Seed Feature Maps-based CNN Models for LEO Satellite Remote Sensing Services

Zhichao Lu[1], Chuntao Ding[2*], Shangguang Wang[3], Ran Cheng[4], Felix Juefei-Xu[5], and Vishnu Naresh Boddeti[6]
[1]School of Software Engineering, Sun Yat-sen University, Zhuhai, China.
[2]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China.
[3]Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China.
[4]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
[5]New York University, New York, NY, USA.
[6]Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA
Email: luzhichaocn@gmail.com, chuntaoding@163.com, sgwang@bupt.edu.cn, ranchengcn@gmail.com, juefei.xu@nyu.edu, vishnu@msu.edu

***Abstract*—Deploying high-performance convolutional neural network (CNN) models on low-earth orbit (LEO) satellites for rapid remote sensing image processing has attracted significant interest from industry and academia. However, the limited resources available on LEO satellites contrast with the demands of resource-intensive CNN models, necessitating the adoption of ground-station server assistance for training and updating these models. Existing approaches often require large floating-point operations (FLOPs) and substantial model parameter transmissions, presenting considerable challenges. To address these issues, this paper introduces a ground-station server-assisted framework. With the proposed framework, each layer of the CNN model contains only one learnable feature map (called the seed feature map) from which other feature maps are generated based on specific rules. The hyperparameters of these rules are randomly generated instead of being trained, thus enabling the generation of multiple feature maps from the seed feature map and significantly reducing FLOPs. Furthermore, since the random hyperparameters can be saved using a few random seeds, the ground station server assistance can be facilitated in updating the CNN model deployed on the LEO satellite. Experimental results on the ISPRS Vaihingen, ISPRS Potsdam, UAVid, and LoveDA datasets for semantic segmentation services demonstrate that the proposed framework outperforms existing state-of-the-art approaches. In particular, the SineFM-based model achieves a higher mIoU than the UNetFormer on the UAVid dataset, with 3.3× fewer parameters and 2.2× fewer FLOPs.**

***Index Terms*—Remote sensing services, CNN, nonlinear transformation, seed feature maps, random seed**

I. Introduction

Euroconsult estimates that around 18,500 small satellites will be launched between 2022 and 2031[1]. As of December 2022, SpaceX has launched more than 3,000 low-earth orbits (LEO) satellites, and OneWeb has launched 462 LEO satellites[2]. While primarily serving communication purposes, these satellites also acquire many remote-sensing images. The large-scale, multi-spectral, and rich-source characteristics of remote sensing images have attracted significant attention from industry and academia for their inherent advantages in applications such as land-use monitoring [1], understanding traffic flow [2], population migration prediction [3], and climate change tracking [4].

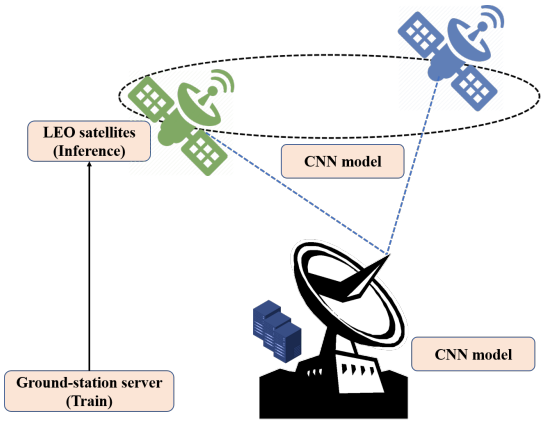


Fig. 1: System architecture for ground station server-assisted training and updating of CNN models.

The conventional method for processing remote sensing image data involves satellites capturing the data, sending it to ground station servers, and using convolutional neural network (CNN) models (e.g., ResNet [5], [6], VGG [7]) for processing to produce high-performance results. However, this approach has three limitations: (i) High Bandwidth Requirements: The transmission of large amounts of raw data from LEO satellites to ground station servers consumes considerable bandwidth. (ii) Slow Response Time: The communication distance between the satellite and the ground station server is long, which can result in extended transmission delays when transmitting large amounts of data. This is further exacerbated by satellites being powered mainly by solar energy, limiting the transmission power and data transmission rate. (iii) Increased Bit Error Rates: The link channel is susceptible to various interferences, resulting in a relatively high bit error rate of the transmitted

*Corresponding author
[1]https://mundogeo.com/en/2022/08/09/smallsats-18500-satellites-weighing-up-to-500-kg-will-be-launched-between-2022-and-2031/
[2]https://planet4589.org/space/con/star/stats.html

data.

Processing remote sensing image data directly on LEO satellites can partially alleviate these limitations, as transmitting processed results instead of raw remote sensing data can significantly reduce bandwidth consumption, transmission delay, and bit error rates. However, this approach also introduces two new challenges: (i) Limited LEO satellite resources and the resource-intensive nature of CNN models pose a challenge for deployment. Performing convolutions in CNN models during inference requires a significant amount of computing resources, as the convolution filter must slide across the remote sensing image multiple times, generating many floating-point operations (FLOPs). For instance, ResNet-50 requires 4.1 billion FLOPs to process an image of size 224×224 [5], [8]. This makes it challenging to deploy high-performance CNN models on resource-limited LEO satellites. (ii) The long lifespan of LEO satellites, typically exceeding five years, and the rapid development of CNN models pose challenges in terms of limited bandwidth resources and many model parameters. Therefore, to maintain high service quality, it is necessary to update the CNN model frequently. However, high-performance CNN models often contain a large number of parameters, making the transmission of these parameters during model updates challenging. For example, CoAtNet-7 [9] has 2440 million parameters.

To address the above challenges, this paper proposes a ground station server-assisted training and updating the framework for CNN models on LEO satellites processing remote sensing images, as depicted in Fig. 1. Our approach begins with introducing SineFM, a novel and efficient feature map generation algorithm. SineFM only requires learning one filter parameter, referred to as the seed filter, for each layer in the CNN model. By inputting data into the seed filter, a single feature map, referred to as the seed feature map, is generated. This seed feature map is then used to generate other feature maps for that layer through nonlinear transformation functions, thereby significantly reducing the FLOPs.

Additionally, we propose that the parameters of the nonlinear transformation function that generates feature maps from the seed feature map be randomly initialized and not trained. This allows the parameters to be saved and easily reproducible using a random seed. As a result, when updating the CNN model, the ground station server only needs to transmit a small number of seed filters and random seeds to the LEO satellites, significantly reducing the number of transmitted parameters.

Experiments on the ISPRS Vaihingen, ISPRS Potsdam, UAVid, and LoveDA datasets demonstrate the strong performance of our proposed framework for semantic segmentation services. Our SineFM-based CNN model slightly improved mIoU (+0.3%) while requiring $3.3\times$ fewer parameters and $2.2\times$ fewer FLOPs. In summary, our main contributions are as follows:

- We propose a novel method, SineFM, to reduce FLOPs in CNN models by using nonlinear transformation to generate feature maps. To our knowledge, this is the first work to use this approach. The hyperparameters of the nonlinear transformation can be saved using a few random seeds, making it possible for the ground station server to send a small number of parameters for deployment and updating the model on the LEO satellite.
- Our theoretical analysis of the SineFM-based layer shows that it can approximate the standard convolutional layer well and achieve better performance.
- Our experiments demonstrate that our proposed framework reduces approximately 3.3 times the model parameter transmission and about 2.2 times the FLOPs.

