Towards Precise Scaling Laws for Video Diffusion Transformers

Supplementary Material

A. Experimental Settings and Main Results

A.1. Models

In our experiments, we employ the Cross-DiT architecture [7], an efficient diffusion transformer model that incorporates cross-attention module to integrate text conditions. This architecture is optimized for high-quality image/video generation at reduced computational costs. Our model setup includes:

- VAE[68] for encoding, PixArt-XL-2-512x512 [7] for initializing, and T5 for text encoding.
- Input sequences of 17 frames with a resolution of 256x256.

A.2. Datasets

We utilize the Panda-70M dataset [9]. A test subset of 2000 samples is randomly selected for validation.

A.3. Main Results

We summarize the key results of our video diffusion transformers. Firstly, the fitting results for the optimal hyperparameters (i.e., learning rate and batch size) across different model sizes and training tokens are:

$$B_{\text{opt}} = \alpha_B T^{\beta_B} N^{\gamma_B} \tag{15}$$

$$\eta_{opt} = \alpha_\eta T^{\beta_\eta} N^{\gamma_\eta} \tag{16}$$

Parameter for B_{opt}	α_B	β_B	γ_B
Value	2.1797×10^4	0.8080	0.1906
Parameter for η_{opt}	α_η	β_η	γ_η
Value	0.0002	-0.0453	-0.1619

Table 3. Fitting results for B_{opt} and η_{opt}

The constant term $\alpha_B = 2.1797 \times 10^4$ predicts tokens per batch. In our experiments, $\alpha_B = 17.0287$ for samples, with exponents unchanged.

Based on the optimal learning rate, we fit the validation loss for any model size and training tokens (Table 4).

$$L(T, N) = \left(\frac{T_c}{T}\right)^{\alpha_T} + \left(\frac{N_c}{N}\right)^{\alpha_N} + L_\infty \tag{17}$$

Parameter	T_{c}	α_T	N_{c}	α_N	L_∞
Value	0.0373	0.2917	0.0082	0.3188	0.4856

Table 4. Fitting results for $L(T, N)$

The optimal model size and training tokens for a fixed compute budget are given by:

$$N_{\text{opt}} = 0.8705 \cdot C^{0.4294} \tag{18}$$

$$T_{\text{opt}} = \frac{4}{3\left(7 + \frac{n_{ctx}}{d}\right)} \cdot C^{0.5706} \tag{19}$$

B. Proof of Key Results

B.1. Upper Bound of Stochastic Gradient

We approximate its gradient using the stochastic function. At iteration k, we randomly sample a mini-batch of data $\{\xi_k^{(b)}\}_{b=1}^B$ from the training data distribution ρ. Using these samples, we compute an estimated gradient g_k as:

$$g_k = \frac{1}{B}\sum_{b=1}^{B} G_{\text{est}}(\theta_k; \xi_k^{(b)}) \tag{20}$$

where $\xi_k^{(b)} \sim \rho$ represents a random data sampled from a distribution ρ at iteration k and $G_{\text{est}}(\theta_k; \xi_k^{(b)})$ is the stochastic gradient estimate for a single sample $\xi_k^{(b)}$.

Let $\mathcal{G}_k^B = \{\theta_k, \{\xi_{k-1}^{(b)}\}_{b=1}^B, \theta_{k-1}, \{\xi_{k-2}^{(b)}\}_{b=1}^B, \ldots, \theta_0\}$ represent the filtration containing all historical variables at and before iteration k. Following [40], we assume the estimated gradient is an unbiased estimate of the true gradient, while the variance of the estimated gradient is bounded:

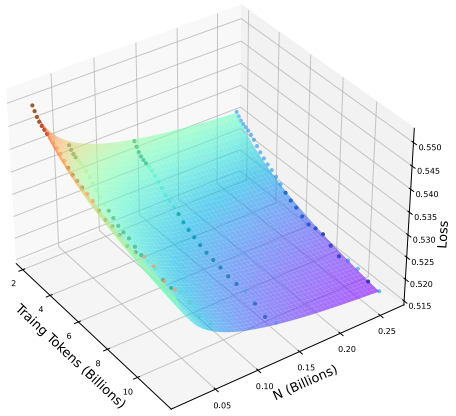
$$\mathbb{E}_{\xi_k \sim \rho}[g_k \mid \mathcal{G}_k^B] = G(\theta_k) \tag{21}$$

$$\mathbb{E}_{\xi_k \sim \rho}[\|g_k - G(\theta_k)\|^2 \mid \mathcal{G}_k^B] \le \sigma_B^2 = \frac{\sigma^2}{B} \tag{22}$$

