A. Details for Figure 1

Implementation. The ground truth noisy data that is the nearest to the generated noisy data $\hat{\mathbf{x}}$ is the projected data of the generated noisy data along the velocity $\mathbf{x}_1 - \mathbf{x}_0$ for flow matching. The corresponding slowed timestep is computed as:

$$m_t = \frac{(\hat{\mathbf{x}}_t - \mathbf{x}_0)^\top (\mathbf{x}_1 - \mathbf{x}_0)}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2^2}. \tag{12}$$

We perform the ODE sampling on 20,000 training images on the SiT-B model for 50 sampling steps from $t = 0.05$ to $t = 1.0$. For each sampling step t, we collect the slowed timesteps m_t across all 20,000 images. The shaded area represents the range of slowed timesteps at each sampling timestep. At each sampling timestep, the lower point is the minimum slowed timestep, and the upper point is the maximum slowed timestep among all the 20,000 images.

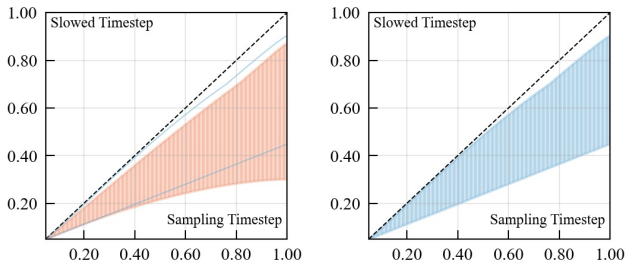
Slow Flow for diffusion with GVP. Figure A.1 shows the Slow Flow phenomenon for diffusion with generalized variance preserving interpolations the on SiT-B model [31]. Different from flow matching that uses linear interpolation, we find the nearest ground truth noisy data by checking 1000 noisy data samples for each generated noisy data and record the corresponding slowed timestep. The observation is consistent to that for flow matching in Figure 1 in the main paper.

B. Details for Figure 3

The distribution for noise x_0 is a Gaussian: $p(x_0) = \mathcal{N}(x_0; 0, 1)$. The distribution for data x_1 is a mixture of two Gaussians: $p(x_1) = 0.5\mathcal{N}(x_1; -2, 0.1^2) + 0.5\mathcal{N}(x_1; 2, 0.1^2)$.

The velocity prediction network is an MLP with 4 hidden layers, 256 hidden dimensions, and the Swish activation function. The optimizer is Adam with learning rate 0.001. The batch size is 2,048. The training iteration number is 26,000 for standard training, and 6,000 from the model trained with 20,000 iterations for MixFlow training.

We use the Euler method with 5 sampling steps. We transform the generated noisy data to a distribution, by using Kernel Density Estimation (KDE) as implemented in the SciPy library. We utilize the default hyperparameter configuration: employ a Gaussian kernel with the bandwidth determined via Scott's Rule.



(a) Standard Training (b) MixFlow Training

Figure A.1. **Slow Flow for diffusion models.** The upper and lower envelopes of the shading area in (b) are shown in (a).

C. More Results for Ablation Studies (Section 5.1)

We present more results in terms of other metrics for ablation studies.

Sampling distributions for t and m_t. Table C.2 presents the results for other metrics, including sFID, IS, Precision, and Recall. The overall observations are consistent to that for the gFID metric.

Mixture range coefficient γ in the distribution $\mathcal{U}[(1-\gamma)t, t]$ for sampling m_t. Table C.3 presents the results for sFID, IS, Precision, and Recall for $\gamma \in \{0.0, 0.3, 0.5, 0.7, 0.8, 0.9, 1.0\}$. The observations are also consistent to that for the gFID metric.

D. More Results for Improvement by MixFlow Training (Section 5.2)

Results for SDE sampling. We present the results for SDE samplers [31] in Table C.1. The evaluation settings are the same: same guidance scale (1.5), and same sampling steps (250). The sampling scheme is first-order Euler-Maruyama integrator. The observations are the same for the ODE sampler.

Input Perturbation [33]. Input Perturbation trains the diffusion model by conducting a Gaussian perturbation on the ground truth noisy data to simulate the inference time prediction errors. We implement the input perturbation algorithm by carefully tuning the perturbation strength.

Table D.1 studies the effect of perturbation strengths. The performance is sensitive to the perturbation strength: too large or too small lead to poor result. In the main paper, we report the best one for strength 0.15.

