DuoFlow: JVP-Free Finite-Difference Mean Flows for One-Step Image Generation

Zeming Li[1,2], Xiangyu Zhang[3], Ping Tan[1,2], and Heung-Yeung Shum[1]

[1] Hong Kong University of Science and Technology
[2] Shenzhen Loop Area Institute
[3] StepFun

Abstract. MeanFlow training commonly enforces the meanflow identity with Jacobian-vector products (JVP). This is computationally expensive and hard to scale, since the JVP path often weakens fused-kernel and compiler-level optimization benefits. We revisit JVP-free MeanFlow from an error-driven perspective. Rather than a drop-in JVP replacement, finite differencing exposes controllable error structure that directly informs algorithm design. We identify two dominant error channels: step-truncation error and trajectory-velocity error.

Guided by this decomposition, we introduce a JVP-free differential MeanFlow framework for from-scratch MeanFlow-style one-step image generation. For step-truncation error, we use stochastic signed one-sided differencing, which needs only one additional forward pass and recovers second-order truncation behavior in expectation. For trajectory-velocity error, we propose DuoFlow, which predicts mean and instantaneous velocities on the same sampled state to improve trajectory consistency in differential updates. We further introduce progressive self-bootstrapping to drive trajectory velocity toward a fixed-point-like self-consistent tendency during training. As a fast derivative-evaluation scheme, our one-forward differential estimator is between 7.17× and 34.83× faster than JVP on directional-derivative micro-benchmarks for batch sizes from 1 to 64, and reduces peak memory by a factor between 1.71× and 2.68×. On ImageNet 256×256 from-scratch training, it improves FID by 30% over a matched JVP-based MeanFlow baseline. Code will be available at here.

Keywords: One-step image generation · Finite-difference mean flows · JVP-free flows

1 Introduction

One-step generation is increasingly important for real deployment in diffusion and flow-based generative modeling [21, 29, 30, 42, 43, 47]. Compared with iterative samplers, one-step models can substantially reduce latency and inference cost [13, 33, 44, 45, 51, 53, 54]. Recent progress includes consistency-style objectives and distillation pipelines [16, 25, 31, 32, 39, 40, 44, 45, 49, 51].

Within this trend, MeanFlow introduces an identity that couples interval-wise mean velocity and instantaneous velocity, providing a principled route to one-step

transport learning [4, 14, 15]. However, practical MeanFlow supervision usually relies on Jacobian-vector products (JVP) to enforce this identity [3, 14, 18]. In modern training stacks, this derivative path is often expensive and difficult to scale, and can weaken fused-kernel execution and compiler-level optimization.

This raises a central question for from-scratch MeanFlow training: can we preserve identity-based supervision while removing JVP without sacrificing quality? Our answer is *error-driven differential training*. Rather than using finite differencing as a direct operator replacement [12, 20, 35], we use it to expose the approximation structure of identity supervision. In MeanFlow, this approximation structure has a distinctive challenge: the directional derivative is evaluated along a trajectory velocity that is itself predicted by the model, so supervision quality depends on both numerical differencing and trajectory consistency. This view reveals two dominant and coupled error channels, step-truncation error and trajectory-velocity error, which then directly guide model and target design.

Guided by this decomposition, we propose **DuoFlow**, a unified JVP-free differential MeanFlow framework for from-scratch one-step generation. To control step-truncation error, we use stochastic signed one-sided differencing, which requires only one additional forward pass and recovers second-order truncation behavior in expectation. To control trajectory-velocity error, we jointly estimate mean and instantaneous velocities on the same sampled state under different temporal interval conditions, so the two estimates remain trajectory-consistent for differential supervision. We further introduce progressive self-bootstrapping to drive trajectory velocity toward a fixed-point-like self-consistent tendency throughout training.

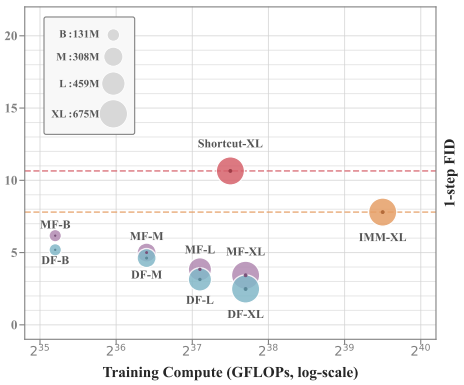


Fig. 1: DuoFlow improves FID over matched MeanFlow.

This integrated design is both efficient and effective. On directional-derivative micro-benchmarks with batch sizes from 1 to 64, the one-forward differential estimator is between **7.17**× and **34.83**× faster than JVP and reduces peak memory by a factor between **1.71**× and **2.68**×. On ImageNet 256×256 generation from scratch, it improves FID by about **30%** over a matched JVP-based MeanFlow baseline (Fig. 1).

Our contributions are threefold:

1. We formulate JVP-free differential MeanFlow training from an explicit error decomposition, showing that identity supervision in this setting is governed by two coupled errors: step-truncation error and trajectory-velocity error.
2. We propose a unified error-controlled design, combining stochastic signed one-sided differencing and same-state joint velocity estimation, to suppress the two errors within one training framework.

3. We introduce a progressive self-bootstrapping principle that drives trajectory velocity toward a fixed-point-like self-consistent tendency, further reducing trajectory-velocity error and improving training stability and final quality.

2 Related Work

From Generative Modeling to One-step MeanFlow. Modern image generation has been largely shaped by diffusion and score-based modeling, which deliver strong quality but typically require multi-step sampling [21,42,43,47]. To reduce inference latency, recent studies have moved toward one-step or few-step generation through consistency-style objectives and distillation pipelines [13, 16, 25, 31, 32, 39, 44, 45, 49, 51, 53]. In parallel, flow-based transport formulations, including Flow Matching [1, 11, 29] and Rectified Flow [30], provide principled velocity-learning views for generative dynamics [2,37]. MeanFlow [14] extends this line by enforcing an identity that couples interval-wise [4] mean velocity and instantaneous velocity during training.

