Simplifying, stabilizing, and scaling continuous-time consistency models

Authors:

Cheng Lu, Yang Song

Conference:

International Conference on Learning Representations (ICLR 2025)

Presenters:

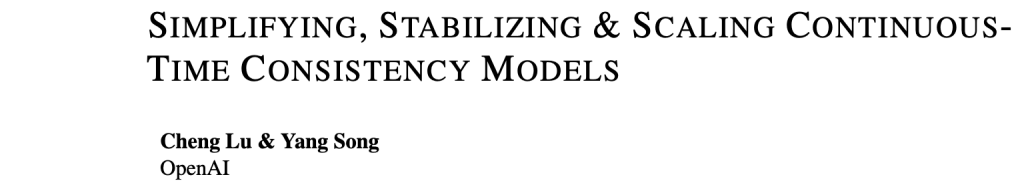
Hamidreza Gandomi, Mohammad Mohammadi

Course Instructor:

Dr. Fatemeh Seyyed Salehi

Trustworthy Generative Machine Learning Lab

Sharif University of Technology


SIMPLIFYING, STABILIZING & SCALING CONTINUOUS-TIME CONSISTENCY MODELS

Cheng Lu & Yang Song
OpenAI

Overview

- Introduction to Consistency Models (CMs)
- Diffusion Models Background
- Challenges in Continuous-Time CMs
- Key Contributions (TrigFlow, Stabilization)
- Detailed Stabilization Techniques
- Experiments and Results
- Comparisons with Diffusion Models
- Future Directions

Abstract

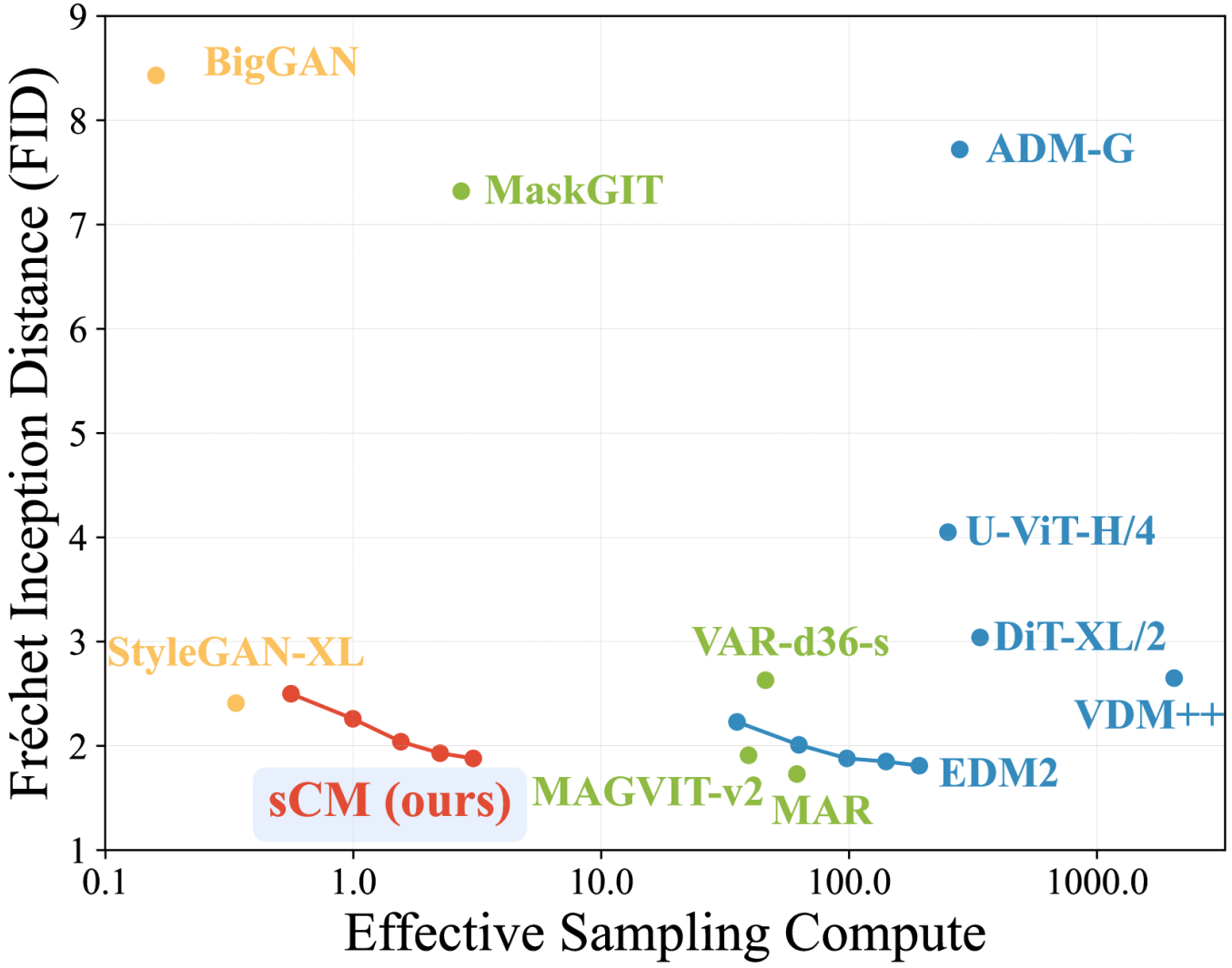
Consistency models (CMs) are diffusion-based generative models designed for fast sampling but often face issues with discretized timesteps, leading to errors and hyperparameter challenges. This work proposes a unified theoretical framework addressing instability in continuous-time formulations by improving diffusion process parameterization, network architecture, and training objectives.

