Multi-Patch Global-to-Local Transformer Architecture For Efficient Flow Matching and Diffusion Model

Supplementary Material

In the supplementary, we first present the results in MPDiT in Sec. 6. Sec. 7 summaries the training time and convergence of our method against the baseline DiT/SiT. Detail of FNO time embedding implementation is provided in Sec. 8. Finally, we include more qualitative results in Sec. 9.

6. ImageNet 512 results

Model	Params(M)	GFLOPs	FID ↓	Prec ↑	Rec ↑
BigGAN [3]	160	-	8.43	**0.88**	0.29
StyleGAN-XL [56]	166	-	2.41	0.77	0.52
MaskGIT [5]	227	-	7.32	0.78	0.50
VQ-GAN [18]	227	-	26.52	0.73	0.31
VAR-d36-s [61]	2300	-	2.63	-	-
ADM-U [16]	731	2813	3.85	0.84	0.53
U-ViT-L/4 [2]	287	76.5	4.67	0.87	0.45
U-ViT-H/4 [2]	501	133.3	4.03	0.84	0.48
Simple Diff [29]	2000	-	4.53	-	-
VDM++ [32]	2000	-	2.65	-	-
DiT-XL/2 [48]	675	524.7	3.04	0.84	0.54
SiT-XL [40]	675	524.7	2.62	0.84	**0.57**
DiM-H [60]	860	708	3.78	-	-
DiffuSSM-XL [72]	673	1066.2	3.41	0.85	0.49
DiCo-XL [1]	701	349.8	2.53	0.83	0.56
MPDiT-XL (ours)	482	228.4	**2.47**	0.83	0.56

Table 8. Quantitative results of ImageNet 512 with MPDiT$_{k=6}$

Method	Epoch	Params(M)	GFLOPs↓	FID↓
No cfg				
DiT-XL/2	600	675	524.7	11.93
MPDiT$_{\{22,6\}}$	120	482	228.4	9.24
MPDiT$_{\{18,6,4\}}$	120	491	138.2	11.77
With cfg				
DiT-XL/2	600	675	524.7	3.04
SiT-XL/2	600	675	524.7	2.62
MPDiT$_{\{22,6\}}$	120	482	228.4	2.47
MPDiT$_{\{18,6,4\}}$	120	491	138.2	3.13

Table 9. Performace of different MPDiT-XL variants on ImageNet 512

Training Details: All 512^2 experiments are conducted on a single node A100 (40GB) with a total batch size of 256, trained for 120 epochs (approximately 600K iterations). Notably, MPDiT reduces memory consumption, allowing the full batch size of 256 to fit within a single node of A100 40GB GPU.

Sampling Details: For sampling process, we adopt Euler sampling with 250 NFEs. For guided sampling, we use cfg scale 1.375. We follow the evaluation protocol from [16] to sample $50,000$ images and compute the evaluation metrics with 1000 provided reference images.

Main Results: As shown in Tab. 8, MPDiT-XL$_{k=6}$, which applies patch size 4 to the first 22 transformer blocks and patch size 2 to the last 6 blocks, achieves an FID of 2.47, outperforming all baselines. Remarkably, it reaches this performance within only 120 training epochs and requires just 228.4 GFLOPs, corresponding to merely 43.5% of the GFLOPs of the DiT and SiT baselines. The non-cherrypicked qualitative results of MPDiT-XL$_{k=6}$ is shown in Fig. 3.

Performance of different variant MPDiT-XL: We further evaluate several MPDiT-XL variants. **MPDiT**$_{\{22,6\}}$ (equivalent to MPDiT-XL$_{k=6}$) denotes a two-stage configuration in which the first 22 Transformer blocks use patch size 4 and the final 6 blocks use patch size 2. **MPDiT**$_{\{18,6,4\}}$ extends this to a three-level hierarchy: the first 18 blocks use patch size 8, the next 6 blocks use patch size 4, and the final 4 blocks use patch size 2. As shown in Tab. 9, MPDiT$_{\{22,6\}}$ already surpasses the DiT/SiT baselines after only 120 training epochs while requiring just 228.4 GFLOPs. To further reduce computational cost, we explore MPDiT$_{\{18,6,4\}}$, which achieves a strong trade-off between performance and efficiency. Its non-guided version outperforms DiT-XL/2 with an FID of **11.77**, despite being trained for only 120 epochs. With classifier-free guidance, it attains an FID of **3.13**, slightly behind DiT-XL/2. However, it is important to highlight that MPDiT$_{\{18,6,4\}}$ uses only $\sim$ 26% of the GFLOPs of DiT/SiT and only is trained for 120 epochs, suggesting substantial room for improvement with longer training. Finally, note that SiT-XL/2 is evaluated using **SDE sampling** (FID 2.62), whereas all MPDiT results are obtained using **ODE sampling**. Since SDE typically yields better sample quality than ODE, the comparison in Tab. 9 is not strictly apple-to-apple.

