Understanding, Accelerating, and Improving MeanFlow Training

Supplementary Material

A. Setups for Observational Study

In this section, we provide additional details for the observational study presented in Sec. 4.

Impact of v-learning on u-learning. For the v-pretraining stage, we set the flow matching ratio (FM ratio) as FM ratio $= 100\%$, *i.e.*, we always sample transitions with $t = r$. Except for the FM ratio (the probability of sampling $t = r$) and the number of pretraining epochs, all hyperparameters (optimizer, learning rate schedule, batch size, network architecture, and regularization) are kept identical to those used in the main training in Sec. 6.

Impact of u-learning on v-learning. For the training setup where we explicitly control the temporal gap Δt, we set the sampling ratio of $t = r$ to zero. Given a randomly sampled (t, r), we first compute the raw gap $\Delta t_{\text{raw}} = t - r$ and then rescale Δt_{raw} into a prescribed target range. Using the rescaled gap $\Delta\tilde{t}$ together with the originally sampled t, we redefine the reward time as $r := t - \Delta\tilde{t}$.

Task affinity analysis. The task affinity score is defined as the cosine similarity between the gradients of the two tasks across training iterations. Concretely, at the end of each training epoch, we randomly sample 5K data points and compute the gradients of the v-loss and the u-loss with 4 intervals described in Fig 5, respectively. We then compute the gradient cosine similarity within each temporal interval of the losses and average these values over all samples and training epochs to obtain the final task-affinity score.

B. Detailed Integration into MeanFlow

Recap of MeanFlow. To recall, the MeanFlow [18] objective in Eq. 4 reduces to standard flow matching by learning the instantaneous velocity v if $t = r$. It can be decomposed into two components, one for the average velocity $\mathcal{L}_u(z_t, r, t)$ and the other for the instantaneous velocity $\mathcal{L}_v(z_t, t)$:

$$\mathcal{L}_{\text{MF}} = \mathbb{E}_{x,\epsilon,t,r}\big[\mathcal{L}_u(z_t, r, t)\,\mathbb{I}(t \neq r) + \mathcal{L}_v(z_t, t)\,\mathbb{I}(t = r)\big], \tag{6}$$

where $\mathbb{I}(\cdot)$ is the indicator function, $\mathcal{L}_u$ is applied only when t does not coincide with r, and $\mathcal{L}_v$ is applied when $t = r$.

Furthermore, MeanFlow introduces additional techniques for improving its training dynamics. Among these techniques, *adaptive loss weighting* deserves particular attention, especially since our method also employs specific weighting strategies. This mechanism rescales the loss according to the magnitude of the regression error. Specifically, let e denote the regression error vector, defined as $u_\theta(z_t, t, t) - v_t(z_t|\epsilon)$ for the instantaneous velocity term $\mathcal{L}_v$, and $u_\theta(z_t, r, t) - \text{sg}(u_{\text{tgt}})$ for the average velocity term $\mathcal{L}_u$. MeanFlow calculates an adaptive weight w_{adp} to normalize the loss scale:

$$w_{\text{adp}} = \text{sg}\left(\frac{1}{(\|e\|_2^2 + c)^p}\right), \tag{7}$$

where c is a small constant for numerical stability and p controls the normalization strength. The final per-sample loss is then computed by multiplying this adaptive weight by the squared L_2 norm of the regression error. Empirically, setting $p = 1$ in MeanFlow has been demonstrated to work effectively. Therefore, we adopt this setting in our loss formulation.

Integration of our method. Let $\mathcal{L}_v^{\text{adp}}(z_t, t)$ and $\mathcal{L}_u^{\text{adp}}(z_t, r, t)$ denote the per-sample loss terms for the instantaneous and average velocities, respectively. The adaptive normalization would be ineffective if our proposed weighting schedules—specifically $\alpha(t)$ for velocity learning and $\beta(\Delta t, s)$ for progressive training— were applied directly to the raw losses. Therefore, we strictly apply these weightings *after* the adaptive normalization.

