Inference Compute-Optimal Video Vision Language Models (vVLMs)

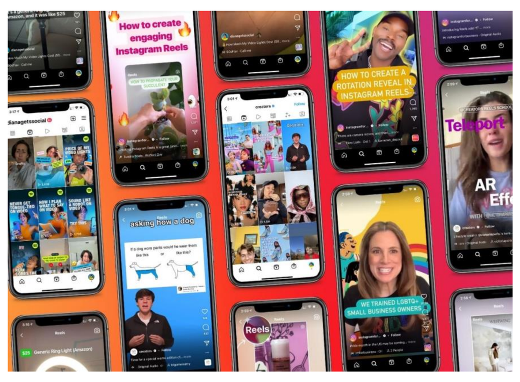
ACL 2025 VIENNA

Peiqi Wang, ShengYun Peng, Xuewen Zhang, Hanchao Yu, Yibo Yang, Lifu Huang, Fujun Liu, Qifan Wang

Code

Paper

Optimizing for inference is crucial for deploying video VLMs at scale



Reels

Tik Tok

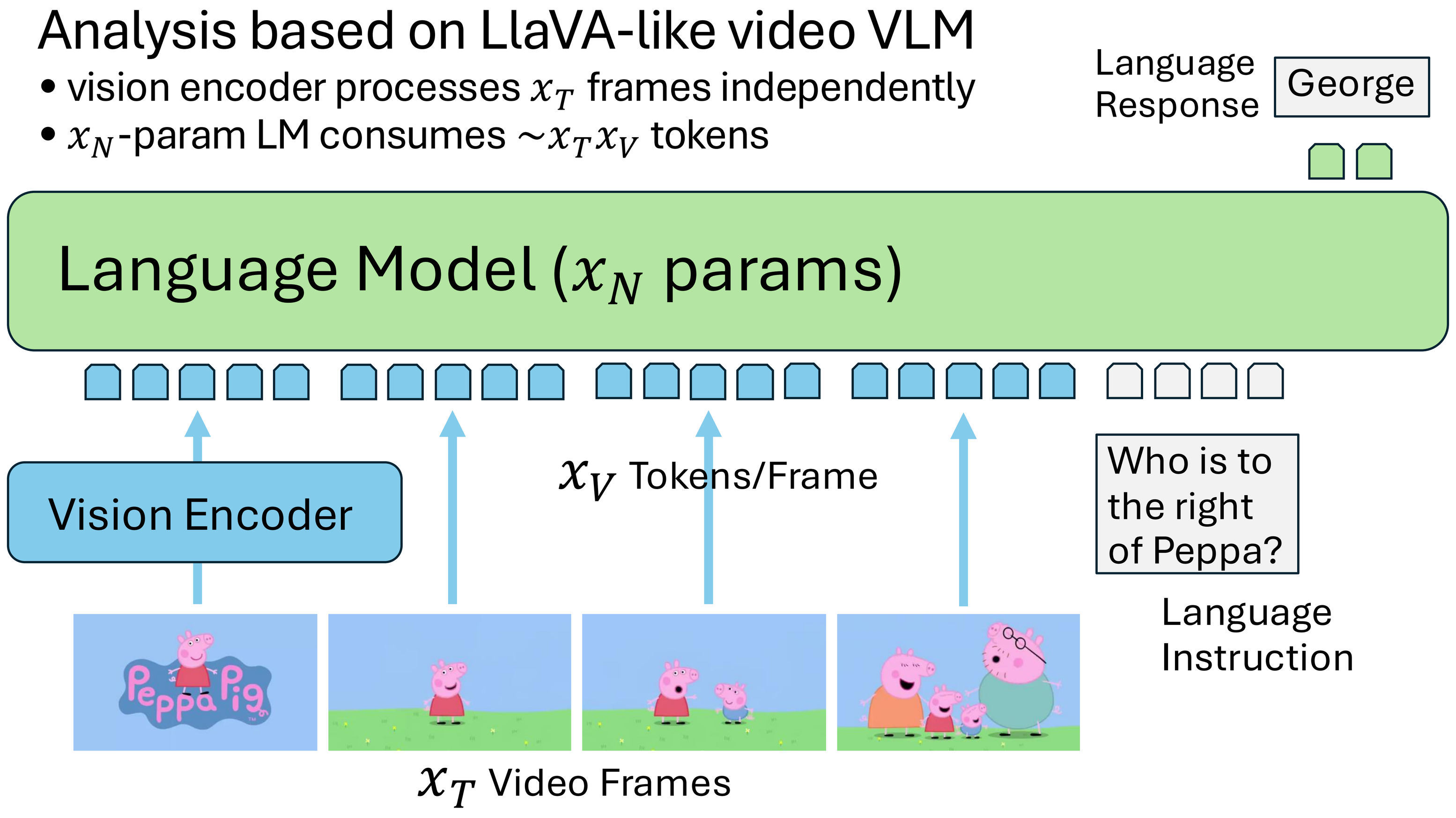
Shorts

Use vVLM to understand videos, e.g., topic tagging, flagging policy violation

- 1M videos for finetuning vs. ~1B videos/month (e.g., TikTok) inference
  → ~340× higher inference compute cost than finetuning compute cost

Analysis based on LLaVA-like video VLM

- vision encoder processes x_T frames independently
- x_N-param LM consumes $\sim x_T x_V$ tokens



Scaling factors $x = (x_N, x_T, x_V)$ affects ① inference compute cost $c(x)$ ② downstream task error f

Experimental Setups

- Dataset
  ~2.2M video instruction datasets, including chat, QA, captioning
