ProReflow: Progressive Reflow with Decomposed Velocity

Supplementary Material

1. Velocity Gap in Long-Range Timesteps

As Fig.1 (a) of the main paper shown, to validate the velocity discrepancy in pretrained diffusion models, we conducted experiments using the Stable Diffusion v1.5 model. The velocity at each timestep is computed as:

$$v_t = 1000 \times (x_{t+1} - x_t) \tag{1}$$

where x_t represents the latent at timestep t. We sample 100 different prompts and average their velocity matrices to obtain reliable statistics. For each pair of timesteps i and j, we compute both L2 distance $|V_i - V_j|_2$ and cosine similarity $\cos(V_i, V_j)$ between their velocities in the 4×64×64 latent space. All experiments use the PNDM scheduler with 1000 inference steps.

2. Add noise to direction or magnitude

To analyze the relative importance of velocity direction versus magnitude in the flow model, we conduct experiments using the 2-Rectified model with 10 inference steps on COCO-5K validation set. For each velocity vector v, we decompose it into direction d and magnitude m components: $v = m \cdot d$, where $|d| = 1$.

For magnitude noise, we first add Gaussian noise to m directly. Then, to ensure comparable perturbations for direction noise, we employ binary search to find an appropriate noise scale that yields the same L2 distance from the original velocity field as the magnitude noise. The directional noise is added to d and then normalized to maintain unit length. This controlled noise injection mechanism enables fair comparison between directional and magnitude perturbations, with results shown in Fig.1 (b) of the main paper.

Algorithm 1: Velocity Decomposition

```
# Add directional constraint to standard
    ↪ MSE loss
def velocity_loss(v_pred, v_target):
    # Standard MSE loss
    l_mse = mse_loss(v_pred, v_target)

    # Additional directional constraint
    l_dir = 1 - cos_similarity(v_pred,
        ↪ v_target)

    # Weight between MSE and directional
        ↪ loss
    return (1-alpha)*l_mse + alpha*l_dir
```

Algorithm 2: Progressive ReFlow

```
# K: window numbers [8,4,2]
# D: training dataset
# t: normalized time in [0,1]

# Progressive window refinement
for windows in K:
    # Training in current stage
    while not converged:
        # Get endpoints of time window
        t = sample_time() # t in [0,1]
        t1, t2 = get_window_bounds(t)

        # Compute trajectory endpoints
        z1 = add_noise(x0, t1)
        z2 = teacher_solve(z1, t1, t2)

        # Linear interpolation
        zt = interpolate(z1, z2, t)
        v_target = (z2 - z1)/(t2 - t1)

        # Update student model
        v_pred = student(zt, t)
        loss = velocity_loss(v_pred,
            ↪ v_target)
        update_params()
```

3. ProReflow Implementary Details

We have presented the pseudocode of ProReflow in Algorithm 1 in the main text. Here we elaborate on its two core components: Progressive ReFlow, which performs stage-wise training with decreasing window numbers [8,4,2], and the velocity decomposition loss which enhances directional alignment by incorporating cosine similarity alongside the standard MSE loss. The implementations are detailed in Algorithm 2 and 1, respectively.