The rest of the paper is structured as follows. First, Section II reviews the relevant literature. Then, Section III describes our proposed framework in detail. Then, Section IV presents our evaluation results. Finally, we conclude the paper in Section V.

II. Background and Related Work

We briefly prior work on convolutional neural network models and server-assisted training and inference, from which our work draws inspiration.

A. Convolution Neural Network Models

Convolutional neural network (CNN) models [5], [7], [10], [11] have achieved overwhelming success in fields such as computer vision, natural processing, and speech recognition, and using them to provide high-quality services has become mainstream. For example, AlexNet [10] won the 2012 ImageNet large-scale vision challenge and greatly improved the accuracy. VGG [7] proved that increasing the depth can affect the performance of the CNN to a certain extent. ResNet [5] extends the depth of the CNN to 152 layers by introducing the residual module and significantly improves its performance on multiple vision tasks. The excellent performance of the above CNN models is largely attributed to the use of convolution operations to extract features. However, the convolution operation extracts features by sliding filters over the input data, which incurs a large number of FLOPs. To this end, many alternatives to convolution operations have been proposed [8], [12]–[16]. For example, MobileNets [12]–[14] utilized the depthwise and pointwise convolutions to construct a unit to approximate the standard convolution. ShuffleNet [15], [16] further explored a channel shuffling operation to ensure model performance while reducing FLOPs.

Different from the above methods, instead of using filters to slide the input data to obtain feature maps, we use nonlinear transformation to generate feature maps, which can significantly reduce the number of FLOPs for running CNN models.

B. Server-assisted Training and Inference

Sufficient computing resources are one of the important supports for training high-performance CNNs, and a large number of server-assisted CNN training researches have appeared in recent years [17]–[23]. For example, Neurosurgeon [19] divides a CNN model into a head and a tail to run on the device and cloud server respectively. SPINN [21] trains and infers CNN models through the collaboration of devices

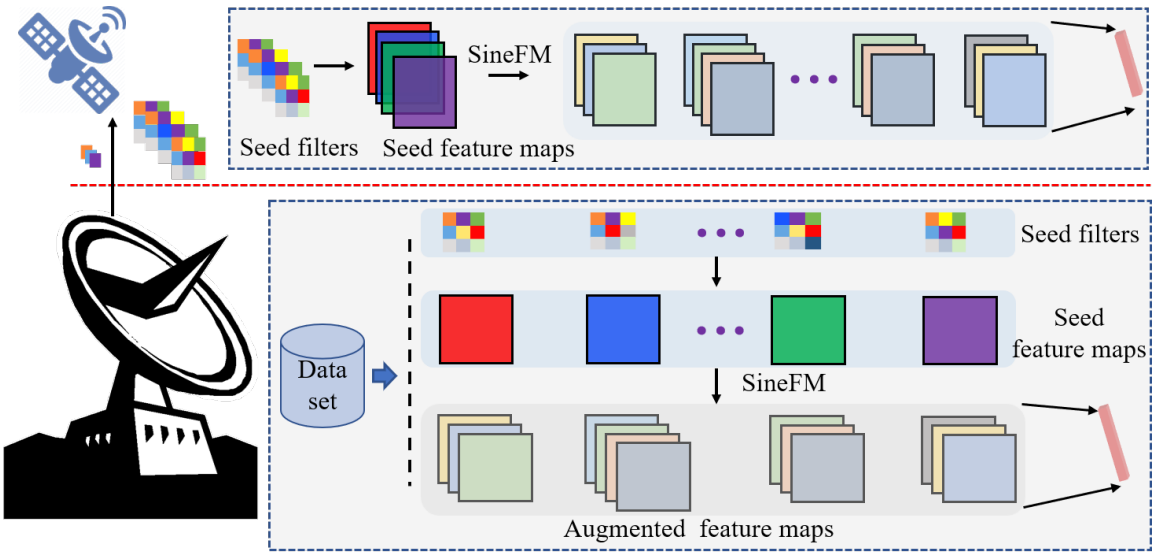


Fig. 2: Overview of the proposed framework. We first design the ground station server-assisted training CNN model method. Then, we design a SineFN algorithm. Each layer of CNN first learns the weights of one filter (i.e., the seed filter). Then, a feature map (i.e., seed feature map) is generated based on the seed filter and the input. Finally, based on the seed feature map, the SineFM generates other required feature maps. This makes the ground station server only need to send the seed filter and a small number of seeds to the LEO satellites. Subsequently, the LEO satellite generates the well-trained CNN based on the seed filters, seeds, and SineFM.

and the cloud server, and uses compression and quantization techniques to reduce the number of model parameter exchange between the devices and the cloud server. However, the above methods are highly dependent on the network conditions between the server and the device. When network conditions are unstable or disconnected, it can greatly degrade the user experience. To solve this problem, MonoCNN [24] and TFormer [25] are proposed, both of which first train the CNN model on the cloud server, and then send the CNN model to the device for deployment and subsequent update. For example, MonoCNN is the first to introduce seed filters and random seeds to generate other filters to reduce the number of model parameters sent from the server to the device.

Different from the above methods, our method uses random seeds to store and reproduce hyperparameters of nonlinear transformation, so that the ground station server only needs to send a small number of parameters to the LEO satellites. In addition, compared with MonoCNN, using nonlinear transformation to generate feature maps greatly reduces the FLOPs required to run CNN models.

III. Design of the Proposed Approach

A. Overview

Fig. 2 illustrates the architecture of the proposed framework. Our goal is to provide a framework suitable for ground station server-assisted LEO satellite deployment and update CNN models to provide high-performance remote sensing image processing services. To achieve this goal, as shown, the proposed framework first trains a CNN on the ground station server, and then sends the well-trained CNN to the LEO satellite to provide remote sensing services.

In the proposed framework, we first propose a cheap feature map generation algorithm, SineFM. Then, we propose to save and reproduce the hyperparameters of the nonlinear transformation with random seeds to reduce the number of parameters transmitted from the ground station server to LEO satellites. We describe the proposed framework in detail as follows: (i) the ground station server-assisted method in Section III-B, (ii) the SineFM algorithm in Section III-C.

B. Design of ground station server-assisted method

We use the ground station server to train the CNN first, and then send the well-trained CNN to the LEO satellites based on the following three considerations. (i) Due to the limitation of size and weight, it is difficult for LEO satellites to provide sufficient computing resources for training high-performance CNN models. However, the ground station server has sufficient computing resources. (ii) LEO satellite-ground bandwidth is limited and frequency band resources are scarce. However, LEO satellite-ground collaborative training or inference CNN models lead to a large amount of satellite-ground data transmission. (iii) Lightweight CNN model technology is developing rapidly. It is mainstream to train lightweight CNN models on the ground station server and then send them to LEO satellites for inference to provide fast response services.

C. Design of SineFM

Convolution operations require a large number of floating-point operations (FLOPs), which is an important indicator to

measure the computing resources required by the CNN model. Here, we count the floating-point operation of one multiplication and one addition as one FLOP. At any given convolutional layer, the FLOPs are equal to $C_i \times K^2 \times W \times H \times C_o$, where C_i is the number of input channels, K is the size of the convolutional filter, W and H are the height and width of the output feature maps, respectively, and C_o is the number of output channels. Typically, a high-performance CNN contains hundreds of convolutional layers, resulting in a large number of FLOPs. This makes CNN models unsuitable to run on resource-constrained LEO satellites.