- Model: LLaVA-like architecture
  - SoViT-400m/14 vision encoder
  - Llama-3.2 series of LMs $x_N \in \{1B, 3B, 8B\}$
- Finetuning
  ① pretrain projector
  ② full finetuning using instruction datasets
- Evaluation
  8 video benchmarks: Video Detailed Caption, ActivityNet-QA, benchmarks VCGBench, LongVideoBench, PerceptionTest, MVBench, Video-MME, Next-QA
  ✅ Metrics: QA accuracy; LM's rating for captions & open-ended QA

Q: Given fixed finetuning data of size n & inference compute budget c (in FLOPs), How to select scaling factors x with the smallest task error f?
where x_N = LM size, x_T = number of frames, x_V = tokens/frame

Optimizing LM training compute vs. vVLM inference compute.
→ we ignore effect of data size n on compute!

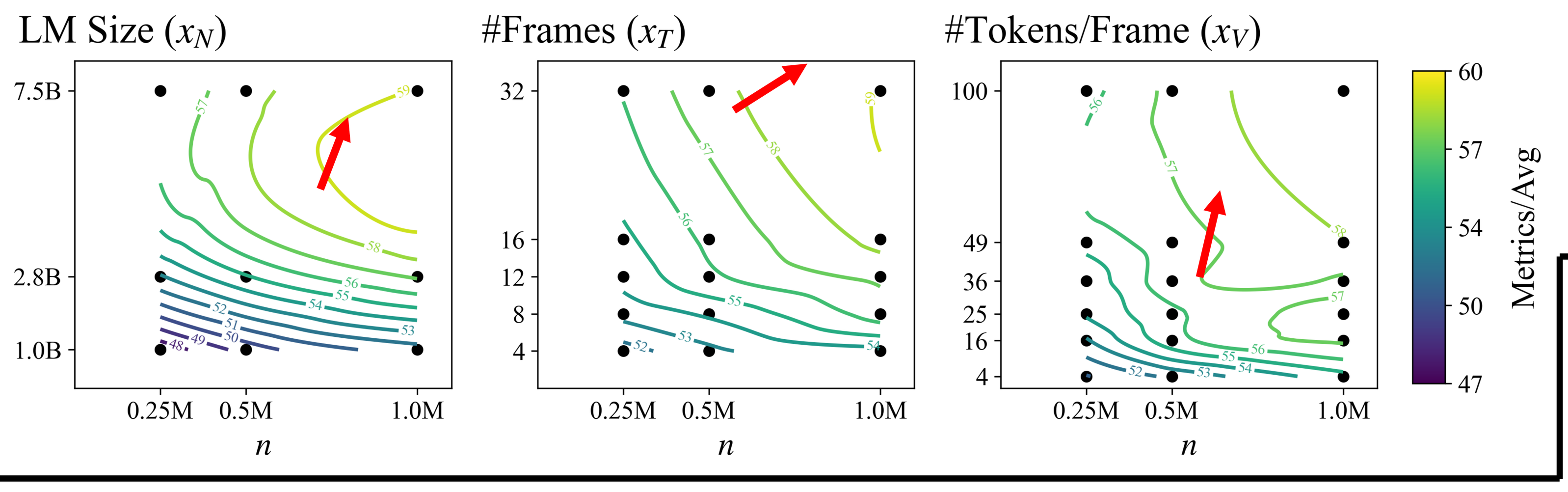
Compute	Problem
Pretraining	$\min_{x_N, n_{pt}: c(x_N, n_{pt}) \leq c_{pt}} f(x_N, n_{pt})$
Inference	$\min_{x: c(x) \leq c} f(x, n)$