Introduction to Consistency Models

- **Generative AI and Diffusion Models**: Diffusion-based generative models have revolutionized AI but face issues like slow sampling speeds.

- **Key Challenges**: Current models often require dozens or hundreds of sampling steps, limiting scalability.
- **Motivation for Consistency Models**: Consistency Models (CMs) offer a pathway for faster, scalable generative sampling.

Introduction to Consistency Models



Challenges with Diffusion Models

- **Slow Sampling Speeds**
  - Diffusion models often require 100+ steps to generate high-quality samples.
- **Discretization Errors**
  - Using discrete timesteps leads to suboptimal sample quality.
- **High Computational Costs**
  - Massive compute is needed for state-of-the-art performance.

Motivation for Consistency Models

- **Faster Sampling**

  Consistency Models reduce the steps needed to generate high-quality samples.

- **Scalable Training**

  sCMs can scale up to 1.5B parameters, maintaining performance.

- **Reduced Compute Costs**

  Efficient algorithms lower the overall computational overhead.

Consistency Models: Discrete vs Continuous-Time

- **Discrete-Time Models:**

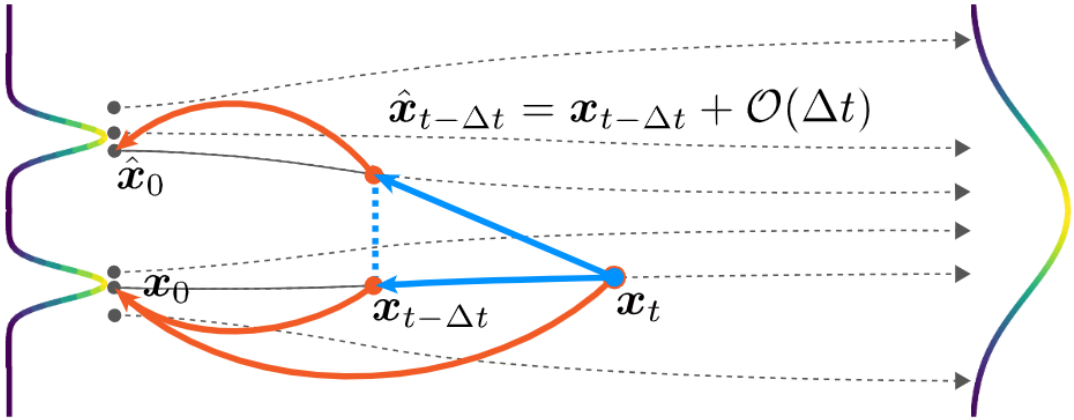
  Operate on fixed timesteps, prone to errors in sampling quality.

$$\mathbb{E}_{\mathbf{x}_t,t}\left[w(t)\,d\big(\mathbf{f}_\theta(\mathbf{x}_t,t),\mathbf{f}_{\theta^-}(\mathbf{x}_{t-\Delta t},t-\Delta t)\big)\right]$$

- **Continuous-Time Models:**

  Eliminate timestep discretization, offering smoother trajectories.

$$\nabla_\theta \mathbb{E}_{\mathbf{x}_t,t}\left[w(t)\,\mathbf{f}_\theta^\top(\mathbf{x}_t,t)\frac{\mathrm{d}\mathbf{f}_{\theta^-}(\mathbf{x}_t,t)}{\mathrm{d}t}\right]$$



Overview of Instability Issues

- **CD Method:**

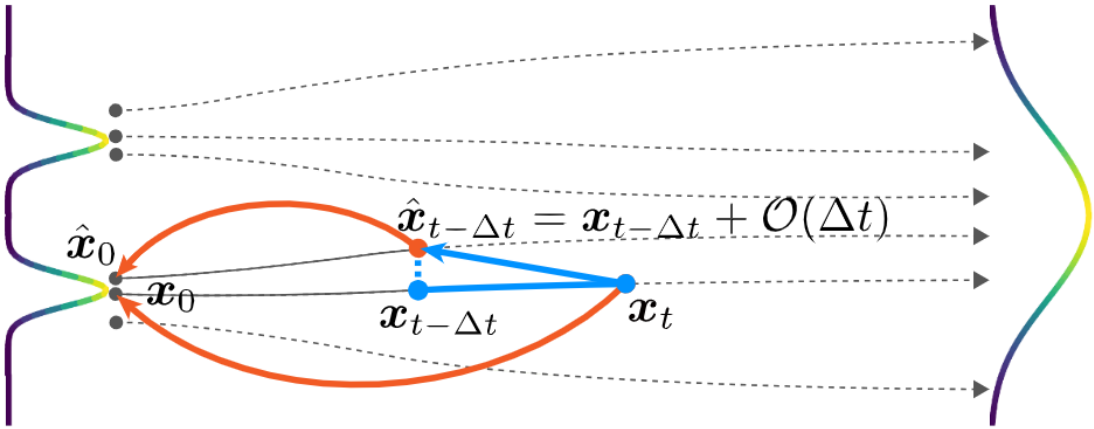
  Solving the PF-ODE to compute $x_{t-\Delta t}$ from x_t.