To this end, we propose SineFM algorithm, whose goal is to replace the use of sliding windows to generate feature maps with nonlinear transformations to reduce the FLOPs required to run CNN models. Next, we explain the SineFM from four parts: generating feature maps through nonlinear transformation, SineFM-based layer, nonlinear transformation, and theoretical analysis.

1) Generating feature maps via nonlinear transformation: At any given layer, we learn a small number of feature maps $\boldsymbol{y}^{(\mathrm{s})}$ by convolving the inputs with a small number of convolutional filters $\boldsymbol{W}^{(\mathrm{s})}$ (i.e, *seed filters*),

$$\boldsymbol{y}^{(\mathrm{s})} = \boldsymbol{W}^{(\mathrm{s})} * \boldsymbol{x} \tag{1}$$

where $\boldsymbol{x}$ are the inputs. Then, we generate the remaining feature maps $\boldsymbol{y}^{(\mathrm{g})}$ by applying nonlinear transformations $\phi(\cdot)$ of the seed feature maps $\boldsymbol{y}^{(\mathrm{s})}$. For simplicity, we use monomial functions as the choice of $\phi(\cdot)$ to illustrate the concept. Thereafter, the forward propagation for this layer can be mathematically formulated as

$$y_i^{(\mathrm{g})} = \phi\big(y_i^{(\mathrm{s})}\big) = y_i^{(\mathrm{s})\circ\beta_\mathrm{i}} \tag{2}$$

$$\tilde{y}_i^{(\mathrm{g})} = \frac{y_i^{(\mathrm{g})} - \frac{1}{n}\sum_i y_i^{(\mathrm{g})}}{\left(\sum_i \left(y_i^{(\mathrm{g})} - \frac{1}{n}\sum_i y_i^{(\mathrm{g})}\right)^2\right)^{\frac{1}{2}}} \tag{3}$$

where β_i is the exponent of the monomial function applied to the i^{th} channel of the seed feature maps. We also normalize the response maps (i.e., zero mean and unit variance) to prevent the responses from vanishing or exploding, and the normalized response map is now referred to as $\tilde{\boldsymbol{y}}^{(\mathrm{g})}$. On the other hand, backpropagation through such a nonlinear transformation of the seed feature maps w.r.t a loss function ℓ (e.g., cross-entropy) requires the computation of $\partial\ell/\partial\boldsymbol{y}^{(\mathrm{s})}$, where

$$\frac{\partial \ell}{\partial y_i^{(\mathrm{s})}} = \frac{\partial \ell}{\partial \tilde{y}_i^{(\mathrm{g})}} \frac{\partial \tilde{y}_i^{(\mathrm{g})}}{\partial y_i^{(\mathrm{g})}} \frac{\partial y_i^{(\mathrm{g})}}{\partial y_i^{(\mathrm{s})}} \tag{4}$$

$$\frac{\partial \tilde{y}_i^{(\mathrm{g})}}{\partial y_i^{(\mathrm{g})}} = \frac{1 - \frac{1}{n}}{\left(\sum_i \left(y_i^{(\mathrm{g})} - \frac{1}{n}\sum_i y_i^{(\mathrm{g})}\right)^2\right)^{\frac{1}{2}}} - \frac{\left(1-\frac{1}{n}\right)\left(y_i^{(\mathrm{g})} - \frac{1}{n}\sum_j s[j]\right)}{\left(\sum_i \left(y_i^{(\mathrm{g})} - \frac{1}{n}\sum_i y_i^{(\mathrm{g})}\right)^2\right)^{\frac{3}{2}}} \tag{5}$$

$$\frac{\partial y_i^{(\mathrm{g})}}{\partial y_i^{(\mathrm{s})}} = \beta_i y_i^{(\mathrm{s})\circ(\beta_\mathrm{i}-1)} \tag{6}$$

2) SineFM-based layer: The core idea of the SineFM[3] is to restrict the CNN network to learn only one (or a few) feature maps at each layer, and through nonlinear transformations, we can generate or augment as many feature maps as needed at each layer. The gist is that there are no parameters associated with the augmented features that need to be updated or learned since they are entirely generated by nonlinear transformations and are no longer updated with the back-propagation procedure. The hyperparameters of the nonlinear transformation can be saved and reproduced by using several random seeds, which makes the ground station server only needs to send a small number of seed filters and random seeds to the LEO satellites to complete the deployment and update of the model. Therefore, SineFM is suitable for ground station server to assist LEO satellites deployment and update CNN models.

As shown in Fig. 2, our SineFM-based layer starts with just one (or a few) seed feature map $\boldsymbol{y}^{(\mathrm{s})}$ learned by the seed filters $\boldsymbol{W}^{(\mathrm{s})}$. If we desire m feature maps in total for one layer, the $m-1$ feature maps $\boldsymbol{y}^{(\mathrm{g})}$ are nonlearnable and are the nonlinear transformation of the seed feature map $\boldsymbol{y}^{(\mathrm{s})}$. In our implementation, we include the seed filter as part of the filter bank in the forward propagation, i.e., $[\boldsymbol{y}^{(\mathrm{s})}, \tilde{\boldsymbol{y}}^{(\mathrm{g})}]$. However, it could be excluded if chosen. The inputs $\boldsymbol{x}$ are initially processed by these seed filters and become m response maps, which are then passed through an activation gate, such as ReLU σ_{relu}, and become m feature maps. The m feature maps are linearly combined using m learnable weights, which can be operationalized through a 1×1 convolution layer.

$$\boldsymbol{y} = \sum_{i=1}^{m} \alpha_i \sigma_{\mathrm{relu}}\left(\left[\boldsymbol{W}^{(\mathrm{s})} * \boldsymbol{x}, \phi\left(\boldsymbol{W}^{(\mathrm{s})} * \boldsymbol{x}\right)\right]\right) \tag{7}$$

where α represents the 1×1 convolution weights, $\alpha(\cdot)$ is an activation, and $\phi(\cdot)$ is the nonlinear transformation function. The PyTorch-like pseudocode of SineFM-based layer is demonstrated in Algorithm 1.

In addition, through a lot of experiments, we find that replacing the convolutional layer with the SineFM-based layer can improve the performance of the model. This is because the

[3]In this paper, we omit the bias terms of convolutional filters for brevity.

Algorithm 1 SineFM-based layer: PyTorch-like Pseudocode

```
# func: non-linear transformation function
# k: fan-out ratio
# C_i, C_s, C_o: No. of input/seed/output
    features
class OurConv2d(nn.Conv2d):
   def __init__(self, C_in, C_out, C_s, func, **
       kwargs):
      super(OurConv2d, self).__init__(
         C_in, C_out, **kwargs)
      self.weight = None # ensure non-learnable
      # No. of generated features
      C_g = C_s * (ceil(C_o/C_s) - 1)
      # seed filters for generating seed feasures
      self.conv = nn.Conv2d(C_i, C_s, **kwargs)
      self.linear = nn.Conv2d(k*C_s, C_o,
         kernel_size=1, **kwargs)

   def forward(self, x):
      y_s = self.conv(x) # seed features
      y_g = func(y_s, k) # generate k*C_s
          features
      y_g = self.linear(normalize(y_g)) # linear
          combination
      return torch.cat([y_s, y_g])
```

rule of the nonlinear transformation function acts to regularize the model, thereby improving the performance of the model.