Training Sweeps

finetune & evaluate vVLMs to collect $(x^{(i)}, n^{(i)}, f^{(i)})$

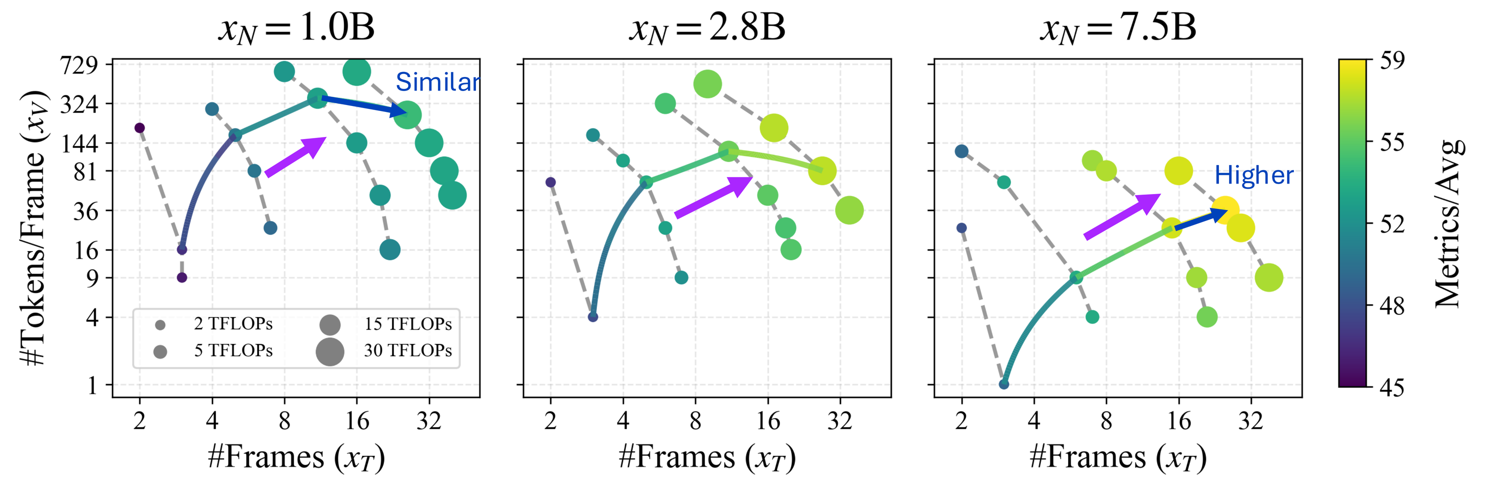
- **Star sweep**: vary one x_k at a time while keeping others fixed around the "star center" $x^* =$ (8B, 32, 196) and finetune vVLM on three data sizes n (in millions): 0.25, 0.5, and 1
- **IsoFLOP sweep**: adjust scaling factors (x_N, x_T, x_V) to maintain a fixed inference compute cost $c(x)$ across 4 target TFLOPs: 2, 5, 15, 30 and finetune vVLM on 2M sample