- **CT Method:**

  Approximating $x_{t-\Delta t}$ in discrete-time CMs

$x_{t-\Delta t} = \alpha_{t-\Delta t} x_0 + \sigma_{t-\Delta t} z$

As $\Delta t \to 0$: Leveraging time derivatives for an unbiased PF-ODE estimate.



TrigFlow Framework: Simplifying Consistency Models

Unified Formulation:

TrigFlow uses simple trigonometric coefficients such as $\cos t$ and $\sin t$. These coefficients replace the more complex relationships in the EDM method

- EDM:

$$c_{\text{skip}}(t) = \frac{\sigma_d^2}{t^2 + \sigma_d^2}, \quad c_{\text{out}}(t) = \frac{\sigma_d \cdot t}{\sqrt{\sigma_d^2 + t^2}}, \quad c_{\text{in}}(t) = \frac{1}{\sqrt{t^2 + \sigma_d^2}}$$

- TrigFlow:

$$c_{\text{skip}}(t) = \cos(t), \quad c_{\text{out}}(t) = \sin(t), \quad c_{\text{in}}(t) \equiv 1/\sigma_d$$

Therefore, Theoretical analysis is streamlined and manual adjustments are minimized.

TrigFlow Framework: Simplifying Consistency Models

Improved Time Conditioning:

- Uses $\cos t$ and $\sin t$ transformations for accurate modeling of time scales.

- Importance of Time Conditioning:
  Helps blend clean data x_0 and noise z accurately.
  Reduces discretization errors and enhances training stability.

- Role of $\cos t$ and $\sin t$:
  Directly adjust inputs to handle varying time scales effectively.

TrigFlow Framework: Simplifying Consistency Models

Enhanced Scalability:

- TrigFlow supports large-scale models with up to 1.5 billion parameters while maintaining efficiency and stability.

- Key Achievements:
  Simplified coefficients reduce manual tuning.
  Essential for real-world applications like high-resolution image or video generation.

Overview of Instability Issues

- **Problem**: Continuous-time consistency models (CMs) often face unstable training dynamics.
- **Observation**: Instability originates from high variance in gradients during training.
- **Impact**: Limits empirical performance and scalability.
- **Goal**: Propose a stable framework with simplified parameterization.

Recap of TrigFlow

- **What is TrigFlow?**
  - Simplifies EDM parameterization.
  - Uses cosine and sine functions for stability.
- **Advantages**:
  - Unified framework for diffusion models and flow matching.
  - Eliminates complex arithmetic relationships.
- **Formulation**:
  - $c_{\text{skip}}(t) = \cos(t),\ c_{\text{out}}(t) = \sin(t)$

Parameterization Improvements

- **Challenges**: Instabilities arise from complex time transformations.

$$\nabla_\theta \mathbb{E}_{\boldsymbol{x}_t,t}\left[w(t)\boldsymbol{f}_\theta^\top(\boldsymbol{x}_t,t)\frac{\mathrm{d}\boldsymbol{f}_{\theta^-}(\boldsymbol{x}_t,t)}{\mathrm{d}t}\right]$$

- This eq. depends on the tangent function, which under TrigFlow formulation is given by:

$$\frac{\mathrm{d}\boldsymbol{f}_{\theta^-}(\boldsymbol{x}_t,t)}{\mathrm{d}t} = -\cos(t)\left(\sigma_d \boldsymbol{F}_{\theta^-}\left(\frac{\boldsymbol{x}_t}{\sigma_d},t\right) - \frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t}\right) - \sin(t)\left(\boldsymbol{x}_t + \sigma_d \frac{\mathrm{d}\boldsymbol{F}_{\theta^-}\left(\frac{\boldsymbol{x}_t}{\sigma_d},t\right)}{\mathrm{d}t}\right)$$

- To stabilize training, it is necessary to ensure the tangent function is stable across different time steps.
- Empirically, it was found that instability originates from $\sin(t)\partial_t \boldsymbol{F}_{\theta^-}$ which can be decomposed according to:

$$\sin(t)\partial_t \boldsymbol{F}_{\theta^-} = \sin(t)\frac{\partial c_{\text{noise}}(t)}{\partial t}\cdot\frac{\partial \text{emb}(c_{\text{noise}})}{\partial c_{\text{noise}}}\cdot\frac{\partial \boldsymbol{F}_{\theta^-}}{\partial \text{emb}(c_{\text{noise}})}$$

Parameterization Improvements

- In that eq. emb(.) refers to the time embeddings, which is either positional embeddings or Fourier embeddings.

Parameterization Improvements

- **Challenges**: When translating EDM formulation to Trigflow formulation, the time transformation becomes $c_{\text{noise}}(t) = \log(\sigma_d \tan t)$, Straightforward derivation shows that with this C, $\sin(t) \cdot \partial_t c_{\text{noise}}(t) = 1/\cos(t)$ blows up whenever $t \to \frac{\pi}{2}$

- **Proposed Solution**: Identity time transformation $c_{\text{noise}}(t) = t$
- **Key Observations:**
  - Reduces numerical instability compared to $\log(\sigma_d \tan(t))$

Positional Time Embeddings

- **Problem**: Fourier embeddings with large scales cause oscillations.

$$\text{emb}(c) = \sin(s \cdot 2\pi\omega \cdot c + \phi)$$

- Using general time embedding above, we'll have:

$$\partial_c \text{emb}(c) = s \cdot 2\pi\omega \cos(s \cdot 2\pi\omega \cdot c + \phi)$$

- With larger Fourier scale $_s$, this derivative has greater magnitudes and oscillates more vibrantly, causing worse instability

- **Solution**: Use positional embeddings with small scales s around 0.02.
- **Key Equation**:
  - $\text{emb}(c) = \sin(0.02 \cdot c + \phi)$

Adaptive Double Normalization

- **AdaGN Layer Issues**: Adds instability to CM training.

$$\boldsymbol{y} = \text{norm}(\boldsymbol{x}) \odot \boldsymbol{s}(t) + \boldsymbol{b}(t)$$

- **Proposed Solution**:
  - Replacing AdaGN with pixel normalization has empirically showed that it retains the expressive power of AdaGN for diffusion training but removes its instability in CM training:

- $y = \text{norm}(x) \cdot \text{pnorm}(s(t)) + \text{pnorm}(b(t))$

Gradient of CT-CM training

- **Gradient Instability**: After applying all the techniques the gradient of continuous-time CM training becomes:

$$\nabla_\theta \mathbb{E}_{\boldsymbol{x}_t, t}\left[-w(t)\sigma_d \sin(t) \boldsymbol{F}_\theta^\top \left(\frac{\boldsymbol{x}_t}{\sigma_d}, t \right) \frac{\mathrm{d}\boldsymbol{f}_{\theta^-}(\boldsymbol{x}_t, t)}{\mathrm{d}t} \right]$$

- We proceed to propose additional techniques to explicitly control this gradient for improved stability!

Tangent Stability

- **Gradient Instability**: Major source is $\sin(t)\partial_t F_\theta$
- **Normalization**: Normalize the tangent function: (setting c=0.1 empirically)
  - $$\frac{df_\theta}{dt} \rightarrow \frac{df_\theta}{dt} / (||df_\theta/dt|| + c)$$
- **Alternatively**, we can clip thetangent within [-1,1], which also caps its variance.
- **Effect**: Substantial stabilization during training doing either.

Adaptive Weighting

- Weighting functions were manually designed for CM training in previous works. It was suboptimal for different data distributions and network architectures.
- We propose to train an adaptive weighting function alongside the CM.
- **Effect:**
  - Eases the burden of hyperparameter tuning
  - Outperforms manually designed weighting functions
  - Better empirical performance
  - Negligible training overhead

Adaptive Weighting

- Weighting functions were manually designed for CM training in previous works. It was suboptimal for different data distributions and network architectures.
- **Objective**:
  - Train an adaptive weighting function $w_\phi(t)$
  - Incorporate prior weighting $w(t) = \frac{1}{\sigma_d \tan(t)}$
- **Loss Function**:

$$L_{\text{sCM}}(\theta, \phi) = \mathbb{E}[e^{w_\phi(t)} ||F_\theta - F_{\theta_-}||^2 - w_\phi(t)]$$

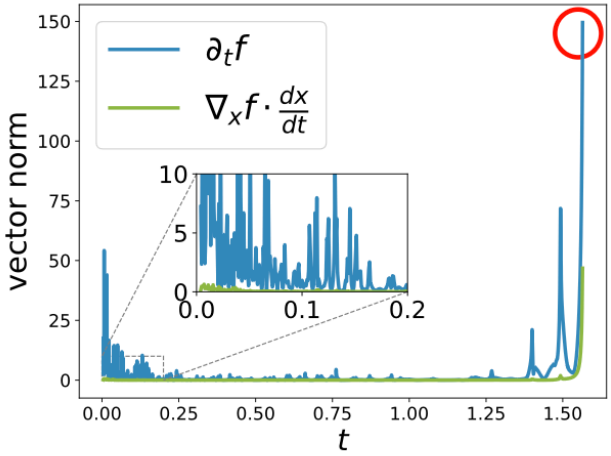
- **Effect:**
  - Eases the burden of hyperparameter tuning
  - Outperforms manually designed weighting functions
  - Better empirical performance
  - Negligible training overhead

Diffusion Finetuning and Tangent Warmup

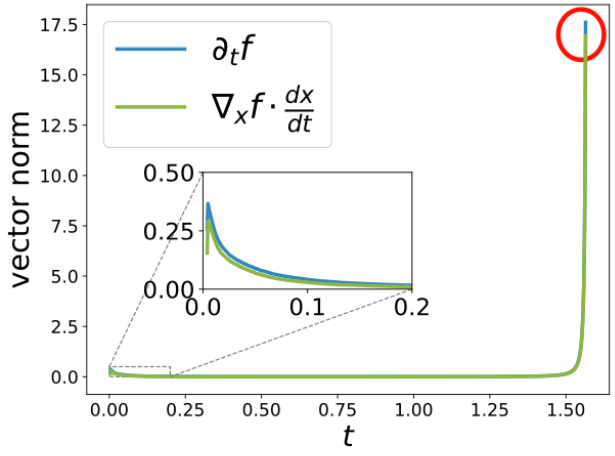
- **Diffusion Finetuning**: For consistency distillation, we find that finetuning the CM from a pretrained diffusion model can speed up convergence, which is consistent with prior works.
- **Problem:** The tangent $\frac{\mathrm{d}\boldsymbol{f}_{\theta^-}}{\mathrm{d}t}$ is decomposed into two parts:
  - The first term $\cos(t)(\sigma_d \boldsymbol{F}_{\theta^-} - \frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t})$ which is relatively stable.
  - The second term $\sin(t)(\boldsymbol{x}_t + \sigma_d \frac{\mathrm{d}\boldsymbol{F}_{\theta^-}}{\mathrm{d}t})$ which causes instability!
- **Tangent Warmup**: Gradually increase the second term with a ratio ($r \cdot \sin(t)$) over 10k iterations. (r linearly increases from 0 to 1 over the first 10k training iterations.)
- **Impact**: Improves convergence without divergence issues.

Stability Visualizations

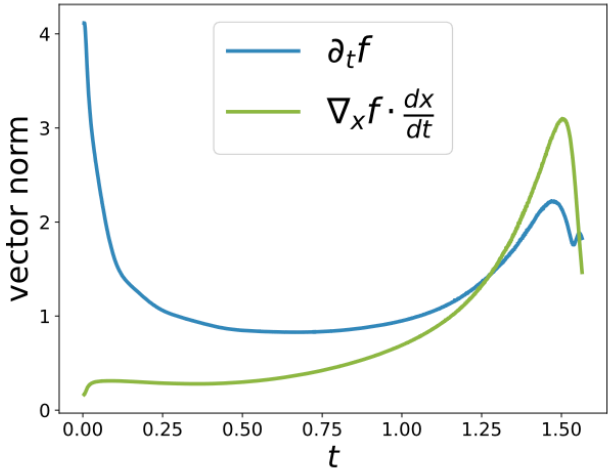
- TrigFlow maintains stability across time steps.



EDM, Fourier scale = 16.0



EDM, positional embedding

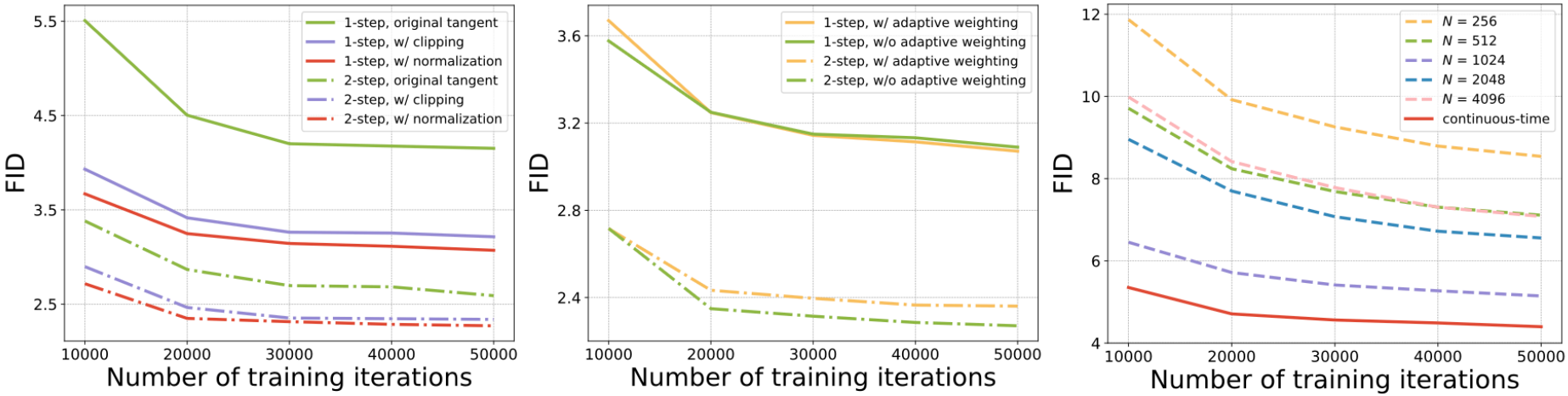


TrigFlow, positional embedding

Figure 4: **Stability of different formulations.** We show the norms of both terms in $\frac{\mathrm{d}\boldsymbol{f}_{\theta-}}{\mathrm{d}t} = \nabla_{\boldsymbol{x}} \boldsymbol{f}_{\theta-} \cdot \frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} + \partial_t \boldsymbol{f}_{\theta-}$ for diffusion models trained with the EDM ($c_{\text{noise}}(t) = \log(\sigma_d \tan(t))$) and TrigFlow ($c_{\text{noise}}(t) = t$) formulations using different time embeddings. We observe that large Fourier scales in Fourier embeddings cause instabilities. In addition, the EDM formulation suffers from numerical issues when $t \to \frac{\pi}{2}$, while TrigFlow (using positional embeddings) has stable partial derivatives for both $\boldsymbol{x}_t$ and t.

Tangent Normalization Techniques

- **Clipping**: Limit gradients to [−1, 1].
- **Effect**: Caps variance, improving stability.
- Normalization vs. Clipping:



(a) Tangent Normalization (b) Adaptive Weighting (c) Discrete vs. Continuous

Figure 5: **Comparing different training objectives for consistency distillation.** The diffusion models are EDM2 (Karras et al., 2024) pretrained on ImageNet 512×512. (a) 1-step and 2-step sampling of continuous-time CMs trained by using raw tangents $\frac{\mathrm{d}\boldsymbol{f}_{\theta^-}}{\mathrm{d}t}$, clipped tangents $\mathrm{clip}(\frac{\mathrm{d}\boldsymbol{f}_{\theta^-}}{\mathrm{d}t}, -1, 1)$ and normalized tangents $(\frac{\mathrm{d}\boldsymbol{f}_{\theta^-}}{\mathrm{d}t})/(\|\frac{\mathrm{d}\boldsymbol{f}_{\theta^-}}{\mathrm{d}t}\| + 0.1)$. (b) Quality of 1-step and 2-step samples from continuous-time CMs trained w/ and w/o adaptive weighting, both are w/ tangent normalization. (c) Quality of 1-step samples from continuous-time CMs vs. discrete-time CMs using varying number of time steps (N), trained using all techniques in Sec. 4.

Challenges in Large-Scale Models

- Implementing all the improvements proposed in previous sections by training large-scale sCMs on a variety of challenging datasets shows some cases that our model is challenged!

- The common setting for training large-scale diffusion models includes using half-precision (FP16) and Flash Attention:
  - **Memory Issues**: FP16 precision can lead to overflow in tangent computations.
  - **Efficiency**: Flash Attention lacks Jacobian-vector product (JVP) support.

Efficient Tangent Computation

- **Problem:** we empirically find that the tangent may overflow in intermediate layers when t is near 0 or π/2.
- To improve numerical precision, we propose to rearrange the computation of the tangent. We take the cosine term in objective below:

$$\mathcal{L}_{\mathrm{sCM}}(\theta,\phi) \coloneqq \mathbb{E}_{\boldsymbol{x}_t,t}\left[\frac{e^{w_\phi(t)}}{D}\left\|\boldsymbol{F}_\theta\left(\frac{\boldsymbol{x}_t}{\sigma_d},t\right)-\boldsymbol{F}_{\theta^-}\left(\frac{\boldsymbol{x}_t}{\sigma_d},t\right)-\cos(t)\frac{\mathrm{d}\boldsymbol{f}_{\theta^-}(\boldsymbol{x}_t,t)}{\mathrm{d}t}\right\|_2^2-w_\phi(t)\right]$$

- And reformulate it as $\cos(t)\sin(t)\frac{dF_\theta}{dt}$,so we can compute the JVP as:

$$\cos(t)\sin(t)\frac{\mathrm{d}\boldsymbol{F}_{\theta^-}}{\mathrm{d}t}=\left(\nabla_{\frac{\boldsymbol{x}_t}{\sigma_d}}\boldsymbol{F}_{\theta^-}\right)\cdot\left(\cos(t)\sin(t)\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t}\right)+\partial_t\boldsymbol{F}_{\theta^-}\cdot(\cos(t)\sin(t)\sigma_d)$$

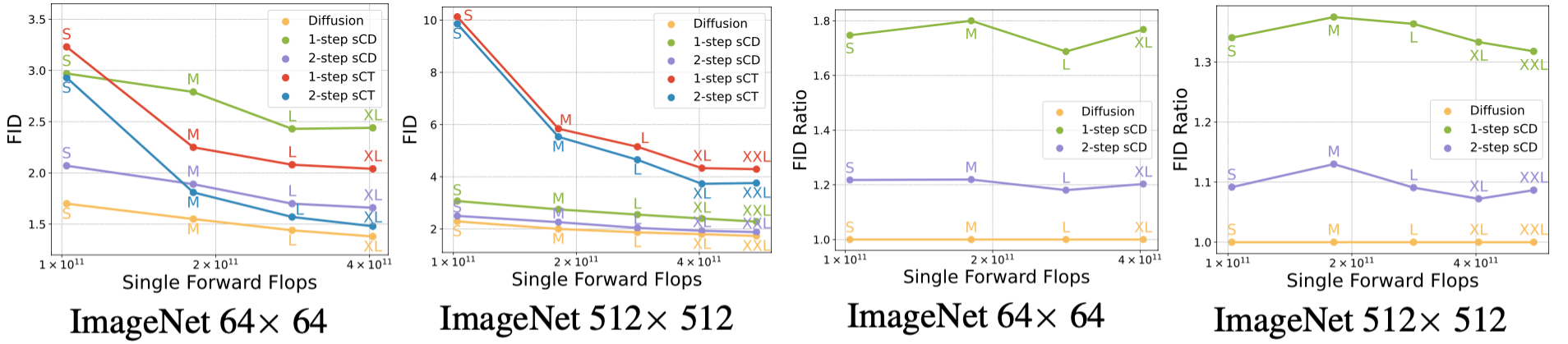
- This rearrangement greatly alleviates the overflow issues in the intermediate layers, resulting in more stable training in FP16

Flash Attention Adaptation

- **Problem**: Flash Attention is used for attention computation in large-scale model training, providing both GPU memory savings and faster training.
  - Flash Attention does not compute the JVP.
- **Proposal**:
  - Modify Flash Attention to compute JVP efficiently.
  - To fill this gap, we propose a similar algorithm that efficiently computes both softmax self-attention and its JVP in a single forward pass in the style of Flash Attention.
- **Result**: Reduced memory usage and stable gradients.

Scalability of sCD vs. sCT

- **Observation**:
  - sCD scales predictably with teacher models.
    - The effective compute per training iteration of sCD is approximately twice that of the teacher model.
    - **Result**: Two-step sCD achieves high-quality samples with <20% compute of teacher models.
  - sCT shows higher variance at larger scales.



(a) FID (↓) as a function of single forward flops. (b) FID ratio (↓) as a function of single forward flops.

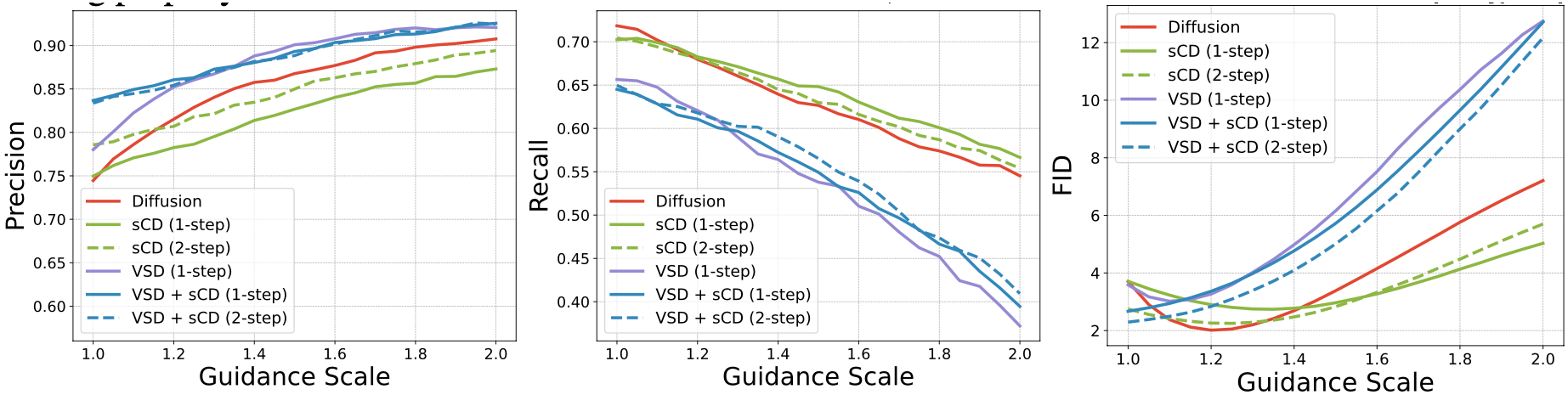
Figure 6: **sCD scales commensurately with teacher diffusion models**. We plot the (a) FID and (b) FID ratio against the teacher diffusion model (at the same model size) on ImageNet 64×64 and 512×512. sCD scales better than sCT, and has a *constant offset* in the FID ratio across all model sizes, implying that sCD has the same scaling property of the teacher diffusion model. Furthermore, the offset diminishes with more sampling steps.