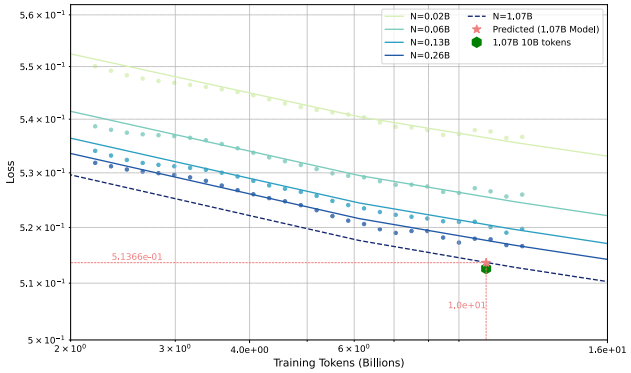
The assumptions indicates that, the stochastic gradient g_k is an unbiased estimate of $G(\theta_k)$, and the variance is bounded by σ_B^2. Using these two assumptions we get:

$$\begin{aligned}
&\mathbb{E}[\|g_k\|^2 \mid \mathcal{G}_k^B] \\
&= \mathbb{E}[\|g_k - G(\theta_k) + G(\theta_k)\|^2 \mid \mathcal{G}_k^B] \\
&= \|G(\theta_k)\|^2 + \mathbb{E}[\|g_k - G(\theta_k)\|^2 \mid \mathcal{G}_k^B] \\
&\le \|G(\theta_k)\|^2 + \sigma_B^2
\end{aligned} \tag{23}$$

where the second equality holds due to Equation (21) and the last inequality holds due to Equation (22).



(a) Performance scaling curve fitted on four small model scales.



(b) Performance scaling curve extrapolated to larger models.

Figure 8. Loss scaling with fixed suboptimal hyperparameters across varying model and compute scales. **Left**: Fitted loss curves under fixed suboptimal hyperparameters across four smaller models, each trained with varying numbers of tokens. **Right**: Extrapolated loss curves extended to larger model scales and compute budgets. The red pentagram indicates the projected loss for a 1.07B model with 10B training tokens, experimental results are shown as green hexagons.

B.2. Convergence Rate of Mini-Batch SGD

Since $L(\theta_k)$ is L-smooth ,we have

$$\begin{aligned}
&\mathbb{E}[L(\theta_{k+1}) \mid \mathcal{G}_k^B] \\
&\le L(\theta_k) + \mathbb{E}[\langle G(\theta_k), \theta_{k+1} - \theta_k \rangle \mid \mathcal{G}_k^B] \\
&\quad + \frac{L}{2}\mathbb{E}[\|\theta_{k+1} - \theta_k\|^2 \mid \mathcal{G}_k^B] \\
&= L(\theta_k) - \eta \mathbb{E}[\langle G(\theta_k), g_k \rangle \mid \mathcal{G}_k^B] \\
&\quad + \frac{L\eta^2}{2}\mathbb{E}[\|g_k\|^2 \mid \mathcal{G}_k^B] \\
&\le L(\theta_k) - \eta\left(1 - \frac{L\eta}{2}\right)\|G(\theta_k)\|^2 + \frac{L\eta^2\sigma_B^2}{2} \\
&\le L(\theta_k) - \frac{\eta}{2}\|G(\theta_k)\|^2 + \frac{L\eta^2\sigma_B^2}{2} \qquad (24)
\end{aligned}$$

By taking expectations over the filtration $\mathcal{G}_k^B$, we have

$$\mathbb{E}[L(\theta_{k+1})] \le \mathbb{E}[L(\theta_k)] - \frac{\eta}{2}\mathbb{E}[\|G(\theta_k)\|^2] + \frac{L\eta^2\sigma_B^2}{2} \qquad (25)$$

This is equivalent to

$$\mathbb{E}[\|G(\theta_k)\|^2] \le \frac{2}{\eta}\left(\mathbb{E}[L(\theta_k)] - \mathbb{E}[L(\theta_{k+1})]\right) + L\eta\sigma_B^2 \qquad (26)$$

Taking the average over $k = 0, 1, \ldots, K$, we have

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}[\|G(\theta_k)\|^2] \le \frac{2(L(\theta_0) - L^\star)}{\eta(K+1)} + L\eta\sigma_B^2 \qquad (27)$$

as required in the main text.