Derivative Estimation for Directional Supervision. Directional-derivative supervision is classically handled by automatic differentiation, where Jacobian-vector products (JVP) are a standard primitive [3, 14, 17, 18, 31, 46]. Finite-difference estimators offer an alternative with long-established numerical analysis foundations [12,20]. More broadly, stochastic perturbation and zeroth-order optimization literature also studies bias-variance trade-offs in derivative estimation under limited evaluations [35]. These works characterize core approximation behaviors, but they are not tailored to MeanFlow identity learning, where *derivative approximation* and *trajectory-velocity quality* are tightly coupled inside the same training objective.

Positioning of This Work. Our method is positioned at the intersection of one-step MeanFlow training and practical derivative supervision. Rather than treating differencing as a direct operator replacement for JVP, we formulate JVP-free training from an *error-driven* perspective and explicitly target two coupled approximation errors, step-truncation error and trajectory-velocity error. This framing leads to a unified training design with stronger controllability of supervision and trajectory consistency, while preserving the original MeanFlow identity objective.

3 Method

MeanFlow is attractive because it shifts generation from many tiny ODE updates to interval-level transport. Yet identity supervision still relies on directional derivatives, which are commonly computed with JVP. In modern training stacks, this JVP path is computationally expensive and hard to scale.

Our premise is that removing JVP should be analysis-driven, not treated as a drop-in estimator swap. Finite differencing makes approximation errors explicit,

while JVP remains a black-box derivative transform for this objective. This view exposes two coupled error channels, step-truncation and trajectory-velocity mismatch, and explains why naive differencing can drift from the identity signal.

We therefore start from error decomposition and keep the MeanFlow identity unchanged. The decomposition directly guides three coupled design choices: a low-cost signed differential estimator, shared-network interval-conditioned velocity supervision, and progressive self-bootstrapped trajectory alignment. Together they form a JVP-free pipeline that is efficient, stable, and faithful to the identity objective.

3.1 Preliminary: From Instantaneous to Mean Velocity Fields

Flow-based generative training typically learns an instantaneous velocity field $v_\theta(z_t, t)$ along a path z_t from noise to data. With linear interpolation states,

$$z_t = (1-t)x_0 + t\epsilon, \qquad x_0 \sim p_{\text{data}},\ \epsilon \sim \mathcal{N}(0, I), \tag{1}$$

the target instantaneous velocity is $\dot{z}_t = \epsilon - x_0$. A standard objective is

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,x_0,\epsilon}\|v_\theta(z_t, t) - (\epsilon - x_0)\|^2. \tag{2}$$

This local supervision is effective, but generation still depends on multi-step integration at inference time.

MeanFlow-style training addresses this gap by learning an interval-wise mean velocity $u(z_t, r, t)$. Instead of relying only on local derivatives, the model learns a large-step transport rule constrained by

$$v(z_t, t) = u(z_t, r, t) + (t-r)\frac{d}{dt}u(z_t, r, t). \tag{3}$$

Our method keeps this identity as the central training principle and focuses on making its supervision practical without JVP.

3.2 Setup and Identity Learning Objective

Following Section 3.1, we use

$$z_t = (1-t)x_0 + t\epsilon, \qquad \dot{z}_t = \epsilon - x_0, \qquad t \in [0, 1]. \tag{4}$$

For each sample, we draw $0 \le r < t \le 1$. At the same state z_t, the network predicts

$$u_\theta(z_t, r, t)\ \text{(mean velocity)}, \qquad v_\theta(z_t, t)\ \text{(instantaneous velocity)}. \tag{5}$$

Identity consistency is enforced as

$$v \approx u + (t-r)\,\mathcal{D}^v u, \tag{6}$$

where $\mathcal{D}^v u$ denotes the directional time derivative of u along trajectory velocity v.

The key obstacle is the derivative term. JVP evaluates it accurately but introduces heavy computational and systems overhead. More importantly for method design, JVP behaves as a black-box derivative transform for this objective. Differential estimation exposes error structure, which lets us directly control approximation terms through supervision design and trajectory-consistent optimization.

3.3 Error-driven Formulation

We begin from the ideal identity-supervision derivative, $\mathcal{D}^{v^\star} u$, where $v^\star$ denotes the ideal trajectory velocity. In training, two approximations are introduced progressively. First, we replace $v^\star$ with the model trajectory velocity v_θ, which induces trajectory error:

$$\mathcal{D}^{v_\theta} u - \mathcal{D}^{v^\star} u. \tag{7}$$

Second, $\mathcal{D}^{v_\theta} u$ is not computed exactly, but approximated by a finite-difference estimator $\hat{\mathcal{D}} u$, which induces truncation error:

$$\hat{\mathcal{D}} u - \mathcal{D}^{v_\theta} u. \tag{8}$$

Combining the two steps gives the full decomposition:

$$\hat{\mathcal{D}} u - \mathcal{D}^{v^\star} u = \underbrace{\left(\hat{\mathcal{D}} u - \mathcal{D}^{v_\theta} u\right)}_{E_{\text{trunc}}} + \underbrace{\left(\mathcal{D}^{v_\theta} u - \mathcal{D}^{v^\star} u\right)}_{E_{\text{traj}}}, \tag{9}$$

where $v^\star$ is the ideal trajectory velocity for identity supervision. E_{trunc} is the step-truncation error from finite-step approximation. E_{traj} is the trajectory-velocity error caused by using an imperfect trajectory velocity in derivative evaluation.

This decomposition gives a direct design map. It is the organizing principle of our method, not a post-hoc interpretation. We reduce E_{trunc} with a low-cost stochastic differencing estimator, and reduce E_{traj} by strengthening same-state trajectory-velocity learning.