3) Design of nonlinear transformation function: There are many nonlinear functions that can be used to generate feature maps. In this work, we primarily consider monomials, polynomials, radial basis functions, and sinusoidal functions given their well-studied theoretical properties and prevalence in the literature. A summary and visualizations of these four types of non-linear transformation functions are provided in Table I and Fig. 3, respectively. In this work, we empirically determine the choice of feature-generating function through ablation analysis. For polynomial and radial basis function families, we have multiple candidates, we first identify the best choice within their respective families.

Figs. 4a and 4b depict the results. Evidently, we observe that (i) Legendre outperforms other alternative polynomial functions and (ii) all radial basis functions perform similarly with Gaussian being slightly more stable. Accordingly, we use the Legendre polynomial and Gaussian radial basis function and provide an overall comparison among the four types of nonlinear transformations considered in Fig. 4c. We observe that the sinusoidal function outperforms other candidate functions. Therefore, we use the sinusoidal function to generate feature maps for the main results presented in this work.

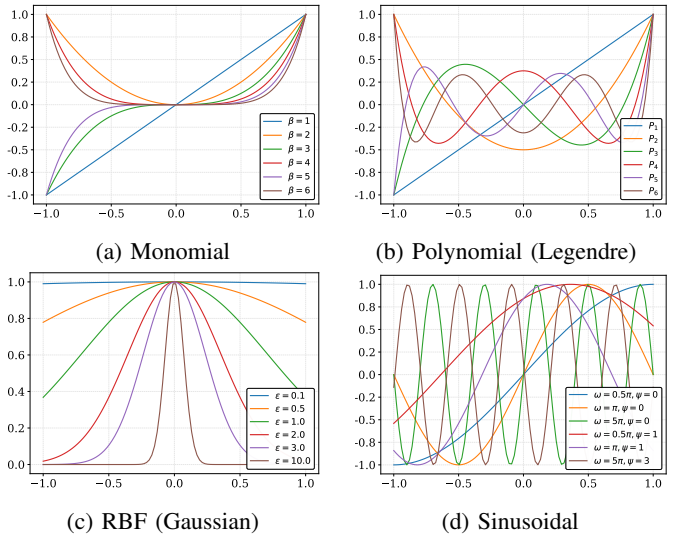
4) Theoretical analysis: In this section, we provide a theoretical analysis of our layer and demonstrate how it can well approximate the standard convolutional layer.

At layer l, let $\boldsymbol{x}_\pi \in \mathbb{R}^{(C\cdot k\cdot k)\times 1}$ be a vectorized single patch from the C-channel inputs at location π, where k is the kernel size of the convolutional filter. Let $\boldsymbol{w} \in \mathbb{R}^{(C\cdot k\cdot k)\times 1}$ be a vectorized single convolution filter from the convolutional filter tensor $\boldsymbol{W} \in \mathbb{R}^{C\times k\times k\times m}$, which contains a total of m generated convolutional filters at layer l. We drop the layer subscription l for brevity.

In a standard CNN, this patch $\boldsymbol{x}_\pi$ is taken as a dot product

TABLE I: Overview of nonlinear transformation functions explored in this work for generating feature maps.

	Transformation	Formulation
	Monomial	$\text{sign}(x)\lvert x\rvert^{\beta}$
Polynomials	Chebyshev	$T_n(x) = \begin{cases} cos(n\ arccos\ x), & \text{for } \lvert x\rvert \le 1 \\ cosh(n\ arcosh\ x), & \text{for } x \ge 1 \\ (-1)^n cosh\big(n\ arcosh(-x)\big), & \text{for } x \le 1 \end{cases}$
	Hermite	$H_n(x) = (-1)^n\, e^{x^2}\, \frac{d^n}{dx^n} e^{-x^2}$
	Legendre	$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} \left(x^2-1\right)^n$
Radial basis functions	Gaussian	$e^{-(\epsilon x)^2}$
	Multiquadratic	$\sqrt{1+(\epsilon x)^2}$
	Inverse Quadratic	$\frac{1}{1+(\epsilon x)^2}$
	Inverse Multiquadratic	$\frac{1}{\sqrt{1+(\epsilon x)^2}}$
	Sinusoidal	$sin(\omega x + \psi)$



(a) Monomial (b) Polynomial (Legendre)

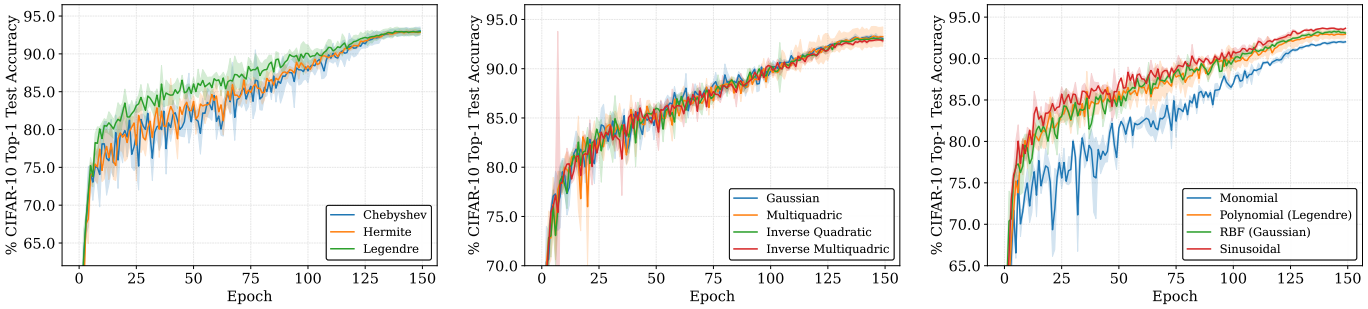
(c) RBF (Gaussian) (d) Sinusoidal

Fig. 3: Visualization of different types of nonlinear transformations.

with the filter $\boldsymbol{w}$, followed by the nonlinearity (e.g., ReLU σ_{relu}), resulting in a single output feature value d_π at the corresponding location π on the feature map. Similarly, each value of the output feature map is a direct result of convolving the entire input map $\boldsymbol{x}$ with a convolutional filter $\boldsymbol{w}$. This microscopic process can be expressed as:

$$d_\pi = \sigma_{\text{relu}}(\boldsymbol{w}^\top \boldsymbol{x}_\pi) \tag{8}$$

For our proposed layer (we again consider monomials as the choice of $\phi(\cdot)$ feature generation functions for illustration), the seed features are the direct results of convolving the inputs $\boldsymbol{x}$ with the seed convolutional filters $\boldsymbol{w}^{(\text{s})}$. Then, we obtain a set of m response maps by applying the monomial transformation to seed features with exponents coefficients of $\beta_1, \beta_2, \ldots, \beta_m$ which are pre-defined and not updated during training. The corresponding output feature map value $\tilde{d}_\pi$ is also a linear combination of the corresponding elements from the m response maps via 1×1 convolution with parameters



(a) Polynomial function (b) Radial basis function (c) Overall comparison

Fig. 4: Comparison of different choices (a) polynomial and (b) radial basis functions for generating features from seed features. (c) Overall comparison among different feature-generating functions.

