A. Implementation details

Diffusion pre-training. We follow official implementations of DDPM, EDM and DiT for generative diffusion pre-training. The networks used in DDPM and EDM are UNets based on Wide ResNet with multiple convolutional down-sampling and up-sampling stages. Single head self-attention layers are used in the residual blocks at some resolutions. For CIFAR-10, we retrieve official checkpoints[1][2] from their codebases. For Tiny-ImageNet, we use official (or equivalent) implementations and similar configurations to train unconditional diffusion models by ourselves. The setting is in Table 5. Transformer-based DiT-XL/2 pre-trained on 256^2 ImageNet is retrieved from its official codebase[3], and we do not train a smaller version (*e.g.* DiT-B/2) due to the high computational cost. The used off-the-shelf VAE model for latent compression is retrieved from Stable Diffusion[4], which has a down-sample factor of 8.

Linear probing and fine-tuning. We use very simple settings for linear probing and fine-tuning experiments (see Table 6 and Table 7) and we intentionally do not tune the hyper-parameters such as Adam β_1/β_2 or weight decays. In contrast with common practices in representation learning, we do not use additional normalization layers before linear classifiers since we find it also works well.

To train latent-space DiTs for recognition efficiently, we store the extracted latent codes through the VAE encoder and train DiTs in an offline manner. We encode 10 versions of the training set with data augmentations and randomly sample one version per epoch at the training. This approach may suffer from insufficient augmentation, and increasing augmentation versions or training with online VAE encoder may improve the recognition accuracy.

Supervised training from scratch. In Figure 4, we present recognition accuracies of truncated UNet encoders trained from scratch and compare them to supervised Wide ResNets. The setting is in Table 8. We intentionally train these supervised models for long duration (200 epochs) to reach maximum performance for fair comparisons.

B. Layer-noise combinations in grid search

In Section 3.2 we have shown that the layer-noise combination affects representation quality heavily. We perform grid searching to find a good enough, if not the best, combination for each model and dataset. For 18-step or 50-step EDM models, we train linear classifiers for 10 epochs with each layer and timestep. For 1000-step DDPM or DiT, we increase the timestep by 5 or 10 to search more efficiently. Table 9 shows the combinations adopted in Section 4.

[1]https://github.com/pesser/pytorch_diffusion
[2]https://github.com/NVlabs/edm
[3]https://github.com/facebookresearch/DiT
[4]Hugging Face/Diffusers

dataset	**CIFAR-10**		**Tiny-ImageNet**	
model	**DDPM**	**EDM**	**DDPM**	**EDM**
architecture	DDPM	DDPM++	DDPM	DDPM++
base channels	128	128	128	128
channel multipliers	1-2-2-2	2-2-2	1-2-2-2	1-2-2-2
attention resolutions	{16}	{16}	{16}	{16}
blocks per resolution	2	4	2	4
full DDAE params	35.7M	55.7M	35.7M	61.8M
pre-training epochs	2000	4000	2000	2000

Table 5. **Network specifications for diffusion pre-training.**

config	**value**
optimizer	Adam with default momentum & weight decay
base learning rate	1e-3
learning rate schedule	cosine decay
batch size per GPU	128
GPUs	4
augmentations	`RandomHorizontalFlip()` and `RandomCrop(32, 4)` for CIFAR-10 or `RandomCrop(64, 4)` for Tiny-ImageNet
training epochs	CIFAR-10 / Tiny-ImageNet: DDPM 10 / 20; EDM 15 / 30; DiT 30 / 30

Table 6. **Linear probing setting.**

config	**value**
optimizer	Adam with default momentum & weight decay
base learning rate	1e-3 (DDPM and EDM), 8e-5(DiT)
learning rate schedule	cosine decay
batch size per GPU	128 (DDPM and EDM), 8 (DiT)
GPUs	4 (DDPM and EDM), 8 (DiT)
augmentations	`RandomHorizontalFlip()` and `RandomCrop(32, 4)` for CIFAR-10 or `RandomCrop(64, 4)` for Tiny-ImageNet
training epochs	CIFAR-10 / Tiny-ImageNet: DDPM 30 / 80; EDM 50 / 100; DiT 50 / 50

Table 7. **Fine-tuning setting.**

config	**value**
optimizer	Adam (DDAE encoder), SGD (Wide ResNet)
base learning rate	5e-4 (DDAE encoder), 0.1 (Wide ResNet)
learning rate schedule	cosine decay
batch size per GPU	128
GPUs	4
augmentations	`RandomHorizontalFlip()` and `RandomCrop(32, 4)` for CIFAR-10 or `RandomCrop(64, 4)` for Tiny-ImageNet
training epochs	200
warmup epochs	5

Table 8. **Setting for training supervised models from scratch.**

model	**dataset@resolution**	**layer**	**timestep**
DDPM	CIFAR-10@32	7/12 (1st block@16)	11/1000
EDM	CIFAR-10@32	6/15 (1st block@16)	4/18
DiT	CIFAR-10@256	12/28	121/1000
DDPM	Tiny-ImageNet@64	2/12 (2nd block@8)	45/1000
EDM	Tiny-ImageNet@64	7/20 (2nd block@16)	14/50
DiT	Tiny-ImageNet@256	13/28	91/1000

Table 9. **Adopted layer-noise combinations.** The numbers following "@" denote image or feature map resolutions.