3.4 JVP-free Stochastic Signed One-sided Differencing

The first goal is to suppress truncation error without re-introducing costly derivative transforms. We therefore propose a stochastic signed one-sided estimator that needs only one extra forward evaluation. We define feasible one-sided radii

$$h_+ = \min(h_{\max}, 1 - t), \qquad h_- = \min(h_{\max}, t - r), \qquad h = \min(h_+, h_-). \tag{10}$$

In practice, $h_{\max}$ is not fixed globally. We use a precision-adaptive base step from machine epsilon:

$$h_{\text{base}} = \begin{cases} \sqrt{\varepsilon_{\text{mach}}}, & \text{for FP16/BF16,} \\ \varepsilon_{\text{mach}}^{1/3}, & \text{for FP32,} \end{cases}$$

and then apply per-sample magnitude control

$$\alpha = \min\left(1, \frac{\mathrm{RMS}(z_t) + 1}{\mathrm{RMS}(v_\theta) + \epsilon}\right), \qquad h_{\max} = \alpha\, h_{\mathrm{base}}.$$

This keeps perturbations numerically stable across training precisions and trajectory scales. Sample $\sigma \sim \mathrm{Unif}\{+1, -1\}$ and construct

$$z_{t+\sigma h} = z_t + \sigma h\, v_\theta(z_t, t), \qquad t' = t + \sigma h. \tag{11}$$

The derivative estimator is

$$\hat{\mathcal{D}}_h^\sigma u_\theta = \frac{u_\theta(z_{t+\sigma h}, r, t') - u_\theta(z_t, r, t)}{\sigma h}. \tag{12}$$

Define $\mathcal{D}^{v_\theta} u_\theta = \partial_t u_\theta + \langle v_\theta, \nabla_z u_\theta \rangle$ at (z_t, r, t). Under standard smoothness assumptions, directional Taylor expansion along $(v_\theta, 1)$ gives

$$\begin{aligned} u_\theta(z_{t+\sigma h}, r, t + \sigma h) = u_0 + \sigma h\, \mathcal{D}^{v_\theta} u_0 + \frac{h^2}{2}\, (\mathcal{D}^{v_\theta})^2 u_0 \\ + \frac{\sigma h^3}{6}\, (\mathcal{D}^{v_\theta})^3 u_0 + \mathcal{O}(h^4), \end{aligned} \tag{13}$$

where $u_0 = u_\theta(z_t, r, t)$. Substituting into Eq. (12),

$$\hat{\mathcal{D}}_h^\sigma u_\theta = \mathcal{D}^{v_\theta} u_0 + \sigma \frac{h}{2} (\mathcal{D}^{v_\theta})^2 u_0 + \frac{h^2}{6} (\mathcal{D}^{v_\theta})^3 u_0 + \mathcal{O}(h^3). \tag{14}$$

Taking expectation over $\sigma \sim \mathrm{Unif}\{\pm 1\}$ cancels the $\mathcal{O}(h)$ term:

$$\mathbb{E}_\sigma\left[\hat{\mathcal{D}}_h^\sigma u_\theta\right] = \mathcal{D}^{v_\theta} u_\theta + \mathcal{O}(h^2). \tag{15}$$

Hence the estimator achieves second-order truncation behavior in expectation with only one extra forward evaluation. For comparison, deterministic one-sided differencing

$$\hat{\mathcal{D}}_h^{\mathrm{det}} u_\theta = \frac{u_\theta(z_{t+h}, r, t + h) - u_\theta(z_t, r, t)}{h}$$

has expansion

$$\hat{\mathcal{D}}_h^{\mathrm{det}} u_\theta = \mathcal{D}^{v_\theta} u_\theta + \frac{h}{2} (\mathcal{D}^{v_\theta})^2 u_\theta + \mathcal{O}(h^2),$$

which has lower per-sample variance (no sign randomness) but a systematic $\mathcal{O}(h)$ bias. By contrast, stochastic signed one-sided differencing removes this first-order bias in expectation and introduces a zero-mean stochastic term. In long-horizon optimization, coherent bias tends to accumulate across updates, while zero-mean variance is partially averaged out by mini-batch training. This motivates our use of signed one-sided differencing: deterministic one-sided can be favorable in early variance-sensitive stages, but signed differencing is typically preferable in later bias-sensitive stages.

3.5 DuoFlow: Interval-conditioned Joint Velocity Estimation for Trajectory Control

Reducing truncation error alone is not enough. If trajectory velocity is inaccurate, differential updates still drift from the identity path. To reduce E_{traj}, we jointly estimate u_θ and v_θ at the same sampled state with shared network parameters but different temporal interval conditions, so the two velocity estimates are constrained by shared local geometry. Using Eq. (12), define

$$\Delta_\theta = (t-r)\,\hat{\mathcal{D}}_h^\sigma u_\theta, \tag{16}$$

and identity-consistent compound velocity

$$\tilde{v}_\theta = u_\theta + \Delta_\theta. \tag{17}$$

We supervise both velocity estimates via

$$u_{\text{target}} = v_{\text{target}} - \text{sg}[\Delta_\theta], \qquad \mathcal{L}_u = \|u_\theta - u_{\text{target}}\|_2^2, \qquad \mathcal{L}_v = \|v_\theta - v_{\text{target}}\|_2^2, \tag{18}$$

where sg[·] is stop-gradient. Because both predictions are anchored at the same state, improving instantaneous velocity directly improves the trajectory used inside differential supervision.