📌 Performance improves as x and n increase, albeit at a diminishing rate



- Fixing LM params x_N, better performance when increase x_T, x_V jointly
- 15 → 30 TFLOPs: larger LM (x_N = 7.5B) makes better use of 2× compute implies bottleneck imposed by LM size!

📌 Important to jointly scale x_N, x_T, and x_V to maximize performance



Performance Modeling

modeling & fitting of $f(x, n; \theta)$

Model task error using add−interact, a simple additive power-law relationship with interaction terms

for k-th scaling factor: $f_k(x_k, n) = \alpha_k x_k^{-a_k} + (\beta_k x_k^{b_k} + \xi_k) n^{-d} + \varepsilon_k$

- Coefficients α_k, β_k, ξ_k represent error reducible by increasing x_k or n
- Coefficient ε_k accounts for irreducible error
- Exponent a_k describe how error scales with x_k in data-unbounded regime
- Exponent d quantifies how error decreases with increasing n
- Exponent b_k determines how x_k affects the impact of increasing n

Estimate $\theta = \{\alpha_k, \beta_k, \xi_k, \varepsilon_k, a_k, b, d\}$ by minimizing MSE between predicted vs. observed performance

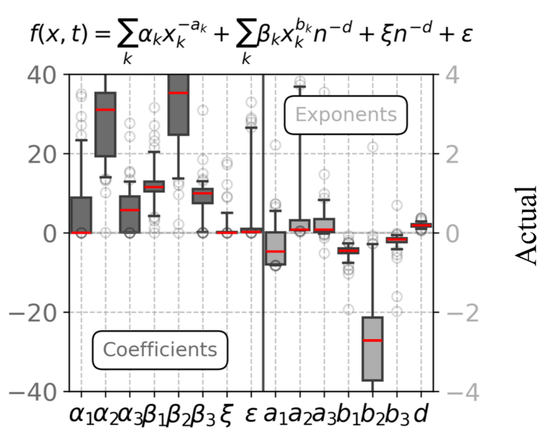
$$\min_\theta \; \Sigma_i \left(\log f(x^{(i)}, n^{(i)}; \theta) - \log f^{(i)}\right)^2$$

📌 Modeling x_k - n interaction is helpful: add−interact achieve the best fit, outperform simpler additive (add) and multiplicative (mult) power law models

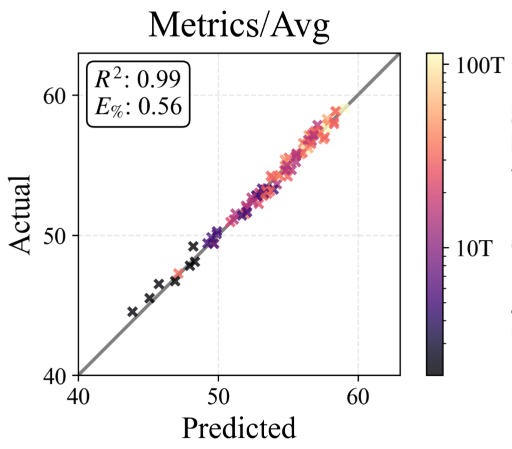
Form	Expressions for $f(x,n)$	Star+IsoFLOP CV(5-fold) MSE ↓	$E_{\%}$ ↓	R^2 ↑	Star → IsoFLOP MSE ↓	$E_{\%}$ ↓	R^2 ↑
mult	$\alpha(\prod_{k=1}^K x_k^{-a_k}) n^{-d} + \epsilon$	1.21	1.62	0.88	6.73	3.55	0.45
add	$\sum_k \alpha_k x_k^{-a_k} + \xi n^{-d} + \epsilon$	0.56	1.11	0.94	2.04	2.15	0.83
add-interact$_s$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