Time Shift sampler [25]. Time-Shift Sampler modifies the sampling process by adjusting the timestep for the next sampling iteration based on the previously sampled data. Table D.3 presents the results on how the two hyperparameters, window size and cutoff value, affect the performance. We report the best result in the main paper.

Epsilon Scaling sampler [34]. Epsilon Scaling adjusts the sampling process by scaling the noise prediction, mitigating the input mismatch between training and sampling. It moves the sampling trajectory closer to the vector field learned in the training phase by scaling the network output epsilon.

Table D.2 studies the effect of scaling strength $\lambda(t) = kt + b$. The performance is sensitive to the choice of (k, b). We report the best result in the main paper.

Text-to-image generation. We present the results for the sub-metrics from GenEval and T2I-CompBench. Table D.4 shows the results for T2I-CompBench across six attributes: Color, Shape, Texture, Spatial, Non-Spatial, and Complex. Table D.5 shows the results for GenEval across six categories: Single Object, Two Objects, Counting, Colors, Position, and Attribute Binding. MixFlow consistently improves the performance across all the metrics.

The high-quality T2I dataset contains 4.5M samples from public sources (Laion-5B, CC12M, Journey-DB and ShareGPT4V) with quality improved by filtering and re-captioning with Qwen2.5-VL-7B-Instruct and Blip3o, and 0.5M synthetic 1024×1024 images generated by SD3.5 and FLUX.

Table C.1. **The performance with the SDE sampler** for class-conditional generation on ImageNet 256×256 and 512×512.

Method	Generation w/o guidance					Generation w/ guidance				
	gFID ↓	sFID↓	IS ↑	Pre. ↑	Rec. ↑	gFID ↓	sFID ↓	IS ↑	Pre. ↑	Rec. ↑
ImageNet 256×256										
SiT-B/2	17.19	6.51	82.9	0.63	0.66	4.10	4.98	192.4	0.79	0.56
+ MixFlow	14.47	5.23	96.3	0.65	0.65	3.74	4.52	212.5	0.80	0.54
SiT-XL/2	8.26	6.32	131.7	0.68	0.67	2.06	4.60	270.3	0.83	0.59
+ MixFlow	7.56	4.75	144.5	0.69	0.67	1.97	4.34	276.8	0.82	0.60
ImageNet 512×512										
SiT-XL/2	9.20	7.33	125.3	0.78	0.64	2.62	4.18	252.2	0.84	0.57
+ MixFlow	8.46	5.76	139.7	0.79	0.64	2.49	4.01	268.9	0.84	0.57

Table C.2. **Studies of sampling distributions for t and m_t in the MixFlow training.**

	Generation w/o guidance					Generation w/ guidance				
	gFID ↓	sFID↓	IS ↑	Pre. ↑	Rec. ↑	gFID ↓	sFID ↓	IS ↑	Pre. ↑	Rec. ↑
$m_t \sim \mathcal{U}[0,t], t \sim \mathcal{U}[0,1]$	16.57	5.30	88.4	0.62	0.64	4.25	4.77	195.9	0.78	0.55
$m_t \sim \mathcal{U}[0,1], t \sim \mathcal{U}[0,1]$	18.27	5.74	75.8	0.62	0.66	5.07	4.89	168.9	0.77	0.56
$m_t \sim \mathcal{U}[0,t], t \sim \text{Beta}(2,1)$	15.64	5.31	90.5	0.63	0.64	3.93	4.74	201.8	0.78	0.55
$m_t \sim \mathcal{U}[0,1], t \sim \text{Beta}(2,1)$	18.25	5.73	76.3	0.62	0.65	5.06	4.88	168.8	0.77	0.57

Table C.3. **Studies of sampling distributions of m_t.** $m_t \sim \mathcal{U}[(1-\gamma)t, t]$.

	Generation w/o guidance					Generation w/ guidance				
	gFID ↓	sFID↓	IS ↑	Pre. ↑	Rec. ↑	gFID ↓	sFID ↓	IS ↑	Pre. ↑	Rec. ↑
SiT-B/2	17.97	6.43	79.4	0.62	0.66	4.46	4.93	182.4	0.78	0.57
$\gamma = 0.0$	18.00	6.45	79.1	0.62	0.66	4.48	5.01	181.1	0.78	0.57
$\gamma = 0.3$	15.75	5.28	84.1	0.63	0.65	4.09	5.04	195.7	0.79	0.56
$\gamma = 0.5$	15.72	5.20	87.8	0.63	0.65	4.00	4.80	197.2	0.79	0.56
$\gamma = 0.7$	15.65	5.22	90.2	0.63	0.66	3.95	4.76	200.6	0.79	0.56
$\gamma = 0.8$	15.64	5.21	92.0	0.63	0.66	3.91	4.76	201.7	0.79	0.56
$\gamma = 0.9$	15.64	5.24	91.0	0.63	0.66	3.93	4.71	201.6	0.79	0.56
$\gamma = 1.0$	15.64	5.31	90.5	0.63	0.64	3.93	4.74	201.8	0.78	0.55