3.6 Progressive Self-bootstrapping and Fixed-point-like Tendency

A second optimization tension appears over training time. Early self-targeting is noisy, but always trusting external targets keeps trajectory dynamics weakly coupled to the model itself. We resolve this with progressive self-bootstrapping. For CFG-active samples, we use

$$v_{\text{ext}} = \omega\,\dot{z}_t + (1-\omega)\,\text{sg}[v_\theta^{\text{unc}}], \tag{19}$$

and for non-CFG samples we use $v_{\text{ext}} = \dot{z}_t$. We define

$$v_{\text{target}} = \kappa\,\text{sg}[v_\theta] + (1-\kappa)\,v_{\text{ext}}. \tag{20}$$

Instead of fixing κ, we compute it from batch-level alignment between v_θ and v_{ext} on CFG-active samples:

$$\bar{d} = \mathbb{E}_{i\in\mathcal{B}_{\text{cfg}}}[\langle v_{\theta,i}, v_{\text{ext},i}\rangle], \quad \bar{q}_v = \mathbb{E}_{i\in\mathcal{B}_{\text{cfg}}}\left[\|v_{\theta,i}\|_2^2\right], \quad \bar{q}_t = \mathbb{E}_{i\in\mathcal{B}_{\text{cfg}}}\left[\|v_{\text{ext},i}\|_2^2\right],$$
$$\kappa = 0.5 + 0.5\cdot\frac{\bar{d}}{\sqrt{\max(\bar{q}_v\bar{q}_t,\epsilon)}}. \tag{21}$$

If a mini-batch has no CFG-active samples, we use full-batch statistics as fallback. As training improves this alignment, κ rises and supervision gradually shifts toward model-consistent dynamics. This progressive alignment drives training toward a fixed-point-like, self-consistent trajectory tendency and continuously shrinks trajectory-velocity mismatch.

3.7 Unified Loss and Optimization

As κ grows, the supervision source changes, and objective imbalance can emerge. To keep optimization stable, we use detached-loss normalization:

$$\bar{\mathcal{L}}_u = \frac{\mathcal{L}_u}{\mathrm{sg}[\mathcal{L}_u] + \epsilon}, \qquad \bar{\mathcal{L}}_v = \frac{\mathcal{L}_v}{\mathrm{sg}[\mathcal{L}_v] + \epsilon}. \tag{22}$$

The final objective is

$$\mathcal{L} = \bar{\mathcal{L}}_u + w_v \, \bar{\mathcal{L}}_v, \qquad w_v = \frac{1}{(1-\bar{\kappa})^2}, \tag{23}$$

where $\bar{\kappa}$ is batch-mean κ, and we follow MeanFlow [14] in using a weighted squared ℓ_2 loss. This keeps instantaneous-velocity estimation sufficiently optimized as self-bootstrapping strengthens.

3.8 Algorithm and Complexity

Training steps. Each iteration uses two regular model evaluations. The first predicts (u_θ, v_θ) at z_t. The second evaluates the shifted state required by Eq. (12). Targets are then constructed and the unified loss is optimized. We summarize the training algorithm in Algorithm 1.

Complexity and systems compatibility. Compared with JVP-based identity training, our method stays in standard forward/backward graphs and avoids JVP transforms. This improves compatibility with fused-attention execution and compiler-level optimization. In directional-derivative micro-benchmarks for batch sizes from 1 to 64, one-forward differencing is between 7.17× and 34.83× faster than JVP and reduces peak memory by a factor between 1.71× and 2.68×.

4 Experiments

4.1 Experimental Setup

Dataset. All of our experiments are conducted on the ImageNet 256×256 dataset [7]. Following common practice in latent-space generative modeling [13, 36, 53], we operate in the latent space of a pre-trained Variational Autoencoder (VAE) tokenizer [38]. Input images are first encoded into a latent representation of size 32×32×4, and our DuoFlow is trained to generate these latents, which are then decoded back into pixel space for evaluation.

Training. All models are trained from scratch using 8 NVIDIA H800 GPUs. Unless otherwise specified, we use a total batch size of 1024 and a constant learning rate of 2e-4 [50] which is linearly warmed up from 0 during the first 10 epochs. We employ the Adam optimizer [26] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 0. The model weights are updated using an exponential moving average (EMA) with a decay rate of 0.9999. The time steps t and r are sampled independently from a logit-normal distribution [9]. We use a random dropping strategy with probability 0.1 for Classifier-Free Guidance (CFG) [22] to stabilize training.

Algorithm 1 DuoFlow: pytorch implementation of training guidance.
Note: v_u and u_Δ are within the no_grad scope.

```
# fn(z, t, r, c): function to predict mean velocity u from r to t
# x: training batch; c: class condition batch; w: CFG scale

# Sample t, r, dt(random one sided to reduce truncation error)
t, r, dt = sample_t_r_dt()
e = randn_like(x)
z = (1 - t) * x + t * e

# class conditional and unconditional velocity
v = fn(z, t, t, c)
v_u = fn(z, t, t, None)

# Use difference approximation to compute dudt
# This introduces two errors: dt(truncation) and v(trajectory)
u = fn(z, r, t, c)
u_Δ = fn(z + dt * v, r, t + dt, c)
dudt = (u_Δ - u) / dt

# Compute CFG target (same as orig MF)
v_cfg = w * (e - x) + (1 - w) * v_u

# Compute kappa, the cosine similarity between v_cfg and v
kappa = metric(v_cfg, v)
v_g = kappa * v + (1.0 - kappa) * v_cfg

# Velocity error to reduce the trajectory error
error_v = v - v_g
error_u = u - (v_g - (t - r) * stopgrad(dudt))
loss = metric(error_v)/ (1 - kappa)^2 + metric(error_u)
```

Evaluation. We evaluate the generative quality of our models using the Fréchet Inception Distance (FID) [19] score, calculated on 50,000 generated samples. For few-step and one-step generation, our primary evaluation setting uses a single Number of Function Evaluations (NFE=1) unless otherwise noted.