$\alpha_1, \alpha_2, \ldots, \alpha_m$. This process follows:

$$\tilde{d}_\pi = \sigma_{\text{relu}} \underbrace{\left(\begin{bmatrix} (\boldsymbol{w}^{(\text{s})\top} \boldsymbol{x}_\pi)^{\circ \beta_1} \\ (\boldsymbol{w}^{(\text{s})\top} \boldsymbol{x}_\pi)^{\circ \beta_2} \\ \cdots \\ (\boldsymbol{w}^{(\text{s})\top} \boldsymbol{x}_\pi)^{\circ \beta_m} \end{bmatrix} \right)^\top}_{1 \times m} \underbrace{\boldsymbol{\alpha}}_{m \times 1}$$
$$= \sigma_{\text{relu}} \left(\begin{bmatrix} \phi_{\beta_1}(\boldsymbol{w}^{(\text{s})})^\top \phi_{\beta_1}(\boldsymbol{x}_\pi) \\ \phi_{\beta_2}(\boldsymbol{w}^{(\text{s})})^\top \phi_{\beta_2}(\boldsymbol{x}_\pi) \\ \cdots \\ \phi_{\beta_m}(\boldsymbol{w}^{(\text{s})})^\top \phi_{\beta_m}(\boldsymbol{x}_\pi) \end{bmatrix} \right)^\top \boldsymbol{\alpha} = \boldsymbol{c}_{\text{relu}}^\top \boldsymbol{\alpha} \quad (9)$$

where $\phi_{\beta_i}(\cdot)$ is the point-wise monomial expansion with exponent β_i. Comparing d_π (Eq 8) and $\tilde{d}_\pi$ (Eq 9), we consider the following two cases (i) when $d_\pi = 0$: since $\boldsymbol{c}_{\text{relu}} = \sigma_{\text{relu}}\left(\left[\cdots \right] \right) \geq 0$, there always exists a vector $\boldsymbol{\alpha} \in \mathbb{R}^{m \times 1}$ such that $\tilde{d}_\pi = d_\pi$. However, when (ii) $d_\pi > 0$, it is obvious that the approximation does not hold when $\boldsymbol{c}_{\text{relu}} = \boldsymbol{0}$. Thus, under the assumption that $\boldsymbol{c}_{\text{relu}}$ is not an all-zero vector, the approximation $\tilde{d}_\pi \approx d_\pi$ will hold.

In summary, SineFM is the core of the proposed framework. There are three advantages as follows: First, based on the seed feature map, using nonlinear transformation instead of sliding windows to generate feature maps, which effectively reduces the FLOPs required to run the CNN model. Second, the hyperparameters of nonlinear transformations are randomly generated and nonlearnable, which allows these parameters to be saved and reproduced with some random seeds, so that only a small number of parameters need to be transmitted from the ground station to the LEO satellite. Third, the nonlinear transformation plays the role of regularizing the model which can improve its performance.

IV. Experimental Evaluation

In this section, we first introduce our experimental setup including the datasets, baselines, and evaluation metrics, followed by the implementation details. Then, we provide an empirical comparison in terms of FLOPs, learnable parameters, and performance on multiple remote sensing datasets.

A. Experimental Setup

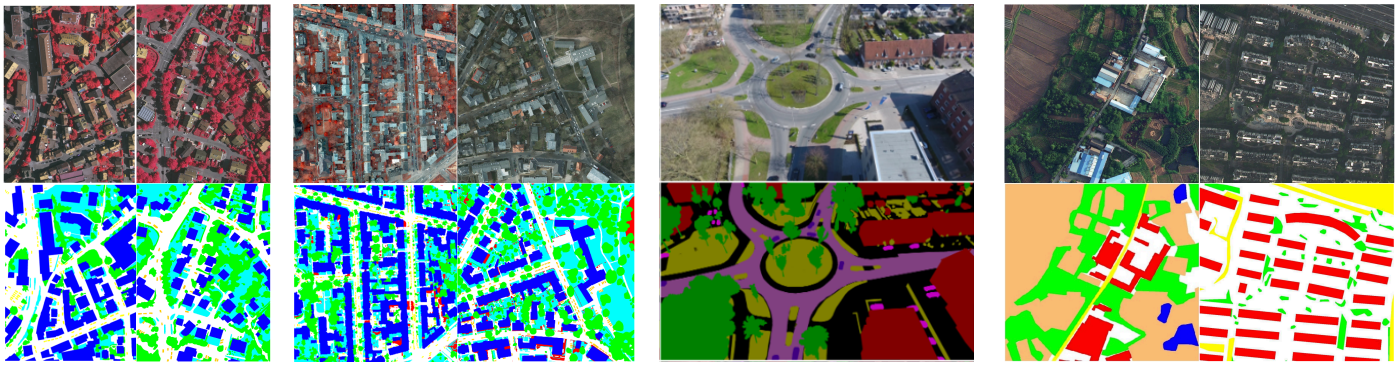
Datasets. Four widely-used remote sensing datasets are considered for evaluating the efficacy of the proposed approach. Sample images of these datasets are provided in Fig. 5.

ISPRS Vaihingen [26] dataset contains 33 high-quality images with topographical information, each with an average resolution of 2494×2064 pixels. These images, referred to as "TOP image tiles," have three color channels (i.e., near-infrared, red, green), as well as a digital surface model and a normalized digital surface model (NDSM) with a ground sampling distance of 9 cm. The dataset includes five categories of objects in the foreground (i.e., impervious surface, building, low vegetation, tree, car) and one category for the background (i.e., clutter). In the experiments conducted, only the TOP image tiles were used, and the DSM and NDSM were disregarded. The images used for testing were identified by the following IDs: 2, 4, 6, 8, 10, 12, 14, 16, 20, 22, 24, 27, 29, 31, 33, 35, and 38.

ISPRS Potsdam [27] dataset includes 38 high-resolution aerial images (with a resolution of 5 cm per pixel) that are 6000×6000 pixels in size. These images include the same categories as the Vaihingen dataset. The dataset includes four types of data: red, green, blue, and near-infrared multispectral bands, as well as digital surface models and normalized digital surface models. For testing purposes, we used images ID: 2_13, 2_14, 3_13, 3_14, 4_13, 4_14, 4_15, 5_13, 5_14, 5_15, 6_13, 6_14, 6_15, and 7_13, and the remaining 23 images[4] were used for training. In the experiments, only the red, green, and blue bands were used, and the images were cropped into 1024×1024 pixel sections.

UAVid [28] is a dataset for semantic segmentation of Unmanned Aerial Vehicles (UAVs) with high resolution. It mainly features urban street scenes with two different sizes of images (3840×2160 and 4096×2160) and 8 different categories. The segmentation of UAVid images presents a challenge due to their high spatial resolution, diverse spatial variations, unclear category definitions, and complex scenes. The dataset consists of 42 sequences, with a total of 420 images. Among these, 200

[4]We exclude image id 7_10 due to error annotations.



(a) ISPRS Vaihingen (b) ISPRS Potsdam (c) UAVid (d) LoveDA

Fig. 5: **Visualization of Datasets.** The input images are shown in the first row, and the ground truth labels are shown in the second row.

images are used for training, 70 images are used for validation, and the remaining 150 images are designated for testing and provided by the official source. In our experiments, each image was divided into 8 patches of 1024×1024 pixels after being padded and cropped.