In particular, when we integrate our method with a loss-weighting method (*e.g.*, MinSNR [25]), the objective function is formulated as:

$$\begin{aligned}\mathcal{L}_{\text{ours}}^{\text{weighting}} = \mathbb{E}_{x,\epsilon,t,r}\big[&\beta(\Delta t, s)\cdot\mathcal{L}_u^{\text{adp}}(z_t, r, t)\cdot\mathbb{I}(t \neq r)\\ &+ \alpha(t)\cdot\mathcal{L}_v^{\text{adp}}(z_t, t)\cdot\mathbb{I}(t = r)\big].\end{aligned} \tag{8}$$

Alternatively, when integrating timestep sampling techniques (*e.g.*, DTD [35]), where the sampling distribution is modified to $p_{\text{acc}}(t)$ instead of applying explicit loss weighting on v, the objective becomes:

$$\begin{aligned}\mathcal{L}_{\text{ours}}^{\text{sampling}} = \mathbb{E}_{x,\epsilon,t\sim p_{\text{acc}}(t),r}\big[&\beta(\Delta t, s)\cdot\mathcal{L}_u^{\text{adp}}(z_t, r, t)\cdot\mathbb{I}(t \neq r)\\ &+ \mathcal{L}_v^{\text{adp}}(z_t, t)\cdot\mathbb{I}(t = r)\big].\end{aligned} \tag{9}$$

C. Additional Experimental Results

C.1. Observational Study on FFHQ Dataset

While our primary analysis was conducted on ImageNet, we verify whether these findings generalize to unconditional image generation on the FFHQ dataset in this section. Figures 8–10 demonstrate that our key observations consistently hold on FFHQ.

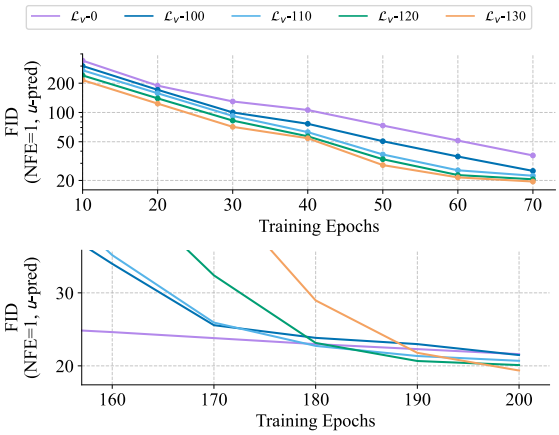

Figure 8. **v-learning facilitates u-learning.** *(Top)* 1-NFE FID during u-finetuning according to v-pretraining epochs. *(Bottom)* 1-NFE FID under a fixed 200-epoch budget with varying allocation between v-pretraining and u-finetuning. Both settings show that investing in v-learning improves u-learning quality.

Learning v facilitates learning u. Figure 8 presents results corresponding to those in Fig. 2 of the main paper. Note that the experimental settings are slightly adjusted to accommodate the different datasets. Since the number of iterations per epoch decreases significantly from ImageNet (5,004 steps) to FFHQ (273 steps), we increase the number of training epochs to compensate for the reduced training volume.

In the first setting (Top), we fix the budget for u-finetuning at 70 epochs while varying the duration of the v-pretraining across $\{0, 100, 110, 120, 130\}$ epochs. In the second setting (Bottom), we fix the total training budget at 200 epochs and allocate $\{0, 100, 110, 120, 130\}$ epochs to v-pretraining, with the remaining epochs used for u-finetuning.

Consistent with our main results, the findings clearly indicate that a heavier investment in v-pretraining yields more stable and accurate u-learning. In the fixed-budget scenario (Bottom), prioritizing v in the early stages is more effective and thereby accelerates convergence. Overall, these results reinforce the conclusion from the main paper that a well-formed instantaneous velocity v is a necessary prerequisite for learning the average velocity u.

Corruption in v-learning disrupts u-learning. Figure 9 corresponds to Fig. 3 in the main paper. Consistent with the previous setup, we extended the training duration to 200 epochs for the FFHQ dataset. We observe trends identical to those reported in Section 4.1: even with small noise ($k = 0.03$), u-learning severely degrades. This confirms that a corrupted instantaneous velocity makes learning the average velocity substantially more difficult.