Table D.1. **The effect of perturbation strength in DDPM-IP.** The performance is sensitive to perturbation strength.

Strength	Generation w/o guidance					Generation w/ guidance				
	gFID ↓	sFID↓	IS ↑	Pre. ↑	Rec. ↑	gFID ↓	sFID ↓	IS ↑	Pre. ↑	Rec. ↑
0.1	17.26	5.82	81.8	0.62	0.66	4.36	4.73	183.9	0.78	0.57
0.15	17.01	5.37	83.1	0.62	0.65	4.32	4.77	185.8	0.78	0.56
0.2	17.55	6.44	81.1	0.62	0.65	4.86	6.56	180.3	0.78	0.55
0.3	17.82	8.45	80.2	0.62	0.62	5.41	9.08	178.9	0.78	0.53
0.5	33.50	61.8	51.5	0.50	0.55	14.46	47.4	125.4	0.68	0.44

RAE. Figure D.1 presents the slow flow phenomenon that is observed in the RAE models. The overall observation is similar: MixFlow training makes slowed timesteps be closer to sampling timesteps, and the range for slowed timesteps be smaller. One difference from Figure 1 in the main paper and Figure A.1 is that the minimum slowed timestep is larger. For example, in Figure D.1 (a), the minimum slowed timestep around sampling timestep 1.00 is about 0.6, larger than 0.2 in Figure 1 in the main paper. This implies that the slowed timestep for RAE is possible to be closer to the sampling timestep.

Regarding the mixture range coefficient γ, we tried two choices: 0.8 that is selected for all the other experiments, and 0.4 ($= 1.00 - 0.60$) that is based on the observation in Figure D.1. Both achieve similar performance (See Table D.6 and Figure D.1). $\gamma = 0.8$ needs 430 training epochs and $\gamma = 0.4$ needs around a half, 200, training epochs. Figure D.1 (d) shows that the upper and lower

Table D.2. **The effect of scaling strength $\lambda(t) = kt + b$ in Epsilon Scaling.** The performance is sensitive to scaling strength.

	Generation w/o guidance					Generation w/ guidance				
	gFID ↓	sFID↓	IS ↑	Pre. ↑	Rec. ↑	gFID ↓	sFID ↓	IS ↑	Pre. ↑	Rec. ↑
$k = 0, b = 1.02$	18.13	6.55	78.1	0.60	0.65	4.78	5.12	179.1	0.77	0.57
$k = 0, b = 1.015$	17.87	6.41	79.1	0.62	0.66	4.49	4.94	181.6	0.78	0.56
$k = 0, b = 1.01$	17.47	6.27	79.6	0.62	0.66	4.46	4.90	182.0	0.78	0.56
$k = 0, b = 1.005$	17.28	6.28	80.3	0.62	0.66	4.40	4.80	182.4	0.78	0.56
$k = 0, b = 0.995$	18.19	6.55	77.9	0.61	0.65	4.87	5.22	178.1	0.77	0.57
$k = 0, b = 0.99$	18.87	6.87	75.01	0.60	0.65	5.07	5.43	175.1	0.77	0.57
$k = 0.0001, b = 1.005$	17.18	6.28	80.8	0.62	0.65	4.39	4.81	182.5	0.78	0.56
$k = 0.0002, b = 1.005$	17.25	6.31	80.1	0.62	0.65	4.42	4.91	182.1	0.78	0.56
$k = 0.0004, b = 1.005$	17.51	6.30	79.1	0.62	0.64	4.47	4.97	179.1	0.78	0.56

envelopes of slowed timesteps for $\gamma = 0.4$ and $\gamma = 0.8$ are almost the same. In the main paper, we report the results for $\gamma = 0.4$.

We additionally report the performance of RAE with 20 sampling steps in Table D.7. One can see that the improvement from MixFlow is larger than that for 50 sampling steps.