LoveDA [29] dataset consists of 5,987 high-quality optical remote sensing images with a resolution of 0.3 meters per pixel for a total size of 1024×1024 pixels. It includes seven types of land cover, namely buildings, roads, water, barren land, forests, agriculture, and backgrounds. Of these images, 2,522 are designated for training, 1,669 for validation, and 1,796 are provided for testing purposes. The dataset covers two types of landscapes, urban and rural, and has been collected from three cities in China: Nanjing, Changzhou, and Wuhan. Due to the multi-scale features of objects, complex backgrounds, and varied distributions of land cover categories, this dataset presents significant challenges.

Baselines. We considered a diverse set of representative baselines collected from the literature, including methods that were developed for general computer vision tasks (e.g., GhostNet [8] and MonoCNN [30]), methods that were tailored for efficient semantic segmentation based solely on CNNs (e.g., BiSeNet [31] and SwiftNet [32]) and assisted by attention modules (e.g., DANet [33] and MANet [34]) or transformers (e.g., Segmenter [35] and SegFormer [36]), and methods that were dedicated to remote sensing, such as UNetFormer [37].

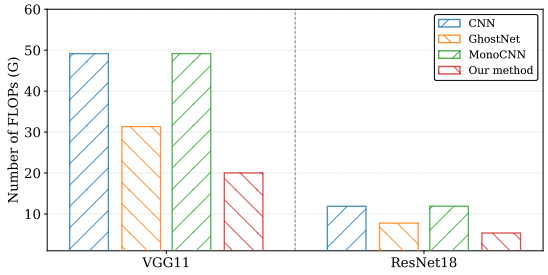
Evaluation Metrics. Our experiments evaluated the performance of all models in two aspects: computational efficiency and prediction accuracy. To measure computational efficiency, we looked at three factors: the number of model parameters ($^{\#}\mathrm{Params}$), the number of floating-point operations ($^{\#}\mathrm{FLOPs}$) required, and the inference speed, which was expressed in terms of frames per second (FPS), for processing a single image. To assess the prediction accuracy, we used three metrics: the mean intersection-over-union (mIoU), the overall accuracy (OA), and the mean F1 score (F1).

Implementation Details. The experiments were carried out using PyTorch with two NVIDIA GTX 2080Ti GPUs. We used AdamW optimizer with an initial learning rate of $6e^{-4}$, which was gradually reduced to zero following the cosine annealing strategy [38]. We incorporate our method in the UNetFormer framework [37] as the backbone feature extractor. For GhostNet [8], we set the reduction ratio to two; for MonoCNN [30], we followed the default settings provided in the original paper. The settings for other baselines were adopted from [37].

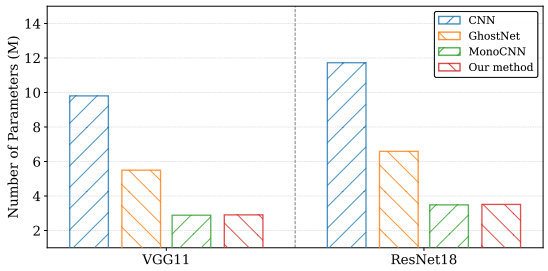
For the UAVid dataset, data augmentation was performed by applying random vertical and horizontal flips, and random brightness changes to input images. The training epoch was set to 40 and the batch size was 8. During testing, additional augmentations such as vertical and horizontal flips were applied using the test-time augmentation (TTA) technique. For the Vaihinge, Potsdam, and LoveDA datasets, the images were randomly cropped into 512×512 patches. The augmentations included random scaling (among [0.5, 0.75, 1.0, 1.25, 1.5]), random vertical and horizontal flips, and random rotations. The training epoch was set to 100 and the batch size was 16. During testing, multi-scale and random flip augmentations were applied.

B. Experimental Results

1) Improvement in Model Efficiency: Our proposed framework achieves the fewest FLOPs and the least amount of model parameter transmission. As shown in Fig. 6, we consider two widely used architectures (VGG11 and ResNet18) and compare the FLOPs and the learnable parameters of SineFM-based model for semantic segmentation applications. Specifically, as shown in Fig. 6a, CNN, GhostNet and MonoCNN contain a large number of feature map generation operations using sliding windows, resulting in a large number of FLOPs. However, our method only requires a sliding operation in each layer of the model to generate a seed feature map, and then uses nonlinear transformations to generate other multiple feature maps based on the seed featur map, avoiding a large number of sliding window operations, thus greatly reducing FLOPs.



(a) FLOPs (G)



(b) Parameters (M)

Fig. 6: Comparison of (a) the number of network parameters required to be transmitted and (b) the number of FLOPs required for each prediction. FLOPs are computed w.r.t. input size of 512×512.

As shown in Fig. 6b, Since all filter parameters in CNN and GhostNet need to be learned, the ground station server needs to send all filter parameters to LEO satellites, resulting in a large amount of model parameter transmission. In our proposed framework, only a single-seed filter parameter and a random seed need to be transmitted in each layer. This is because we can generate a seed feature map based on this seed filter, and then generate other feature maps of this layer based on the seed feature map and nonlinear transformation. The hyperparameters of the nonlinear transformation are randomly initialized and remain fixed so that these hyperparameters can be saved and reproduced by a random number seed. Therefore, compared with transmitting all model parameters, our proposed framework can greatly reduce the amount of model parameter transmission.

Additionally, as shown in Fig. 6b, the number of model parameters transmission by MonoCNN is the same as that of our SineFM-based model, because they both need to transmit only a small number of seed filters and seeds. The biggest difference between the two is the way to generate feature maps, as shown in Fig. 6a, our method reduces more FLOPs.

2) Improvement in Model Performance: Our proposed SineFM-based model consistently outperforms other state-of-the-art models on ISPRS Vaihingen and Potsdam in terms of mean F1, OA, and mIoU. As shown in Table II and Table III, on the ISPRS Vaihingen dataset, compared with BiSeNet, our method achieves a 6.3% higher mean F1, a 4.4% higher OA, and a 7.4% higher mIoU. On the ISPRS Potsdam dataset, our method achieves a 3.4% higher mean F1, a 3.6% higher OA, and a 5.7% higher mIoU. This is because the nonlinear transformation acts to regularize the model when generating feature maps using the nonlinear transformation. Therefore, our method achieves consistently better performance.

TABLE II: Performance on ISPRS Vaihingen.

Method	mean F1 (↑)	OA (↑)	mIoU (↑)
BiSeNet [31]	84.3	87.1	75.8
DANet [33]	79.6	88.2	69.4
SwiftNet [32]	88.3	90.2	79.6
ABCNet [39]	89.5	90.7	81.3
Segmenter [35]	84.1	88.1	73.6
UNetFormer [37]	90.4	91.0	82.7
Our method	90.6	91.5	83.2

TABLE III: Performance on ISPRS Potsdam

Method	mean F1 (↑)	OA (↑)	mIoU (↑)
BiSeNet [31]	89.8	88.2	81.7
DANet [33]	88.9	89.1	80.3
SwiftNet [32]	91.0	89.3	83.8
ABCNet [39]	92.7	91.3	86.5
Segmenter [35]	89.2	88.7	80.7
UNetFormer [37]	92.8	91.3	86.8
Our method	93.2	91.8	87.4

TABLE IV: Performance on UAVid.