In-distribution (Star+IsoFLOP CV) · Extrapolation (Star → IsoFLOP)

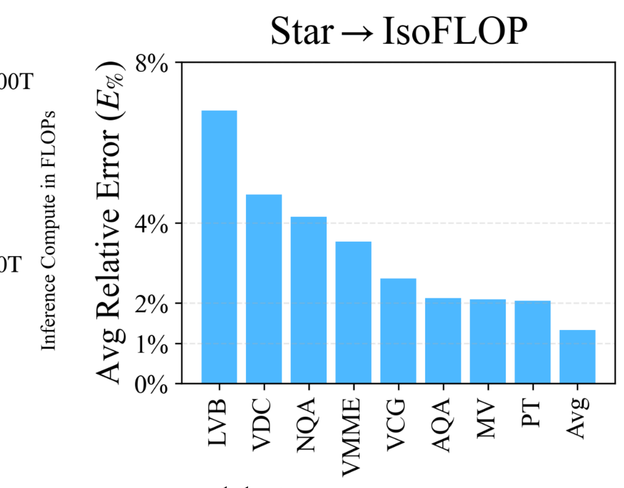
📌 Bagging add−interact provides a reasonable fit for predicting task performance



Some variability in bootstrap resampled parameter estimates due to small sample size for fitting



Bagged add−interact fits well on training data



Bagged add−interact extrapolates well for Avg performance, but struggles on LongVideoBench (LVB) & Next-QA (NQA)

Constrained Optimization

solve for the Inference compute-optimal frontier $x^*(c; n) = \mathrm{argmin}_{x: c(x) \leq c} f(x, n)$

Allocate inference compute across $x = (x_N, x_T, x_V)$ to minimize task error

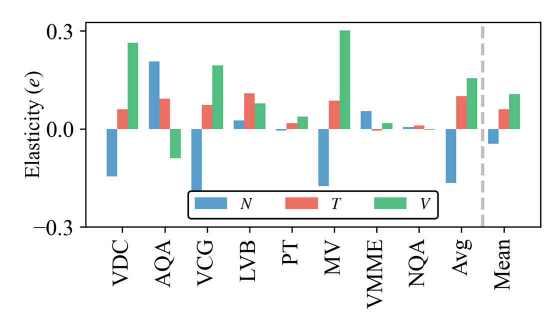
$$x^*(c; n) = \mathrm{argmin}_{x: c(x) \leq c} f(x, n)$$

Inference compute cost for both the vision encoder and LM is measured in FLOPs

$$c(x) = 2x_T(\underbrace{0.43e9}_{\text{\# vision encoder parameters}} \cdot \underbrace{768}_{\text{\# visual features}} + x_N x_V) \quad \text{for SoViT-400m/14}$$

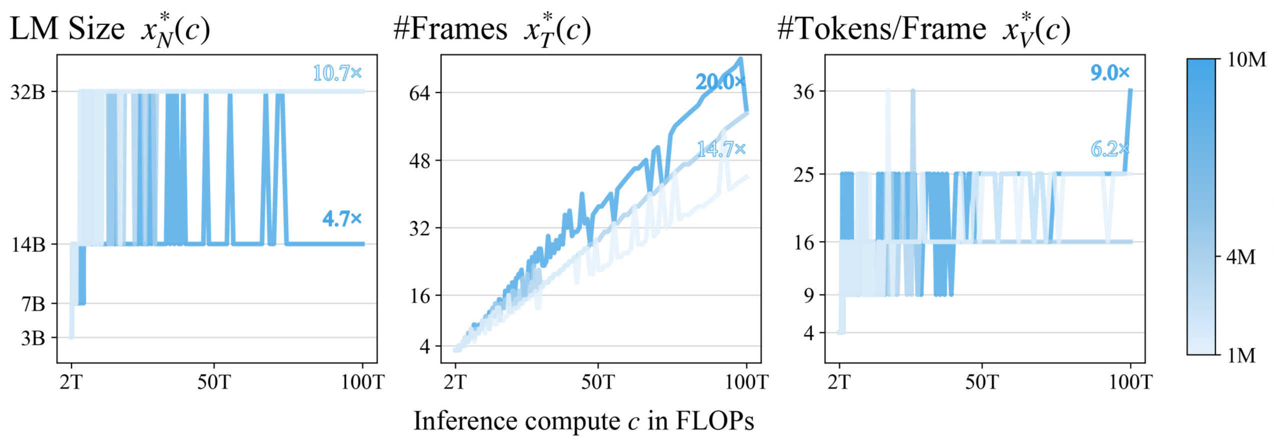
No analytical solution because ① $c(x)$ considers vision encoder's cost ② x is discrete
→ just solve with brute-force search!

