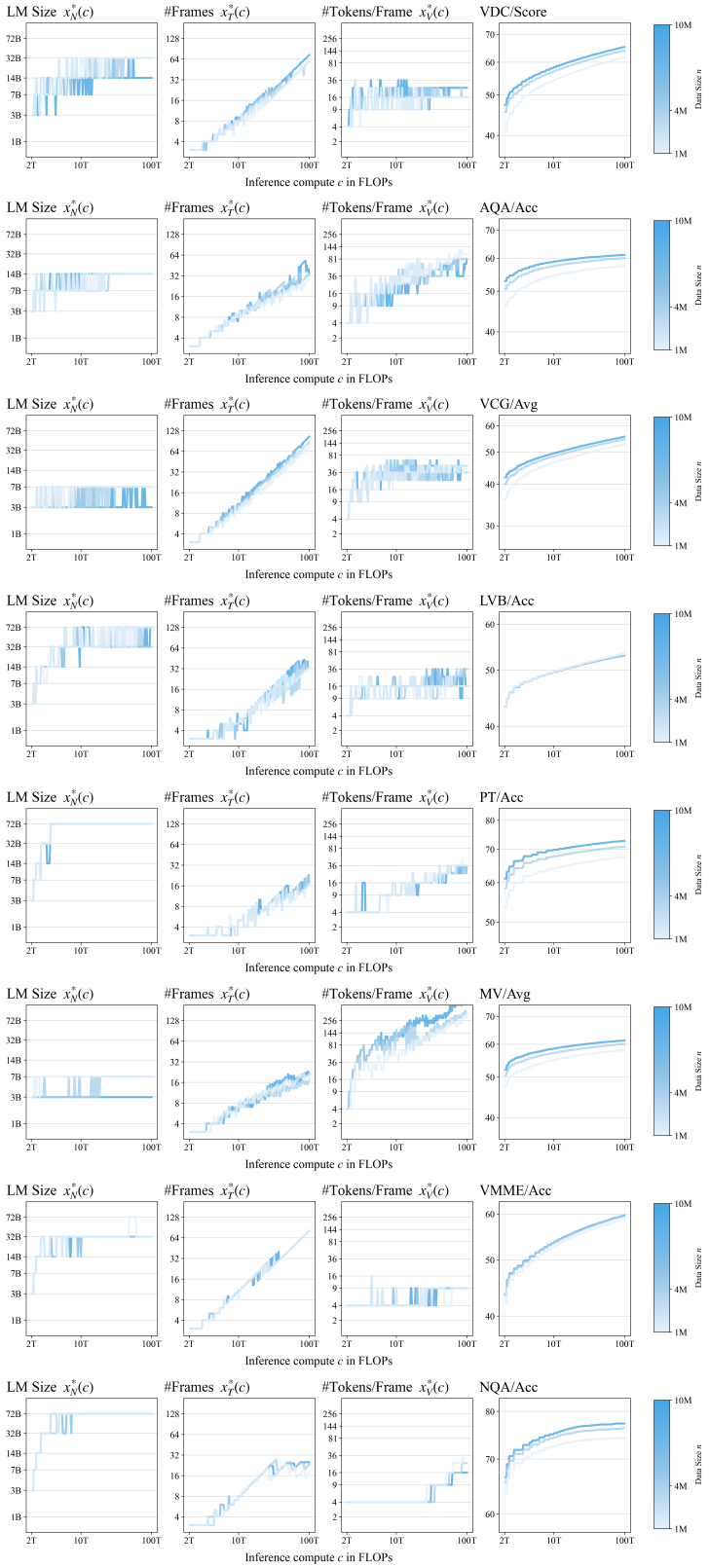

Figure 10: **Task-Specific Predicted Compute-Optimal Frontier.**