E. SOTA Results on ImageNet 512×512 without Guidance.

We present the comparison with state-of-the-art models on ImageNet 512×512 for generation without guidance in Table D.8. We compare with the methods that have reported performance on ImageNet 512×512 without guidance. We follow the same evaluation setting as in the main paper. MixFlow + RAE achieves the best performance across multiple metrics, including gFID, sFID, IS and Precision.

F. Improvement over Qwen-Image

We verify MixFlow on Qwen-Image (20B MMDiT), that is larger than SD 3.5 and differs in text encoder and RoPEs. The setup follows the same procedure as SD 3.5: we finetune the model for 10K steps and then post-train with MixFlow for 10K steps. A baseline model (ft-20K) is trained for 20K steps without MixFlow.

Table F.1 presents the results. MixFlow consistently improves the performance across both GenEval and DPG-Bench for both 50 and 10 sampling steps. This indicates that MixFlow is effective for larger models with different architectures.

G. Qualitative Results

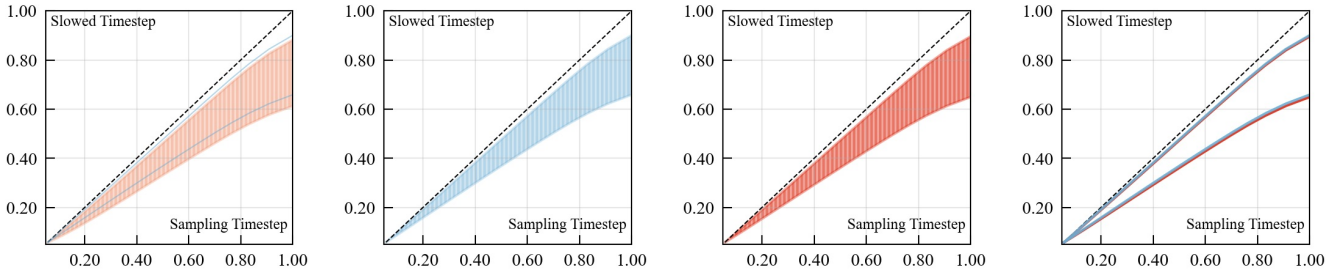
Class-conditional generation on ImageNet 256×256 and 512×512. We present the visualization results for MixFlow + RAE class-conditional generation on ImageNet 256×256 and 512×512 in Figures I.1 to I.12. The results demonstrate high semantic diversity and fine details.

H. Example Code

The MixFlow training implementation is very simple. For example, only **6 lines** of code are modified for the RAE implementation. The modification example is shown in Figure I.13.

I. Extensions

The initial results show that the MixFlow training algorithm also benefit (1) training the SiT model from scratch and training the shortcut model [5] for few-step sampling.



(a) Standard Training (b) MixFlow Training, $\gamma = 0.4$ (c) MixFlow Training with $\gamma = 0.8$ (d) Upper and lower envelopes of $\gamma = 0.8$ and $\gamma = 0.4$

Figure D.1. **Slow Flow for the RAE model.**

Table D.3. **The effect of window size and cutoff value in Time-Shift Sampler.**

	Generation w/o guidance					Generation w/ guidance				
	gFID ↓	sFID↓	IS ↑	Pre. ↑	Rec. ↑	gFID ↓	sFID ↓	IS ↑	Pre. ↑	Rec. ↑
$w = 5$, cutoff $= 500$	18.24	6.57	78.1	0.61	0.66	4.54	4.89	177.1	0.77	0.56
$w = 5$, cutoff $= 400$	18.20	6.50	78.3	0.61	0.66	4.54	4.89	178.4	0.77	0.56
$w = 5$, cutoff $= 300$	18.17	6.48	79.1	0.62	0.66	4.52	4.87	179.2	0.78	0.56
$w = 5$, cutoff $= 200$	18.14	6.44	79.5	0.62	0.66	4.50	4.82	180.9	0.78	0.56
$w = 5$, cutoff $= 100$	18.13	6.43	79.9	0.62	0.66	4.49	4.82	181.7	0.78	0.56
$w = 5$, cutoff $= 50$	18.15	6.45	79.1	0.62	0.66	4.50	4.82	180.9	0.78	0.56
$w = 2$, cutoff $= 100$	18.13	6.40	80.1	0.62	0.66	4.49	4.90	181.2	0.78	0.56
$w = 8$, cutoff $= 100$	18.12	6.41	79.5	0.62	0.66	4.49	4.80	180.5	0.78	0.56
$w = 10$, cutoff $= 100$	18.14	6.44	79.1	0.62	0.66	4.50	4.80	180.3	0.78	0.56