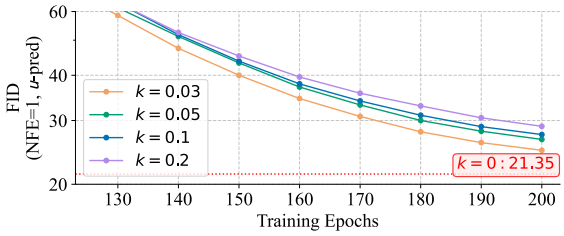

Figure 9. **Corruption in v-learning disrupts u-learning.** 1-NFE FID when training with $\mathcal{L}_{\text{MF}}$ while injecting Gaussian noise scaled by $k \cdot \|v_t(z_t|\epsilon)\|$ into the target velocity of $\mathcal{L}_v$. Even small noise ($k = 0.03$) disrupts v-learning and severely degrades u-learning performance compared to clean training ($k = 0$).

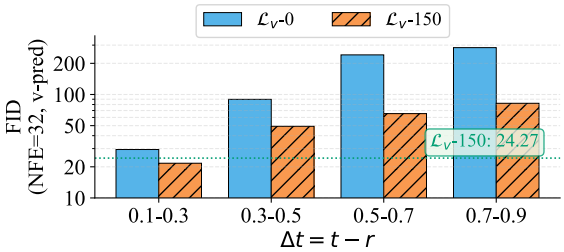

Figure 10. **Impact of Δt of u-learning on v-learning.** 32-NFE FID after 150 epochs of u finetuning across different Δt ranges, starting from either random initialization (blue) or v-pretrained model (orange, 150 epochs). Small Δt enables constructing and improving v, while large Δt degrades pretrained v. The green line denotes the performance of the v-pretrained model.

Impact of u-Learning on v-Learning. Figure 10 corresponds to Figure 4 in the main text. To adapt to the FFHQ dataset, we extended the training duration for both stages (pretraining and finetuning) from 40 to 150 epochs. Specifically, we compare models starting from either random initialization or a model pretrained with v-loss for 150 epochs, followed by 150 epochs of u-finetuning restricted to specific Δt ranges.

The findings align perfectly with those on ImageNet:

- **Small Δt supervision forms v:** Restricting u-learning to small temporal gaps ($\Delta t \in [0.1, 0.3]$) proves to be a viable proxy for v-learning, achieving 32-NFE FID scores comparable to pure v-pretraining (green line) when trained from scratch. Furthermore, it yields additional performance gains when applied to the v-pretrained model, indicating effective refinement of the instantaneous velocity.
- **Large Δt supervision deteriorates v.** In contrast, supervision with larger temporal gaps ($\Delta t > 0.3$) results in poor 32-NFE FID across both initializations. This confirms that large-Δt supervision fails to establish v when training from scratch and also disrupts an already well-formed velocity field.

Method	FID (1-NFE)↓	FID (2-NFE)↓
MeanFlow-B/2	12.90	9.81
+ MinSNR	12.82	9.65
+ DTD	12.41	9.61
+ $\mathcal{L}_u$ weighting.	12.24	9.52
+ DTD + $\mathcal{L}_u$ weighting.	**11.33**	**9.19**

Table 6. **Ablation of method components.** Velocity acceleration methods and $\mathcal{L}_u$ weighting each improve upon vanilla MeanFlow training, with their combination achieving the best performance.

C.2. Ablation of Individual Components on FFHQ dataset

To further demonstrate and analyze the effectiveness of our method, we conducted an additional ablation study on the FFHQ dataset, mirroring the experimental design of Table 3 in the main paper. For this experiment, we utilized the DiT-B/2 [51] architecture and trained all models for 400 epochs. The results are summarized in Table 6.

Consistent with the results on ImageNet, applying velocity acceleration methods alone improves upon vanilla MeanFlow training ($12.90 \rightarrow 12.41$ with DTD at 1-NFE), demonstrating that rapid formulation of v benefits training performance. However, the improvement from MinSNR is relatively marginal ($12.90 \rightarrow 12.82$). As discussed in Section 6.2, this is likely because explicit loss weighting strategies interfere with the adaptive loss weighting mechanism inherent to MeanFlow, consequently reducing robustness across different model scales. Moreover, applying only progressive weighting on $\mathcal{L}_u$ improves the performance to 12.24 at 1-NFE, demonstrating the benefit of proper temporal gap scheduling. The combined approach yields the strongest results, achieving an FID of 11.33 at 1-NFE and 9.19 at 2-NFE.