	Method	#Params (↓)	#FLOPs (↓)	mIoU (↑)
CNN	BiSeNet [31]	12.9M	51.8G	63.7
	DANet [33]	12.6M	39.6G	62.8
	SwiftNet [32]	11.8M	51.6G	63.3
	MANet [34]	12.0M	51.7G	64.8
	ABCNet [39]	14.0M	62.9G	65.0
Transformer	SegFormer	13.7M	63.3G	68.2
	CoaT [40]	11.1M	105G	68.0
	SwinUNet [41]	41.4M	237G	68.3
	UNetFormer [37]	11.7M	46.9G	70.0
	Our method	3.52M	21.4G	70.3

Our proposed SineFM-based model consistently outperforms other state-of-the-art models on UAVid and LoveDA datasets while significantly reducing the number of learnable parameters and FLOPs. As shown in Table IV, compared with the ABCNet, our method improves mIoU by about 5.3%, but reduces the number of learning parameters by nearly $3\times$ and FLOPs by $3\times$. Compared with UNetFormer, our method improves mIoU by about 0.3%, but reduces the number of learnable parameters by nearly $3\times$ and FLOPs by $2\times$. Akin, as shown in Table V, compared with the UNetFormer, our method improves mIoU by 0.4 and achieves higher FPS (+16). There are three main reasons for these results. (i) The feature maps generated by nonlinear transformations are replaced by sliding window-based feature maps, which effectively reduce the FLOPs of running the model. (ii) The nonlinear transformation parameters are randomly generated and fixed, which effectively reduces the learnable parameters of the model. (iii)

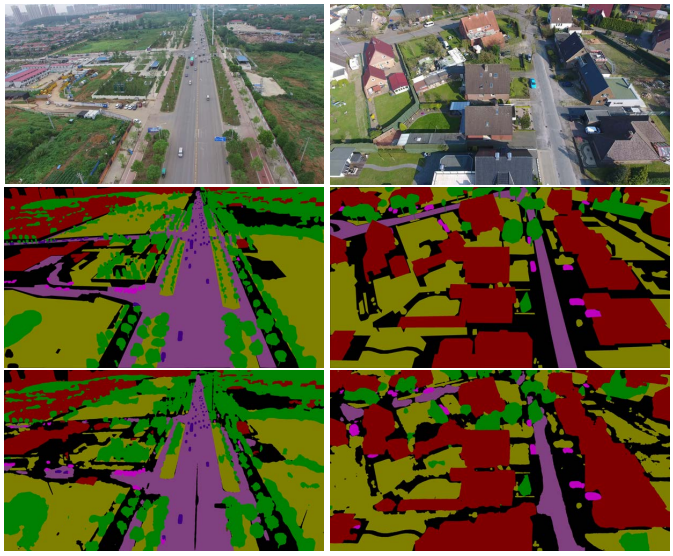


Fig. 7: Qualitative comparison on UAVid dataset. We show input images, ground truth, and predictions from our method from top to bottom.

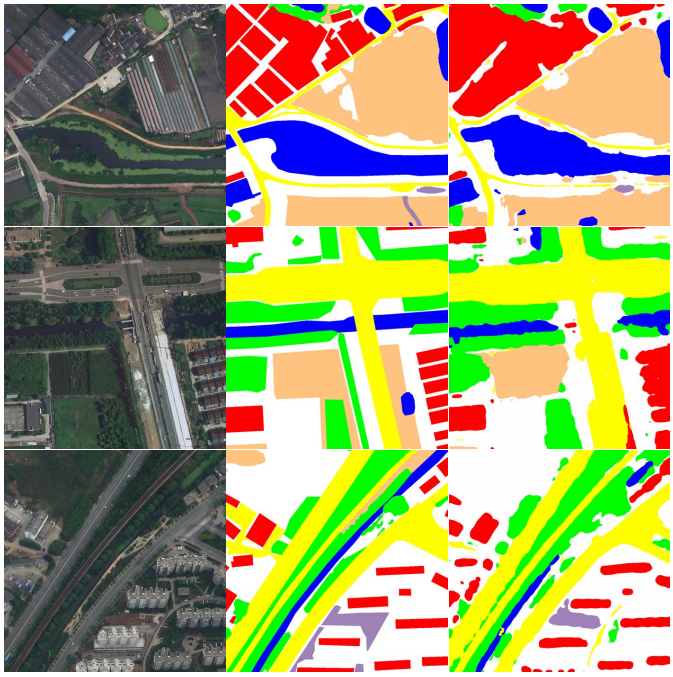


Fig. 8: Qualitative comparison on LoveDA dataset. We show input images, ground truth, and predictions from our method from left to right.

The nonlinear transformation function regularizes the model, thereby improving the performance of the model. Therefore, our proposed framework is especially suitable for ground station server to assist LEO satellite deployment and update the CNN models to provide high-quality services. Additionally, we also provide qualitative visualization on both datasets, as shown in Fig. 7 and Fig. 8.

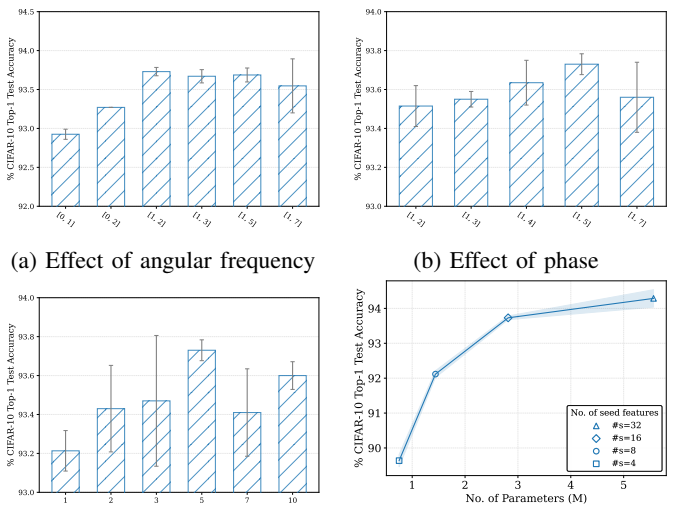
TABLE V: Performance on LoveDA.

Method	#FLOPs (↓)	FPS (↑)	mIoU (↑)
PSPNet [42]	106G	52.2	48.3
DeepLabV3+ [43]	95.8G	53.7	47.6
SemanticFPN [44]	103G	52.7	48.2
SwinUperNet [45]	349G	19.5	50.0
UNetFormer [37]	46.9G	115	52.4
Our method	21.4G	131	52.8

C. Hyperparameter Analysis

The performance of the sinusoidal function for generating features critically depends on the choice of hyperparameters, namely the angular frequency ω and phase ψ (see Table I for more details). For both angular frequency and phase, we randomly sample a ω and ψ from a uniform distribution of $[a, b]$, where a and b are the lower and upper bounds. To understand the effect of ω and ψ, we gradually vary the bounds while keeping all other factors unchanged. Figs. 9a and 9b depict the results of ω and ψ, respectively. In general, we observe that setting the lower bound $a \geq 1$ leads to improved performance. In particular, we identify that sampling ω and ψ from $[1, 2]$ and $[1, 5]$ respectively yields the best result.