📌 Inference compute-optimal x_N ↑ and x_T, x_V ↓ as data size n grows across benchmarks, with task-specific variations



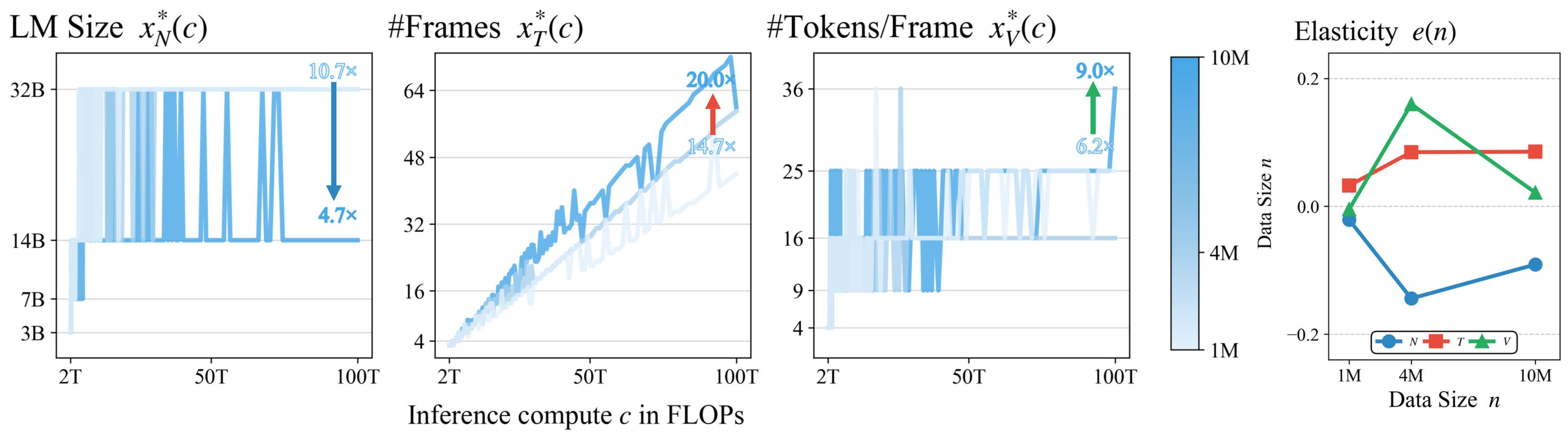
Elasticity across benchmarks

📌 the efficiency frontier $x^*(c; n)$ requires joint scaling of (x_N, x_T, x_V) at varying rates and is non-monotonic (due to x discrete)



Inference compute-optimal frontiers for vVLMs trained on 1M ($x^*(c; 1M)$) vs. 10M samples ($x^*(c; 10M)$)

📌 Inference compute-optimal x_N ↑ and x_T, x_V ↓ as data size n grows



Elasticity $e_k(c, n) = \frac{\partial x_k^*(c;n)}{\partial n} \cdot \frac{n}{x_k^*(c;n)}$ quantifies the sensitivity of $x^*(c; n)$ to changes in data size
→ $e_T > 0$ implies frontier shifts upwards