7. Convergence & Efficiency Analysis

Imagenet 256: Tab. 1 shows that MPDiT-XL achieves an FID of 2.05 with only 240 training epochs and 59.3 GFLOPs. This corresponds to just 17.2% of the training epochs and 49.9% of the per-iteration GFLOPs of the DiT baseline. Overall, the total training compute of MPDiT-XL amounts to only 8.8% of that required by DiT and SiT, indicating a convergence that is approximately $11.36\times$ faster.

Figure 3. Qualitative Result of Imagenet 512 with cfg=4

Imagenet 512: Tab. 8 demonstrates that MPDiT-XL achieves an FID of 2.47, outperforming all compared methods while training for only 120 epochs with 228.4 GFLOPs. This corresponds to merely 20% of the training epochs and 43.5% of the per-iteration GFLOPs of DiT/SiT, resulting in a total training compute of only 8.7%. Consequently, MPDiT converges approximately 11.5× faster than DiT/SiT.

Inference Time: Under the same GPU and number of function evaluations (NFEs), MPDiT achieves more than 2× faster sampling compared to the DiT and SiT baselines, while also consuming less memory. This improved efficiency enables MPDiT to sample effectively across a wider range of GPU devices.

Training Memory Consumption: During training, under identical settings, MPDiT enables significantly larger batch sizes than baseline DiT/SiT. For 256 × 256 resolution, we can fit a total batch size of 1024 on a single node with 8 A100 (40GB) GPUs, which is infeasible for DiT/SiT due to their higher GFLOPs. For 512 × 512 resolution, MPDiT similarly fits a batch size of 256 on the same hardware, while DiT/SiT cannot. These results demonstrate that MPDiT allows efficient diffusion/flow matching training without requiring multi-node clusters or higher-memory GPUs.

8. FNO Time Embedding Details

The detail implementation of FNO time embedding is provided in Algorithm 1 and Algorithm 2. Note that in Algorithm 1, we use 1D grid as the time features. We have tried to replace 1D grid time feature with cos-sin sinusoidal time feature like in traditional time embedding and find out the model unable to converge.

9. More Qualitative Results

More qualitative results are shown in Fig. 4, Fig. 5, Fig. 6, Fig. 7, Fig. 8, Fig. 9, Fig. 10, Fig. 11, Fig. 12 and Fig. 13. All images are non-cherry pick and is sampled with Euler 250 steps and cfg scale is 4.

Algorithm 1 PyTorch code of FNO time embedding

```
import torch
import torch.nn.functional as F

class FNOTimestepEmbedder(nn.Module):
   def __init__(self, hidden_size, modes=16, width=32):
      super().__init__()
      self.modes = modes
      self.width = width

      # lift scalar time to width-dim signal
      self.fc0 = nn.Linear(1, self.width)

      # spectral convolution layers
      self.conv0 = SpectralConv1d(self.width, self.width, self.modes)
      self.conv1 = SpectralConv1d(self.width, self.width, self.modes)
      self.conv2 = SpectralConv1d(self.width, self.width, self.modes)

      # local 1x1 convs
      self.w0 = nn.Conv1d(self.width, self.width, 1)
      self.w1 = nn.Conv1d(self.width, self.width, 1)
      self.w2 = nn.Conv1d(self.width, self.width, 1)

      # projection to model dimension
      self.fc1 = nn.Linear(self.width, hidden_size)

   def forward(self, t):
      B = t.shape[0]
      t = t.unsqueeze(-1)             # (B, 1)

      # build 1D grid and center around timestep
      grid = torch.linspace(-1, 1, 32, device=t.device)
      grid = grid.unsqueeze(0).expand(B, -1) # (B, 32)
      grid = grid + t                 # (B, 32)

      # lift to width dimension
      x = self.fc0(grid.unsqueeze(-1)) # (B, 32, W)
      x = x.permute(0, 2, 1)          # (B, W, 32)

      # three FNO blocks (spectral + local conv)
      x = F.gelu(self.conv0(x) + self.w0(x))
      x = F.gelu(self.conv1(x) + self.w1(x))
      x = self.conv2(x) + self.w2(x)

      # average pool and project to model dim
      x = x.mean(dim=-1)              # (B, W)
      x = self.fc1(x)
      return x
```

Algorithm 2 PyTorch code of 1D spectral convolution

```
import torch

class SpectralConv1d(nn.Module):
   def __init__(self, in_channels, out_channels, modes):
      super().__init__()
      self.in_channels = in_channels
      self.out_channels = out_channels
      self.modes = modes

      # Fourier weights (real and imaginary parts)
      self.weights_real = nn.Parameter(
         torch.randn(in_channels, out_channels, modes)
      )
      self.weights_imag = nn.Parameter(
         torch.randn(in_channels, out_channels, modes)
      )

   def forward(self, x):
      # store dtype and switch to float32 for FFT
      dtype = x.dtype
      x_fp32 = x.float()

      # compute FFT
      x_ft = torch.fft.rfft(x_fp32, dim=-1)
      xr, xi = x_ft.real, x_ft.imag

      # allocate output spectrum
      B, C_out = x.shape[0], self.out_channels
      G = x.shape[-1] // 2 + 1
      out_r = torch.zeros(B, C_out, G, device=x.device)
      out_i = torch.zeros(B, C_out, G, device=x.device)

      # complex multiplication on first modes
      if self.modes <= xr.shape[-1]:
         xr_m = xr[:, :, :self.modes]
         xi_m = xi[:, :, :self.modes]

         wr = self.weights_real.float()
         wi = self.weights_imag.float()

         real = torch.einsum("bim,iom->bom", xr_m, wr) - \
              torch.einsum("bim,iom->bom", xi_m, wi)
         imag = torch.einsum("bim,iom->bom", xr_m, wi) + \
              torch.einsum("bim,iom->bom", xi_m, wr)

         out_r[:, :, :self.modes] = real
         out_i[:, :, :self.modes] = imag

      # inverse FFT to return to spatial domain
      out_ft = torch.complex(out_r, out_i)
      y = torch.fft.irfft(out_ft, n=x.shape[-1], dim=-1)

      # convert back to original precision
      return y.to(dtype)
```

Figure 4. Qualitative images of class 113 ”snail”

Figure 5. Qualitative images of class 33 ”loggerhead, loggerhead turtle, Caretta caretta”

Figure 6. Qualitative images of class 84 ”peacock”

Figure 7. Qualitative images of class 37 ”box turtle, box tortoise”

Figure 8. Qualitative images of class 88 ”macaw”

Figure 9. Qualitative images of class 207 ”golden retriever”

Figure 10. Qualitative images of class 417 ”balloon”

Figure 11. Qualitative images of class 947 ”mushroom”

Figure 12. Qualitative images of class 980 ”volcano”

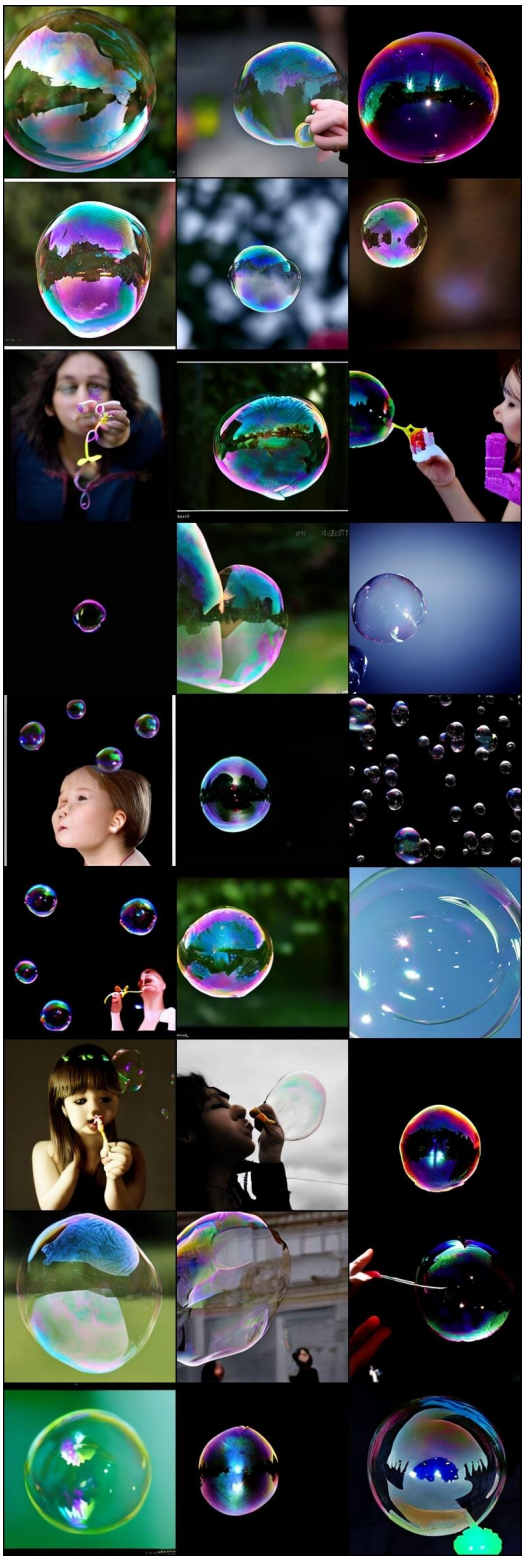

Figure 13. Qualitative images of class 971 ”bubble”