Table D.4. **The performance for text-to-image generation in T2I-CompBench.**

Method	Color ↑	Shape ↑	Texture ↑	Spatial ↑	Non-Spatial ↑	Complex ↑
SD 3.5	0.8135	0.6555	0.7841	0.2850	0.3129	0.4202
SD 3.5-ft-20k	0.8142	0.6556	0.7842	0.2852	0.3133	0.4208
SD 3.5-ft-10k	0.8140	0.6556	0.7839	0.2855	0.3132	0.4205
+ MixFlow	0.8612	0.7013	0.8112	0.3642	0.3402	0.4756

Table D.5. **The performance for text-to-image generation in Geneval.**

Method	Single Obj.	Two Obj.	Count-ing	Colors	Posi-tion	Attr. Bind.
SD 3.5	0.98	0.78	0.50	0.81	0.24	0.52
SD 3.5-ft-20k	0.98	0.78	0.51	0.80	0.24	0.52
SD 3.5-ft-10k	0.98	0.78	0.51	0.80	0.24	0.51
+ MixFlow	0.98	0.79	0.63	0.82	0.27	0.56

Table D.6. **The effect of mixture range coefficient γ for RAE.**

γ	Epochs	Generation w/o guidance					Generation w/ guidance				
		gFID ↓	sFID↓	IS ↑	Pre. ↑	Rec. ↑	gFID ↓	sFID ↓	IS ↑	Pre. ↑	Rec. ↑
0.4	200	1.43	4.81	239.8	0.80	0.64	1.10	4.40	259.7	0.78	0.67
0.8	200	1.57	4.95	233.5	0.80	0.63	1.21	4.60	248.9	0.78	0.65
0.8	430	1.43	4.85	239.1	0.80	0.64	1.10	4.42	259.3	0.78	0.66

Table D.7. **MixFlow on RAE.** The gain for 20 sampling steps is larger than 50 sampling steps.

Method	Generation w/o guidance					Generation w/ guidance				
	gFID ↓	sFID↓	IS ↑	Pre. ↑	Rec. ↑	gFID ↓	sFID ↓	IS ↑	Pre. ↑	Rec. ↑
50 *sampling steps*										
RAE	1.51	5.31	242.9	0.79	0.63	1.13	4.74	262.6	0.78	0.67
+ MixFlow	1.43	4.81	239.8	0.80	0.64	1.10	4.40	259.7	0.78	0.67
20 *sampling steps*										
RAE	1.92	6.32	232.8	0.78	0.64	1.35	5.37	254.4	0.77	0.67
+ MixFlow	1.64	5.39	238.7	0.80	0.63	1.18	4.77	255.8	0.78	0.66

Table D.8. **Class-conditional performance on ImageNet** 512×512 without guidance.

Method	gFID↓	sFID↓	IS↑	Prec.↑	Rec.↑
Autogressive					
MAGViT-v2 [57]	3.07	–	213.1	–	–
Diffusion					
DiT [37]	11.93	–	–	–	–
SiT [31]	9.20	7.33	125.3	0.78	**0.64**
EDM2 [20]	1.91	–	–	–	–
RAE [61]	1.61	4.96	234.1	0.81	0.63
MixFlow + RAE	**1.55**	**4.67**	**237.0**	**0.82**	0.60

Table F.1. **The performance of MixFlow over Qwen-Image.**

Method	GenEval Avg. ↑		DPG-Bench Avg. ↑	
	50 steps	10 steps	50 steps	10 steps
Qwen-Image	0.87	0.75	88.38	81.47
Qwen-Image-ft-20k	0.87	0.75	88.46	81.54
Qwen-Image-ft-10k	0.87	0.75	88.50	81.50
+ MixFlow	**0.90**	**0.86**	**90.32**	**87.65**

Figure I.1. **Uncurated** 256×256 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “Sulphur-crested cockatoo” (89)

Figure I.2. **Uncurated** 256×256 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “golden retriever” (207)

Figure I.3. **Uncurated** 256×256 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “husky” (250)

Figure I.4. **Uncurated** 256×256 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “arctic wolf” (270)

Figure I.5. **Uncurated** 256×256 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “panda” (388)

Figure I.6. **Uncurated** 256×256 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “balloon” (417)

Figure I.7. **Uncurated** 512×512 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “Loggerhead sea turtle” (33)

Figure I.8. **Uncurated** 512×512 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “macaw” (88)

Figure I.9. **Uncurated** 512×512 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “lion” (291)

Figure I.10. **Uncurated** 512×512 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “red panda” (387)

Figure I.11. **Uncurated** 512×512 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “ice cream” (928)

Figure I.12. **Uncurated** 512×512 **MixFlow-XL samples.**
AutoGuidance Scale = 1.5
Class label = “coral reef” (973)

(a) RAE: Uniform sampling for the training timestep

```
def sample(self, x1):
    """Sampling x0 & t based on shape of x1 (if
    needed)
      Args:
        x1 - data point; [batch, *dim]
    """

    x0 = th.randn_like(x1)
    dist_options = self.time_dist_type.split("_")
    t0, t1 = self.check_interval(self.train_eps,
    self.sample_eps)
    if dist_options[0] == "uniform":
        t = th.rand((x1.shape[0],)) * (t1 - t0) +
     t0
    # ...


    t = t.to(x1)
    t = self.time_dist_shift * t / (1 + (self.
    time_dist_shift - 1) * t)
    return t, x0, x1
```

(b) MixFlow: Beta sampling for the training timestep

```
def sample(self, x1, gamma=0.4):
    """Sampling x0 & t based on shape of x1 (if
    needed)
      Args:
        x1 - data point; [batch, *dim]
    """

    x0 = th.randn_like(x1)
    dist_options = self.time_dist_type.split("_")
    t0, t1 = self.check_interval(self.train_eps,
    self.sample_eps)
    if dist_options[0] == "uniform":
        t = th.rand((x1.shape[0],)) * (t1 - t0) +
     t0
    # ...
    # Sample t from Beta(2,1)
    t = 1 - th.sqrt(t)
    t = t.to(x1)
    # Sample mt from unshifted t
    mt = t + th.rand_like(t) * gamma * (1 - t)
    # Apply shift to both t and mt
    t = time_dist_shift * t / (1 + (
    time_dist_shift - 1) * t)

    mt = time_dist_shift * mt / (1+(time_dist_shift-1) * mt)
    return t, mt, x0, x1
```

(c) RAE: Standard training loss

```
def training_losses(self, model,  x1,
    model_kwargs=None):
    """Loss for training the score model
    Args:
    - model: backbone model; could be score,
    noise, or velocity
    - x1: datapoint
    - model_kwargs: additional arguments for the
    model
    """
    if model_kwargs == None:
        model_kwargs = {}

    t, x0, x1 = self.sample(x1)



    t, xt, ut = self.path_sampler.plan(t, x0, x1)
    model_output = model(xt, t, **model_kwargs)
    B, *_, C = xt.shape
    assert model_output.size() == (B, *xt.size()
    [1:-1], C)

    terms = {}
    terms['pred'] = model_output
    if self.model_type == ModelType.VELOCITY:
        terms['loss'] = mean_flat(((model_output
    - ut) ** 2))
    else:
        # ...

    return terms
```

(d) MixFlow: Training loss with mixed slowed interpolations

```
def training_losses(self, model, x1, model_kwargs
    =None):
    """Loss for training the score model
    Args:
    - model: backbone model; could be score,
    noise, or velocity
    - x1: datapoint
    - model_kwargs: additional arguments for the
    model
    """
    if model_kwargs == None:
        model_kwargs = {}

    t, mt, x0, x1 = self.sample(x1)
    # Compute slowed interpolation

    _, xt, ut = self.path_sampler.plan(mt, x0, x1)
    model_output = model(xt, t, **model_kwargs)
    B, *_, C = xt.shape
    assert model_output.size() == (B, *xt.size()
    [1:-1], C)

    terms = {}
    terms['pred'] = model_output
    if self.model_type == ModelType.VELOCITY:
        terms['loss'] = mean_flat(((model_output
    - ut) ** 2))
    else:
        # ...

    return terms
```

Figure I.13. **Only 6 lines of code are modified.** for the RAE implementation. Two functions in https://github.com/bytetriper/RAE/blob/main/src/stage2/transport/transport.py are modified. Left: RAE; Right: MixFlow + RAE. Note: in the RAE implementation , $t = 1$ corresponds to the noise and $t = 0$ corresponds to the clean data. The lines corresponds to the modified code.