On the other hand, we can create multiple sinusoidal functions with different sets of ω and ψ at the same time,



(a) Effect of angular frequency (b) Effect of phase

(c) Effect of fan-out ratio (d) Effect of No. of seed features

Fig. 9: Ablation studies on various hyperparameters.

and we use a hyperparameter (i.e., fan-out ratio) to control this process. Ablative analysis has also been carried out to understand the effect of fan-out ratio. Fig. 9c depicts the results. In general, we observe that more sinusoidal functions simultaneously (i.e., larger fan-out ratio) leads to better performance. Notably, we identify that setting fan-out ratio to five yields the best performance.

Finally, the computational complexity (i.e., the number of parameters and FLOPs) of our method can be controlled

via the number of seed filters. Fig. 9d depicts the trade-off between performance and computational complexity. In general, we observe that using 16 seed filters yields the best trade-off in terms of minimizing the number of parameters and maximizing performance.

V. Conclusion

This paper introduces a novel framework for ground station server-assisted deployment and updating of Low Earth Orbit (LEO) satellites. The proposed framework minimizes the computational requirements and reduces the transmission of model parameters by employing the SineFM technique for nonlinear transformation in generating feature maps for Convolutional Neural Network (CNN) models. The use of random seeds to save and reproduce the hyperparameters of the nonlinear transformation enables the ground station server to transmit a smaller number of parameters to the LEO satellite.

The proposed framework undergoes a thorough analysis of various nonlinear transformation functions and includes a theoretical analysis to support its implementation. Experimental results demonstrate that the proposed framework achieves improved performance in remote sensing image semantic segmentation applications while also reducing the number of floating point operations and parameters transmitted.

In future work, we plan to deploy the CNN model generated by the SineFM algorithm on LEO satellites and evaluate the network transmission traffic sent to the LEO satellite during model updates, assisted by the ground station server. This research represents a significant step towards optimizing the deployment and updating processes of LEO satellites and holds great promise for further advancements in the field.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 62202039, 62106097, 62032003), the National Key Research and Development Program of China (No. 2022ZD0118502), and the Fundamental Research Funds for the Central Universities, Sun Yat-sen University (No. 23qnpy94).

References

[1] C. Robinson, L. Hou, K. Malkin, R. Soobitsky, J. Czawlytko, B. Dilkina, and N. Jojic, "Large scale high-resolution land cover mapping with multi-resolution data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 726–12 735. 1

[2] S. Workman and N. Jacobs, "Dynamic traffic modeling from overhead imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 312–12 321. 1

[3] X. Chen, "Nighttime lights and population migration: Revisiting classic demographic perspectives with an analysis of recent european data," *Remote Sensing*, vol. 12, no. 1, pp. 1–13, 2020. 1

[4] P. Akiva, M. Purri, and M. J. Leotta, "Self-supervised material and texture representation learning for remote sensing tasks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2012, pp. 8193–8205. 1

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. 1, 2

[6] S. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. H. S. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 2, pp. 652–662, 2021. 1

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, 2015, pp. 1–14. 1, 2

[8] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "Ghostnet: More features from cheap operations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1577–1586. 2, 7

[9] Z. Dai, H. Liu, Q. V. Le, and M. Tan, "Coatnet: Marrying convolution and attention for all data sizes," in *Proceedings of the Neural Information Processing Systems*, 2021, pp. 3965–3977. 2

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Neural Information Processing Systems*, 2012, pp. 1106–1114. 2

[11] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261–2269. 2

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," in *CoRR abs/1704.04861*, 2017, pp. 1–9. 2

[13] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520. 2

[14] A. Howard, R. Pang, H. Adam, Q. V. Le, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, and Y. Zhu, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1314–1324. 2

[15] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856. 2

[16] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 122–138. 2

[17] C. Ding, A. Zhou, Y. Liu, R. N. Chang, C.-H. Hsu, and S. Wang, "A cloud-edge collaboration framework for cognitive service," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1489–1499, 2022. 2

[18] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, Y. Ma, S. Chen, and P. Hou, "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 249–261, 2018. 2

[19] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. N. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 615–629. 2

[20] H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, and W. Zhu, "Jalad: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution," in *Proceedings of the IEEE International Conference on Parallel and Distributed Systems*, 2018, pp. 671–678. 2

[21] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "Spinn: synergistic progressive inference of neural networks over device and cloud," in *Proceedings of the Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–15. 2

[22] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proceedings of the IEEE International Conference on Distributed Computing Systems*, 2017, pp. 328–339. 2

[23] C. Ding, A. Zhou, X. Ma, and S. Wang, "Cognitive service in mobile edge computing," in *Proceedings of the International Conference on Web Services*, 2020, pp. 181–188. 2

[24] C. Ding, Z. Lu, F. Juefei-Xu, V. N. Boddeti, Y. Li, and J. Cao, "Towards transmission-friendly and robust cnn models over cloud and device," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2022. 3

[25] Z. Lu, C. Ding, F. Juefei-Xu, V. N. Boddeti, S. Wang, and Y. Yang, "Tformer: A transmission-friendly vit model for iot devices," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 598–610, 2023. 3

[26] "2D Semantic Labeling - Vaihingen data," https://www.isprs.org/education/benchmarks/UrbanSemLab/2d-sem-label-vaihingen.aspx, accessed: 2023-02-05. 6
[27] "2D Semantic Labeling Contest - Potsdam," https://www.isprs.org/education/benchmarks/UrbanSemLab/2d-sem-label-potsdam.aspx, accessed: 2023-02-05. 6
[28] Y. Lyu, G. Vosselman, G.-S. Xia, A. Yilmaz, and M. Y. Yang, "Uavid: A semantic segmentation dataset for uav imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 165, pp. 108 – 119, 2020. 6
[29] J. Wang, Z. Zheng, A. Ma, X. Lu, and Y. Zhong, "Loveda: A remote sensing land-cover dataset for domain adaptive semantic segmentation," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021, pp. 10 347–10 357. 7
[30] C. Ding, Z. Lu, F. Juefei-Xu, V. N. Boddeti, Y. Li, and J. Cao, "Towards transmission-friendly and robust cnn models over cloud and device," *IEEE Transactions on Mobile Computing*, 2022. 7
[31] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 334–349. 7, 8
[32] M. Oršić and S. Šegvić, "Efficient semantic segmentation with pyramidal fusion," *Pattern Recognition*, vol. 110, p. 107611, 2021. 7, 8
[33] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3146–3154. 7, 8
[34] R. Li, S. Zheng, C. Zhang, C. Duan, J. Su, L. Wang, and P. M. Atkinson, "Multiattention network for semantic segmentation of fine-resolution remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–13, 2021. 7, 8
[35] R. Strudel, R. G. Pinel, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision*, 2021, pp. 7242–7252. 7, 8
[36] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 077–12 090, 2021. 7
[37] L. Wang, R. Li, C. Zhang, S. Fang, C. Duan, X. Meng, and P. M. Atkinson, "Unetformer: A unet-like transformer for efficient semantic segmentation of remote sensing urban scene imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 190, pp. 196–214, 2022. 7, 8, 9
[38] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," in *Proceedings of the International Conference on Learning Representations*, 2017, pp. 1–16. 7
[39] R. