Can Generative Geospatial Diffusion Models Excel as Discriminative Geospatial Foundation Models?

Supplementary Material

6. Diffusion Models

Below, we provide a concise mathematical overview of discrete diffusion models (DMs).

6.1. Diffusion Model Essentials

Let $\mathbf{x}_0 \sim p(\mathbf{x}_0)$ be a sample from an underlying data distribution. A forward diffusion process iteratively adds Gaussian noise over T discrete timesteps, producing corrupted samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$. One common choice to model each step is:

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\big(\mathbf{x}_t;\ \sqrt{1-\beta_t}\,\mathbf{x}_{t-1},\ \beta_t\,\mathbf{I}\big), \qquad (5)$$

where $\beta_t \in (0,1)$ controls the noise variance. One can obtain $\mathbf{x}_t$ directly from the original image $\mathbf{x}_0$ given the cumulative effect of t nose-adding steps:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \quad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i, \qquad (6)$$

where $\alpha_t = 1 - \beta_t$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. As t grows, $\mathbf{x}_t$ becomes increasingly noisy; at $t = T$, the corrupted $\mathbf{x}_T$ approximates a pure Gaussian distribution, losing most structure of $\mathbf{x}_0$.

To generate novel samples starting from pure noise, a diffusion model learns a reverse denoising process $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$, parameterized by θ, which conceptually "denoises" $\mathbf{x}_t$ step by step until recovering $\mathbf{x}_0$:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\big(\mathbf{x}_{t-1};\ \boldsymbol{\mu}_\theta(\mathbf{x}_t, t),\ \Sigma_\theta(\mathbf{x}_t, t)\big). \qquad (7)$$

A common training objective is to minimize the distance between the true noise $\boldsymbol{\epsilon}_t$ and the model's predicted noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$:

$$\mathcal{L}_{\text{simple}}(\theta)\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},t}\big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2\big]. \qquad (8)$$

At inference, sampling proceeds from $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively applies p_θ to yield a final $\mathbf{x}_0$.

6.2. Feature Extraction using Diffusion Models

While diffusion models (DMs) are primarily designed for image generation from Gaussian noise, our goal is to extract their learned representations for real images. To achieve this, we first *invert* a real image into a noisy state and then perform the reverse denoising process.

To illustrate the inversion step, we revisit DDIM [49], a widely adopted sampling approach known for faster generation and invertibility. A common deterministic formulation for going from $\mathbf{x}_t$ to $\mathbf{x}_{t-1}$ is:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\hat{\mathbf{x}}_0 + \sqrt{1-\alpha_{t-1}}\ \epsilon_\theta(\mathbf{x}_t, t). \qquad (9)$$

where $\hat{\mathbf{x}_0} = \frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\,\epsilon_\theta(\mathbf{x}_t,t)}{\sqrt{\alpha_t}}$ is the predicted clean image. By removing explicit Gaussian noise additions at each step, the process becomes deterministic, allowing a "mirror pass" that encodes $\mathbf{x}_0$ to $\mathbf{x}_T$. If we then use $\mathbf{x}_T$ as the start of the usual sampling procedure, we recover the original $\mathbf{x}_0$.

We leverage this property by reversing the order of timesteps, going from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$, starting from $\mathbf{x}_0$, and then running the denoising process on $\mathbf{x}_t$ to extract features.

While one could alternatively introduce noise into a real image by selecting a timestep and manually adding noise via Eq. (6), this approach introduces stochastic variations. To ensure consistency, we adopt DDIM inversion, leveraging its deterministic nature to repurpose diffusion models for discriminative tasks.

7. Datasets Description

There are six classification datasets in GEO-Bench [27]:

m-bigearthnet It contains 120 × 120 images with 43 land cover classes. The dataset includes 20,000 training samples, 1,000 validation samples, and 1,000 test samples. It consists of 12 spectral bands obtained from Sentinel-2 imagery, with a spatial resolution of 10.0m for the RGB channels.

m-brick-kiln It consists of 64 × 64 images with 2 classes, focusing on brick kiln detection. The dataset includes 15,063 training samples, 999 validation samples, and 999 test samples. The imagery is derived from Sentinel-2 with 10 spectral bands and a resolution of 10.0m for RGB. Additional Sentinel-1 data is included.

m-eurosat It consists of 64 × 64 images spanning 10 land cover classes. The dataset contains 2,000 training samples, 1,000 validation samples, and 1,000 test samples. It includes 13 spectral bands captured from Sentinel-2 with an RGB resolution of 10.0m.

m-forestnet This dataset contains 332 × 332 images and covers 12 classes related to forest monitoring. It includes 6,464 training samples, 989 validation samples, and 993 test samples. The dataset comprises 6 spectral bands obtained from Landsat-8, with a spatial resolution of 15.0m for the RGB channels.

m-pv4ger This dataset comprises 320 × 320 images covering 2 classes, with 11,814 training samples, 999 validation

samples, and 999 test samples. The imagery is obtained from RGB data, with a spatial resolution of 0.1m.

m-so2sat This dataset consists of 32 × 32 images spanning 17 different land cover classes. It contains 19,992 training samples, 986 validation samples, and 986 test samples. The images are derived from Sentinel-2 data with 13 spectral bands and a spatial resolution of 10.0m for RGB.