Benchmarks for CIFAR-10 and ImageNet

- **sCM** outperforms all previous few-step methods that do not rely on joint training with another network and is on par with, or even exceeds, the best results achieved with adversarial training.
  - Notably, the 1-step FID of sCD-XXL on ImageNet 512×512 surpasses that of StyleGAN-XL.
- The two-step FID of sCD-XXL outperforms all generative models except diffusion and is comparable with the best diffusion models that require 63 sequential steps!
- The two-step sCM model significantly narrows the FID gap with the teacher diffusion model to within **10%**, achieving FIDs of 2.06 on CIFAR-10 (compared to the teacher FID of 2.01), 1.48 on ImageNet 64×64 (teacher FID of 1.33), and 1.88 on ImageNet 512×512 (teacher FID of 1.73).
- Additionally, we observe that sCT is more effective at smaller scales but suffers from increased variance at larger scales, while sCD shows consistent performance across both small and large scales.

Comparison with VSD

- **Observation**: VSD struggles with diversity at high guidance scales.
- In VSD fidelity is increased while diversity is decreased! (Causing severe mode collapse)



(a) Precision score (↑) (b) Recall score (↑) (c) FID score (↓)

Figure 7: **sCD has higher diversity compared to VSD**: Sample quality comparison of the EDM2 (Karras et al., 2024) diffusion model, VSD (Wang et al., 2024; Yin et al., 2024b), sCD, and the combination of VSD and sCD, across varying guidance scales. All models are of EDM2-M size and trained on ImageNet 512×512.

Summary of TrigFlow Improvements

- S for Simplification, stability, and scalability. (sCM)
- Unified framework for diffusion and flow models.

Stability Enhancements

- Techniques like tangent normalization and adaptive weighting.

Experimental Highlights

- FID improvements across datasets.

Benchmark Comparisons

- **Result**: Narrowed FID gap to <10%.

Advantages Over GANs

- Superior sample quality and scalability.

Practical Applications

- Use cases in image generation and more.

Limitations

- Scalability of sCT at large resolutions.

Future Directions

- Extensions to 3D and video generation.

Key Takeaways

- Continuous-time CMs offer unmatched scalability.

Final Visuals

- High-quality samples from ImageNet 512×512 w/ 2-step generation.

Pseudo-code for sCM Training (Appx.)

Algorithm 1 Simplified and Stabilized Continuous-time Consistency Models (sCM).

Input: dataset $\mathcal{D}$ with std. σ_d, pretrained diffusion model $\boldsymbol{F}_{\text{pretrain}}$ with parameter θ_{pretrain}, model $\boldsymbol{F}_\theta$, weighting w_ϕ, learning rate η, proposal $(P_{\text{mean}}, P_{\text{std}})$, constant c, warmup iteration H.

Init: $\theta \leftarrow \theta_{\text{pretrain}}$, Iters $\leftarrow 0$.

repeat

$\boldsymbol{x}_0 \sim \mathcal{D}$, $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \sigma_d^2 \boldsymbol{I})$, $\tau \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$, $t \leftarrow \arctan(\frac{e^\tau}{\sigma_d})$, $\boldsymbol{x}_t \leftarrow \cos(t)\boldsymbol{x}_0 + \sin(t)\boldsymbol{z}$

$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} \leftarrow \cos(t)\boldsymbol{z} - \sin(t)\boldsymbol{x}_0$ if consistency training else $\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} \leftarrow \sigma_d \boldsymbol{F}_{\text{pretrain}}(\frac{\boldsymbol{x}_t}{\sigma_d}, t)$

$r \leftarrow \min(1, \text{Iters}/H)$ ▷ Tangent warmup

$\boldsymbol{g} \leftarrow -\cos^2(t)(\sigma_d \boldsymbol{F}_{\theta^-} - \frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t}) - r \cdot \cos(t)\sin(t)(\boldsymbol{x}_t + \sigma_d \frac{\mathrm{d}\boldsymbol{F}_{\theta^-}}{\mathrm{d}t})$ ▷ JVP rearrangement

$\boldsymbol{g} \leftarrow \boldsymbol{g}/(\|\boldsymbol{g}\| + c)$ ▷ Tangent normalization

$\mathcal{L}(\theta, \phi) \leftarrow \frac{e^{w_\phi(t)}}{D}\|\boldsymbol{F}_\theta(\frac{\boldsymbol{x}_t}{\sigma_d}, t) - \boldsymbol{F}_{\theta^-}(\frac{\boldsymbol{x}_t}{\sigma_d}, t) - \boldsymbol{g}\|_2^2 - w_\phi(t)$ ▷ Adaptive weighting

$(\theta, \phi) \leftarrow (\theta, \phi) - \eta \nabla_{\theta,\phi} \mathcal{L}(\theta, \phi)$

Iters $\leftarrow$ Iters $+ 1$

until convergence

Q&A

- Thank you for your attention. I'm happy to take questions.