4.2 Main Results: One-step Generation Quality

ImageNet 256×256 Comparisons. We evaluate DuoFlow on the class-conditional ImageNet 256×256 benchmark, with results against matched from-scratch diffusion/flow baselines shown in the left panel of Tab. 1. In the single-step (1-NFE) setting, DuoFlow achieves the lowest FID among these compared from-scratch diffusion/flow methods. DuoFlow-XL/2 obtains an FID of 2.48, improving over MeanFlow-XL/2 (FID 3.43) with a relative gain of about 28%. The same trend is observed at the L/2 scale (3.14 vs. 3.84).

method	params	NFE	FID(↓)
1-NFE diffusion/flow from scratch			
iCT-XL/2 [44]	675M	1	34.24
Shortcut-XL/2 [13]	675M	1	10.60
MeanFlow-B/2 [14]	131M	1	6.17
MeanFlow-M/2	308M	1	5.01
MeanFlow-L/2	459M	1	3.84
MeanFlow-XL/2	676M	1	3.43
DuoFlow-B/2	131M	1	5.19
DuoFlow-M/2	308M	1	4.72
DuoFlow-L/2	459M	1	3.14
DuoFlow-XL/2	676M	1	**2.48**
DuoFlow-XL/2†	676M	1	**1.99**
2-NFE diffusion/flow from scratch			
iCT-XL/2 [44]†	675M	2	20.30
iMM-XL/2 [53]	675M	1×2	7.77
MeanFlow-XL/2	676M	2	2.93
DuoFlow-XL/2	676M	2	**1.96**

method	params	NFE	FID(↓)
GANs			
BigGAN [5]	112M	1	6.95
GigaGAN [24]	569M	1	3.45
StyleGAN-XL [41]	166M	1	2.30
autoregressive/masking			
AR w/ VQGAN [10]	227M	1024	26.52
MaskGIT [6]	227M	8	6.18
VAR-*d*30 [48]	2B	10×2	1.92
MAR-H [27]	943M	256×2	1.55
diffusion/flow			
ADM [8]	554M	250×2	10.94
LDM-4-G [38]	400M	250×2	3.60
SimDiff [23]	2B	512×2	2.77
DiT-XL/2 [36]	675M	250×2	2.27
SiT-XL/2 [34]	675M	250×2	2.06
+REPA [52]	675M	250×2	1.42
LightningDiT [28]	675M	250×2	**1.35**

Table 1: Class-conditional generation on ImageNet 256×256. Left: We compare few-step diffusion and flow models trained from scratch. All results are reported with Classifier-Free Guidance (CFG), where applicable; 1×2 denotes that CFG effectively doubles the NFE. Our DuoFlow models were trained for 240 epochs. For the 2-NFE evaluation, the DuoFlow-XL/2 model uses a modified training configuration to ensure stability. Competitor results are primarily sourced from MeanFlow [14]. † indicates that more epochs (640) are used during training. **Right**: gray entries provide contextual results from other paradigms and different NFE regimes, not direct matched baselines.

The advantage remains in the 2-NFE setting. DuoFlow-XL/2 reaches an FID of 1.96, compared with 2.93 for MeanFlow-XL/2, corresponding to a relative improvement of about 33%. These results are obtained with a fully from-scratch training pipeline, without external teacher/pretrained generative models or distillation, and using the same fixed VAE tokenizer as prior from-scratch baselines. Unless otherwise stated, all primary comparisons in this section use the 240-epoch setting; † marks extended-training results (640 epochs) reported separately. The gray right panel of Tab. 1 lists contextual results from different generation paradigms, parameter regimes, and NFE settings; we do not use them as direct matched baselines.

DuoFlow targets the derivative-supervision path in MeanFlow-family training. Its improvements are potentially complementary to techniques such as REPA [52], LightningDiT [50], and the concurrent iMF work [15], and combining them is a natural direction for future study.

To complement our quantitative results, we provide qualitative examples of class-conditional generation from our DuoFlow-XL/2 model in Fig. 2. Even with a single function evaluation, our model generates diverse and high-fidelity images with coherent global structures.

Fig. 2: 1-NFE Generation. Curated generated samples from our DuoFlow-XL/2 model(2.48 FID) with class-conditional guidance on ImageNet 256×256.

Batch	Fast mean (ms)	JVP mean (ms)	Speedup (JVP/Fast)	Fast peak mem(MB)	JVP peak mem(MB)	Mem ratio (JVP/Fast)
1	2.55	88.75	34.83	1337.2	2667.8	2.0
2	3.11	91.26	29.34	1604.2	2744.3	1.71
4	4.18	91.33	21.88	1202.2	2806.6	1.75
8	6.71	92.86	13.83	1636.4	2992.0	1.83
16	10.85	93.34	8.60	1700.2	3363.2	1.98
32	20.01	149.75	7.48	1827.8	4101.2	2.24
64	38.61	276.99	7.17	2083.1	5583.0	2.68

Table 2: Efficiency and memory of directional-derivative evaluation. Comparison between one-forward differential proxy (Fast) and eager `torch.func.jvp` (JVP) across batch sizes. Latency is reported as mean per-iteration time; speedup and memory ratio are both computed as `JVP/Fast`.

4.3 Efficiency and Scalability of Derivative Evaluation

Benchmark protocol. We first isolate directional-derivative computation and benchmark it directly. All runs use SiT-XL/2 in `torch.bfloat16` on CUDA with VAE-latent-like fake inputs of shape $(B, 4, 32, 32)$. For each batch size $B \in \{1, 2, 4, 8, 16, 32, 64\}$, we use `warmup_iters=10` and `bench_iters=50`. The fast path uses one additional forward pass as the differential-evaluation proxy, while the JVP path uses eager `torch.func.jvp` with fused attention disabled. Latency is reported as mean per-iteration time, and peak memory is measured by `torch.cuda.max_memory_allocated` over warmup plus benchmark iterations. This is a derivative-evaluation benchmark rather than a full training wall-clock measurement.

Results. As shown in Tab. 2, the one-forward differential proxy consistently outperforms JVP across all tested batch sizes. The speedup (JVP/Fast) ranges from 34.83× at batch size 1 to 7.17× at batch size 64. The peak-memory ratio

Variant	Signed Diff.	Joint Velocity	Prog. Bootstr.	FID↓
MeanFlow (JVP baseline)				14.13†
DuoFlow w/ deterministic one-sided diff.				19.06(-34.9%)
DuoFlow w/ signed one-sided diff.	✓			14.11(+0.0%)
DuoFlow + joint velocity	✓	✓		11.30(+20.0%)
DuoFlow (full)	✓	✓	✓	**9.17**(+35.1%)

Table 3: Component-wise ablation of DuoFlow on ImageNet 256×256 (1-NFE). We incrementally add each component while keeping the backbone, training budget, and evaluation protocol fixed. Without sign randomization, one-sided differencing degrades to FID 19.06; adding signed differencing recovers FID to 14.11, then joint velocity and progressive bootstrapping further improve to 11.30 and 9.17. The full model achieves a 35.1% relative FID improvement over the reproduced MeanFlow baseline. †: Reproduced MeanFlow with total batch size 1024 and JVP-based identity training.