In addition, six semantic segmentation datasets are included:

m-pv4ger-seg It is the segmentation variant of m-pv4ger, containing 320 × 320 images with 3,000 training samples, 403 validation samples, and 403 test samples. The dataset has 3 spectral bands (RGB) with a spatial resolution of 0.1m.

m-nz-cattle It contains 500 × 500 images with 2 classes, including 524 training samples, 66 validation samples, and 65 test samples. The imagery consists of 3 spectral bands (RGB) with an unknown spatial resolution.

m-NeonTree It includes 400 × 400 images with 2 classes, consisting of 270 training samples, 94 validation samples, and 93 test samples. The dataset comprises 5 spectral bands (RGB + Hyperspectral + Elevation).

m-cashew-plantation It comprises 256 × 256 images with 7 classes, featuring 1,350 training samples, 400 validation samples, and 400 test samples. The imagery is sourced from Sentinel-2 with 10 spectral bands and an RGB resolution of 10.0m.

m-SA-crop-type This dataset consists of 256 × 256 images with 10 classes. It contains 3,000 training samples, 1,000 validation samples, and 1,000 test samples. The imagery is sourced from Sentinel-2 with 10 spectral bands and an RGB resolution of 10.0m.

m-chesapeake-landcover This dataset consists of 256 × 256 images with 7 land cover classes. It contains 3,000 training samples, 1,000 validation samples, and 1,000 test samples. The dataset includes 4 spectral bands (RGBN) with a spatial resolution of 1.0m.

8. Evaluation Details

We provide additional details regarding the evaluation process in this section

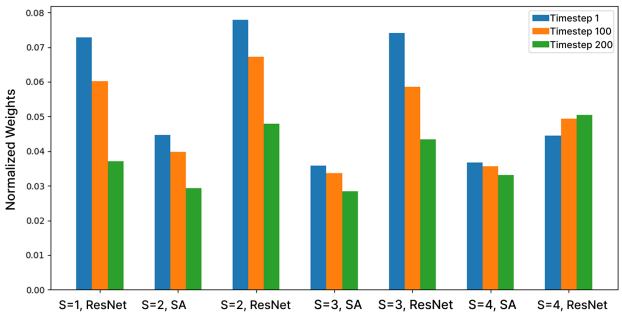
For task training criteria, we utilize nn.CrossEntropyLoss() from the PyTorch library for all tasks, except for the m-bigearthnet dataset, which follows a multi-label classification setup and nn.BCEWithLogitsLoss() is applied.

Regarding training schedules, all models are trained for 60 epochs on the m-cashew-plantation and m-sa-crop-type datasets, while the remaining datasets undergo training for 40 epochs.

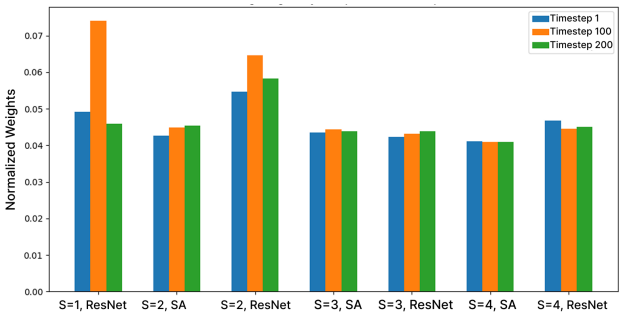
Additionally, we apply data augmentation techniques for all datasets to enhance model generalization. During training, images undergo random horizontal flipping, vertical flipping, and color jittering based on a probabilistic threshold of 0.5.

9. Visualizations of Global Weighting

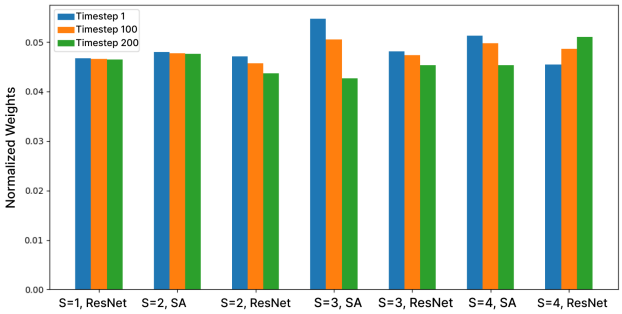
In Sec. 4.4, we demonstrate that features from different blocks and timesteps contribute differently depending on the dataset. Our global weighted fusion method effectively aggregates these features to enhance performance. Fig. 7 visualizes the learned weight distributions across blocks and timesteps at different scales, illustrating how our fusion strategy dynamically adjusts feature importance for each dataset. This automated weighting helps reduce the need for manual feature selection, promoting adaptive and optimal feature integration.



(a) Weight allocations for m-chesapeake-landcover dataset.



(b) Weight allocations for m-pv4ger-seg dataset.



(c) Weight allocations for m-NeonTree dataset.

Figure 7. Normalized weight allocations in Global Weighted Fusion across different datasets.