B.3. Stepwise Loss of Mini-Batch SGD

We approximate $G(\theta_k)$ using a batch of B samples:

$$g_k = \frac{1}{B}\sum_{b=1}^{B} G_{\text{est}}(\theta_k, \xi_k^{(b)}) \quad , \xi_k^{(b)} \sim \rho \qquad (28)$$

Following [40], the estimated gradient is unbiased, and its variance decreases inversely with the batch size B.

With the Hessian matrix H_k representing the second derivatives of $L(\theta_k)$ with respect to θ_k, the change in loss is approximately:

$$L(\theta_k - \eta g_k) \approx L(\theta_k) - \eta G(\theta_k)^\top g_k + \frac{1}{2}\eta^2 g_k^\top H_k g_k \qquad (29)$$

To obtain a more stable estimate, we compute the expectation over multiple updates:

$$\begin{aligned}
\mathbb{E}\left[L(\theta_k - \eta g_k)\right] &\approx L(\theta_k) - \eta\|G(\theta_k)\|^2 \qquad (30) \\
&\quad + \frac{1}{2}\eta^2\left(G(\theta_k)^\top H_k G(\theta_k) + \frac{\text{tr}(H_k\Sigma_k)}{B}\right)
\end{aligned}$$

This allows us to express the expected loss change per update step as follows:

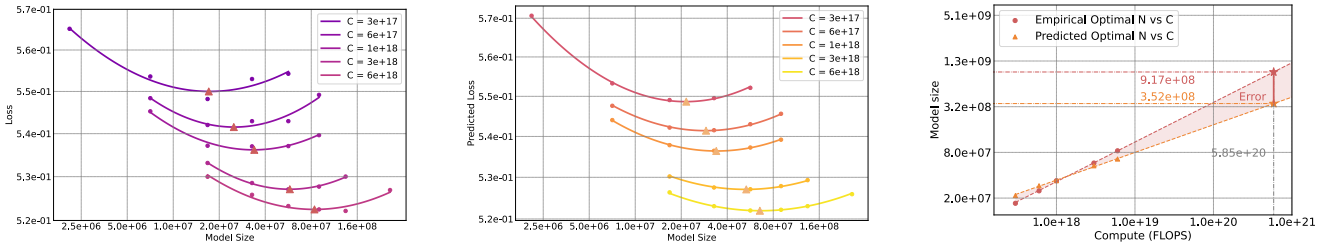
$$\begin{aligned}
\Delta L_k &= \mathbb{E}\left[L(\theta_k - \eta g_k)\right] - L(\theta_k) \qquad (31) \\
&\approx -\eta\|G(\theta_k)\|^2 + \frac{1}{2}\eta^2\left(G(\theta_k)^\top H_k G(\theta_k) + \frac{\text{tr}(H_k\Sigma_k)}{B}\right)
\end{aligned}$$

C. Experimental Conclusions with Fixed Suboptimal Hyperparameters

To demonstrate that using optimal hyperparameters can yield more accurate and robust performance predictions, we simply fixed the parameters at $B = 128$, $\eta = 2.5313 \times 10^{-4}$.

$$L(T, N) = \left(\frac{T_c}{T}\right)^{\alpha_T} + \left(\frac{N_c}{N}\right)^{\alpha_N} + L_\infty \qquad (32)$$

We used the same hyperparameters as in the fitting experiment to test the 1.07B model on 10B training tokens (Figure 8b).



(a) Empirical Loss vs. N (b) Predicted Loss vs. N (c) Optimal model scaling

Figure 9. Empirical and predicted optimal model size on fixed suboptimal batch size and learning rate. **Left**: Empirical loss as a function of model size N for various compute budgets C, with a parabolic fit to identify minimum loss points. **Middle**: Predicted loss across model sizes, using Equation (36) to predict loss for different values of N. **Right**: Optimal model scaling with compute budgets, comparing empirical results (circles) and predicted results (triangles), confirming the accuracy of Equation (36) for predicting the optimal model size.

Operation	Parameters	FLOPs
Self-Attention:QKV	$3n_{\text{layer}}d^2$	$2n_{\text{layer}}3d^2$
Self-Attention:No Mask	–	$4n_{\text{layer}}n_{ctx}d$
Self-Attention:Project	$n_{\text{layer}}d^2$	$2n_{\text{layer}}d^2$
Cross-Attention:Q	$n_{\text{layer}}d^2$	$2n_{\text{layer}}d^2$
Cross-Attention:KV	$2n_{\text{layer}}d^2$	$2n_{\text{layer}}2(n_{text}/n_{ctx})d^2$
Cross-Attention:No Mask	–	$4n_{\text{layer}}n_{text}d$
Cross-Attention:Project	$n_{\text{layer}}d^2$	$2n_{\text{layer}}d^2$
FeedForward(SwiGLU)	$n_{\text{layer}}3dd_{ff} = n_{\text{layer}}8d^2$	$16n_{\text{layer}}d^2$
Total	$16n_{\text{layer}}d^2$	$C = 3C_{forward} = 3 * (\frac{7+n_{ctx}/d}{4}N)$

Table 5. Parameter counts and compute estimates for the Cross-DiT model. The input dimensions are $f \times h \times w$, where f is the number of frames, and h and w are the height and width of each frame, respectively. To ensure consistency across models of varying sizes, we maintain proportional scaling in both model width (d) and depth (n_{layer}), with $d/n_{\text{layer}} = 128$ and the number of attention heads equal to the number of layers.

Parameter	T_c	α_T	N_c	α_N	L_∞
Value	0.0541	0.2515	0.0052	0.4101	0.4783

Table 6. Fitting results for $L(T, N)$ on fixed suboptimal hyperparameters

To evaluate the model's fitting performance on the data points, we calculate the mean squared error (MSE) Equation (33) between the fitted values and the actual data points.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{33}$$

With fixed hyperparameters, the MSE of the fitted results is 4.31×10^{-7}, while using optimal hyperparameters reduces it to 2.35×10^{-7}, a 45.5% improvement. This means the optimal hyperparameters are more effective for accurately capturing the model's power-law performance.

To explore the relationship between optimal model size and compute budget, we fixed the compute budget to get the IsoFLOPs curve from Equation (36) and validated it through experiments (Figure 9).

$$\hat{N}_{\text{opt}} = 9.5521 \cdot C^{0.3643} \tag{34}$$

$$N_{\text{opt}} = 0.0130 \cdot C^{0.5224} \tag{35}$$

Equation (34) differs from Equation (35) by an absolute error of 0.1581, or 30.26%, indicating a significant discrepancy, larger than the one observed in the fitting results under optimal hyperparameters. This discrepancy stems from the poor fit of Equation (36), particularly for lower compute budgets.

D. Parameters and Compute

Based on the Diffusion Transformer (DiT) [43], Cross-DiT architecture incorporate cross-attention modules to inject text conditions and streamline the computation-intensive class-condition branch to improve efficiency [8].

To analyze the computational complexity of the CrossDiT model, we consider the parameter counts and the number of floating-point operations (FLOPs) required for a forward pass. Table 5 summarizes the parameters and compute budget for each operation within the architecture. These operations include self-attention and cross-attention, as well as feed-forward layers im-

plemented with SwiGLU activation. The total compute cost C is derived by summing the contributions from these components. Notably, the total parameter count scales with the number of layers (n_{layer}) and the model width (d), ensuring proportional scaling across models of varying sizes. For consistency, $d/n_{\text{layer}} = 128$, and the number of attention heads is equal to n_{layer}.

E. Precise Scaling Laws in Image Generation

The image generation can be seen as a degraded version of video generation, where the video frames are reduced to a single frame. To demonstrate that our scaling law approach also applies to diffusion transformer-based image generation, we conducted experiments using the same setup as in video generation, but fixed the number of generated frames to 1 for image generation.

Following the method in main text, we fit $\eta_{opt}(N, T)$, $B_{opt}(N, T)$. The fitting results are:

Parameter for B_{opt}	α_B	β_B	γ_B
Value	5.6624×10^4	0.1495	0.0378
Parameter for η_{opt}	α_η	β_η	γ_η
Value	0.0001	-0.1868	-0.2396

Table 7. Fitting results for B_{opt} and η_{opt} for image generation

Then, based on the optimal hyperparameters, we get the result of $L(N, T)$.

$$L(T, N) = \left(\frac{T_c}{T}\right)^{\alpha_T} + \left(\frac{N_c}{N}\right)^{\alpha_N} + L_\infty \tag{36}$$

Parameter	T_{c}	α_T	N_{c}	α_N	L_∞
Value	0.0235	0.4183	0.0039	0.2935	0.6183

Table 8. Fitting results for $L(T, N)$ for image generation

We extrapolated the model size to 1.07B and trained with 2B training tokens. The predicted loss was 0.6414, the actual loss was 0.6340, resulting in an error of 1.167%, which can also make accurate predictions for image generation.