Setting	h_{base}	RMS/α	FID↓	Cos-to-JVP↑	Stability
$0.25 \times h_{\text{base}}$ stress	0.25×	on	16.52	0.282	diverged
$0.5 \times h_{\text{base}}$	0.5×	on	10.54	0.47	mild osc.
Default	1.0×	on	9.17	0.74	stable
$5.0 \times h_{\text{base}}$	5.0×	on	9.12	0.93	stable
$10.0 \times h_{\text{base}}$	10.0×	on	9.16	0.87	stable
$50.0 \times h_{\text{base}}$ stress	50.0×	on	14.21	0.50	diverged
No RMS rescaling	1.0×	off	9.68	0.70	mild osc.

Table 4: Step-size sensitivity under the matched ImageNet ablation setup. The default, 5×, and 10× settings remain stable with similar FID, while overly small or large steps and disabling RMS rescaling degrade stability or quality.

(JVP/Fast) ranges from 1.71× to 2.68×, indicating substantial memory savings for the fast path throughout the sweep.

From operator-level gains to end-to-end training. The micro-benchmark isolates the derivative path and should not be interpreted as full training wall-clock speed. To measure the end-to-end effect, we compare matched XL/2 training runs with the MeanFlow trainer and our DuoFlow trainer under the same 8-GPU setup, batch size, BF16 precision, dataloader, and logging frequency. MeanFlow takes 0.89s/iter, while DuoFlow takes 0.35s/iter, giving a 2.54× wall-clock speedup; peak memory is reduced from 97GB to 73GB.

4.4 Ablation Study

To isolate how each design choice contributes, we run a component-wise ablation in Tab. 3 under a matched setup. We use the same training settings as in Sec. 4.1, except that the total training length is reduced from 240 epochs to 80 epochs, the learning rate is set to 4e-4, and warmup is set to 4 epochs.

Starting from a deterministic one-sided differencing variant, we observe a clear quality drop (FID 19.06), which indicates that naive one-sided differencing introduces non-negligible bias/noise in identity supervision. Enabling signed differencing reduces FID from 19.06 to 14.11, showing that sign randomization is a key ingredient for robust differential training.

κ controller	Setting	Best FID↓
No self-bootstrapping	$\kappa = 0$	11.26
Fixed	$\kappa = 0.5$	9.45
Fixed	$\kappa = 0.6$	9.36
Fixed	$\kappa = 0.7$	9.91
Fixed	$\kappa = 0.9$	35.33
Progressive	alignment-gated	**9.17**

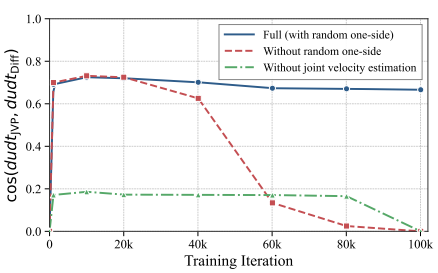
Table 5: Self-bootstrapping stability under different κ controllers. Progressive alignment-gated κ performs best in the matched ImageNet ablation setup, while fixed $\kappa = 0.9$ over-trusts self-bootstrapping and collapses.

Adding interval-conditioned joint velocity estimation yields a clear improvement (from 14.11 to 11.30), supporting the role of trajectory-velocity error control. Adding progressive self-bootstrapping further improves FID to 9.17. Overall, the three components are complementary, and the full model achieves a 35.1% relative FID improvement over the reproduced MeanFlow baseline (from 14.13 to 9.17).

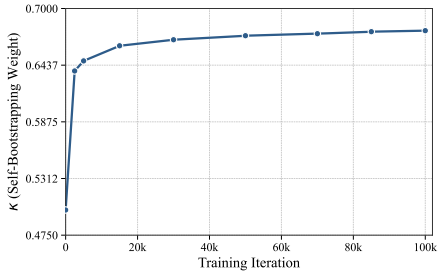
We further test the sensitivity of the finite-difference step-size policy in Tab. 4. The default, 5×, and 10× base-step settings remain stable with nearly identical FID, showing that the method is not tuned to a single fragile step value. By contrast, overly small steps, overly large stress-test steps, and removing RMS-based rescaling degrade quality or stability. Tab. 5 evaluates self-bootstrapping controllers. The progressive alignment-gated κ gives the best FID, moderate fixed κ values are slightly worse, and fixed $\kappa = 0.9$ collapses, confirming the need to avoid premature over-trust in self-bootstrapped targets.

Discussion on the deterministic one-sided differencing. We observe that deterministic one-sided finite differences sometimes lead to training divergence in the mid-stage (FID 19.06 in Tab. 3 is the best FID before divergence), while stochastic signed differences maintain stability. This is attributed to the systematic $O(h)$ truncation error of deterministic forward differences, which accumulates coherently across the batch and drives the model parameters into high-curvature regions of the loss landscape. In contrast, random-signed differences yield zero-mean $O(h^2)$ errors in expectation, allowing error self-cancellation through batch averaging. We validate this by comparing the differential errors of deterministic and stochastic signed differences with JVP-derived $\mathcal{D}^v u$ as the ground truth (as shown in Fig. 3 (a)). We find that the deterministic one-sided differencing (denoted as 'Without random one-side') has a larger differential error than the stochastic signed differences (denoted as 'Full'), which confirms our observation.

Another intriguing phenomenon observable in Fig. 3 (a) is that the deterministic one-sided differencing ('without random one-side') exhibits higher cosine similarity with the JVP reference during early training iterations compared to the stochastic signed variant ('Full'). We attribute this to a fundamental shift in the training dynamics: *early training is variance-dominated, whereas later training becomes bias-dominated*. In the initial phase, the trajectory velocity estimates suffer from substantial approximation errors, making the variance-reducing property of deterministic one-sided differencing advantageous despite its inherent $O(h)$ bias.



(a) Cosine similarity $\mathcal{D}^v u$ between JVP-based and differencing-based over training iterations. The higher the cosine similarity, the more accurate the differencing-based approximation is.



(b) κ over training iterations, showing a gradual upward trend and then saturation, which indicates a fixed-point-like trajectory-velocity tendency.

Fig. 3: Analysis of differencing accuracy and κ curriculum.

However, as training progresses and the velocity estimates become more accurate, the systematic truncation error of deterministic forward differences becomes the dominant source of inaccuracy, causing the stochastic signed approach—which achieves $O(h^2)$ error through random sign cancellation—to eventually outperform the deterministic variant.

Notably, a key advantage of the differential formulation is that it introduces an ***explicit error budget***: the derivative mismatch consists of a truncation term and a trajectory-velocity term. Because finite differencing adds truncation error, the trajectory term must be reduced more aggressively to keep overall supervision accurate. Using JVP as a truncation-free oracle, we can explicitly decompose and monitor these two components, then directly optimize the trajectory part. After truncation is stabilized, the residual mismatch is primarily trajectory-driven, enabling more targeted refinement of trajectory velocity. In a purely JVP-based pipeline, the ideal trajectory velocity is not directly observable, so this decomposition-based guidance is less accessible.

Discussion on the joint velocity estimation. Fig. 3 (a) also compares the differencing approximation quality with and without interval-conditioned joint velocity estimation. The curve labeled 'Without joint velocity estimation' shows substantially lower cosine similarity with the JVP reference compared to the 'Full' configuration across all training iterations. This indicates that when the velocity estimate $v_\theta(z_t, t)$ and the conditional field estimate $u_\theta(z_t, r, t)$ are computed independently, the finite-difference approximation $\frac{u(z_t+\sigma h v_\theta)-u(z_t)}{\sigma h}$ deviates significantly from the true directional derivative $\mathcal{D}^{v_\theta} u$ due to velocity-field mismatch. By contrast, with joint velocity estimation—where a single forward pass yields both estimates from coupled representations—the differencing-based derivative achieves much higher fidelity to the JVP ground truth, as reflected by the elevated cosine similarity.

Discussion on the progressive self-bootstrapping. Fig. 3 (b) illustrates the evolution of the self-bootstrapping weight κ throughout training. The monotonic increase in κ reflects the progressive refinement of trajectory-velocity estimation quality.

Recall that κ is computed from the cosine similarity between the CFG-guided target velocity v_{cfg} and the model's conditional velocity prediction $v_\theta(z_t, t)$—specifically, $\kappa = 0.5 + 0.5 \cdot \cos(v_{\text{cfg}}, v_\theta)$. As training progresses, the velocity estimates become increasingly accurate, driving the cosine similarity upward and thereby elevating κ. The eventual saturation of κ indicates a fixed-point-like self-consistent trajectory tendency. This stabilization is crucial for the overall MeanFlow identity system: as the self-bootstrapping mechanism matures, the compound velocity prediction $v_{\text{compound}} = u + (t-r)\partial_t u$ becomes more consistent, enabling stable long-term training dynamics.

4.5 Scaling Capabilities.

We evaluate the scaling capabilities of DuoFlow in Fig. 4. We compare the performance with different model sizes (i.e Base/Medium/Large/XLarge) and different training epochs (i.e 80/160/240). Compared with MeanFlow, DuoFlow consistently achieves lower FID, exhibiting similar scaling capabilities across different model sizes.

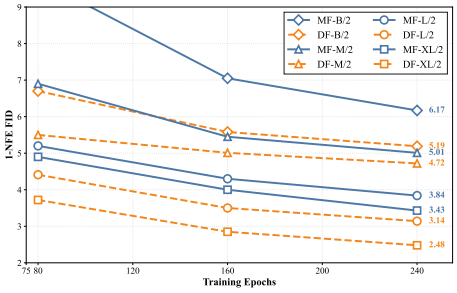


Fig. 4: DuoFlow Scaling Capabilities. We evaluate the scaling capabilities with 1-NFE on ImageNet 256×256.

5 Conclusion

We presented DuoFlow, a JVP-free differential MeanFlow framework for from-scratch MeanFlow-style one-step image generation. Our key claim is that JVP-free MeanFlow should be designed from an *error-driven* perspective rather than treated as a direct operator substitution. By analyzing differential identity supervision, we identified two coupled error channels, step-truncation error and trajectory-velocity error, and used this decomposition to guide three coupled components: stochastic signed one-sided differencing, same-state joint velocity estimation, and progressive self-bootstrapping toward a fixed-point-like tendency. This design keeps MeanFlow identity training practical in standard forward/backward pipelines while preserving supervision quality. Empirically, DuoFlow achieves substantial operator-level efficiency gains in directional-derivative evaluation and strong generative quality improvements on ImageNet 256×256 from-scratch training. Extending and validating this error-driven JVP-free training principle in text-to-image, video, and higher-resolution regimes is an important direction for future work.

Acknowledgments

This work is supported by the Shenzhen Loop Area Institute under grant FPF10120250006.

References

1. Albergo, M.S., Boffi, N.M., Vanden-Eijnden, E.: Stochastic interpolants: A unifying framework for flows and diffusions. arXiv preprint arXiv:2303.08797 (2023)
2. Albergo, M.S., Vanden-Eijnden, E.: Building normalizing flows with stochastic interpolants. In: ICLR (2023)
3. Baydin, A., Pearlmutter, B., Andreyevich, A., Siskind, J.: Automatic Differentiation in Machine Learning: a Survey. Journal of Machine Learning Research **18**(153), 1–43 (2018)
4. Boffi, N.M., Albergo, M.S., Vanden-Eijnden, E.: Flow map matching. arXiv preprint arXiv:2406.07507 (2024)
5. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: ICLR (2019)
6. Chang, H., Zhang, H., Jiang, L., Liu, C., Freeman, W.T.: Maskgit: Masked generative image transformer. In: CVPR (2022)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
8. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. NeurIPS **34** (2021)
9. Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al.: Scaling rectified flow transformers for high-resolution image synthesis. In: Forty-first international conference on machine learning (2024)
10. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: CVPR (2021)
11. Fjelde, T., Mathieu, E., Dutordoir, V.: An introduction to flow matching. `https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html` (January 2024), cambridge Machine Learning Group Blog
12. Fornberg, B.: Generation of finite difference formulas on arbitrarily spaced grids. Mathematics of computation **51**(184), 699–706 (1988)
13. Frans, K., Hafner, D., Levine, S., Abbeel, P.: One step diffusion via shortcut models. In: ICLR (2025)
14. Geng, Z., Deng, M., Bai, X., Kolter, J.Z., He, K.: Mean flows for one-step generative modeling. arXiv preprint arXiv:2505.13447 (2025)
15. Geng, Z., Lu, Y., Wu, Z., Shechtman, E., Kolter, J.Z., He, K.: Improved mean flows: On the challenges of fastforward generative models. arXiv preprint arXiv:2512.02012 (2025)
16. Geng, Z., Pokle, A., Luo, W., Lin, J., Kolter, J.Z.: Consistency models made easy. arXiv preprint arXiv:2406.14548 (2024)
17. Grathwohl, W., Chen, R.T., Bettencourt, J., Sutskever, I., Duvenaud, D.: Ffjord: Free-form continuous dynamics for scalable reversible generative models. arXiv preprint arXiv:1810.01367 (2018)
18. Griewank, A., Walther, A.: Evaluating derivatives: principles and techniques of algorithmic differentiation. SIAM (2008)
19. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. NeurIPS (2017)
20. Higham, N.J.: Accuracy and stability of numerical algorithms. SIAM (2002)
21. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. NeurIPS (2020)

22. Ho, J., Salimans, T.: Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598 (2022)
23. Hoogeboom, E., Heek, J., Salimans, T.: simple diffusion: End-to-end diffusion for high resolution images. In: ICML (2023)
24. Kang, M., Zhu, J.Y., Zhang, R., Park, J., Shechtman, E., Paris, S., Park, T.: Scaling up gans for text-to-image synthesis. In: CVPR (2023)
25. Kim, D., Lai, C.H., Liao, W.H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., Ermon, S.: Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In: ICLR (2024)
26. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
27. Li, T., Tian, Y., Li, H., Deng, M., He, K.: Autoregressive image generation without vector quantization. NeurIPS (2024)
28. Lin, S., Wang, A., Yang, X.: Sdxl-lightning: Progressive adversarial diffusion distillation. arXiv preprint arXiv:2402.13929 (2024)
29. Lipman, Y., Chen, R.T.Q., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. In: ICLR (2023)
30. Liu, X., Gong, C., qiang liu: Flow straight and fast: Learning to generate and transfer data with rectified flow. In: ICLR (2023)
31. Lu, C., Song, Y.: Simplifying, stabilizing and scaling continuous-time consistency models. In: ICLR (2025)
32. Luo, S., Tan, Y., Huang, L., Li, J., Zhao, H.: Latent consistency models: Synthesizing high-resolution images with few-step inference (2023)
33. Luo, W., Hu, T., Zhang, S., Sun, J., Li, Z., Zhang, Z.: Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. NeurIPS (2024)
34. Ma, N., Goldstein, M., Albergo, M.S., Boffi, N.M., Vanden-Eijnden, E., Xie, S.: Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In: ECCV (2024)
35. Nesterov, Y., Spokoiny, V.: Random gradient-free minimization of convex functions. Foundations of Computational Mathematics **17**(2), 527–566 (2017)
36. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: CVPR (2023)
37. Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: ICML (2015)
38. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2021)
39. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. In: ICLR (2022)
40. Sauer, A., Lorenz, D., Blattmann, A., Rombach, R.: Adversarial diffusion distillation. In: ECCV (2024)
41. Sauer, A., Schwarz, K., Geiger, A.: Stylegan-xl: Scaling stylegan to large diverse datasets (2022)
42. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: ICML (2015)
43. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: ICLR (2021)
44. Song, Y., Dhariwal, P.: Improved techniques for training consistency models. In: ICLR (2024)
45. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models. In: ICML (2023)
46. Song, Y., Garg, S., Shi, J., Ermon, S.: Sliced score matching: A scalable approach to density and score estimation. In: Uncertainty in artificial intelligence. pp. 574–584. PMLR (2020)

47. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: ICLR (2021)
48. Tian, K., Jiang, Y., Yuan, Z., Peng, B., Wang, L.: Visual autoregressive modeling: Scalable image generation via next-scale prediction. NeurIPS (2024)
49. Yang, L., Zhang, Z., Zhang, Z., Liu, X., Xu, M., Zhang, W., Meng, C., Ermon, S., Cui, B.: Consistency flow matching: Defining straight flows with velocity consistency. arXiv preprint arXiv:2407.02398 (2024)
50. Yao, J., Yang, B., Wang, X.: Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 15703–15712 (2025)
51. Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W.T., Park, T.: One-step diffusion with distribution matching distillation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 6613–6623 (2024)
52. Yu, S., Kwak, S., Jang, H., Jeong, J., Huang, J., Shin, J., Xie, S.: Representation alignment for generation: Training diffusion transformers is easier than you think. In: ICLR (2025)
53. Zhou, L., Ermon, S., Song, J.: Inductive moment matching. arXiv preprint arXiv:2503.07565 (2025)
54. Zhou, M., Zheng, H., Wang, Z., Yin, M., Huang, H.: Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In: ICML (2024)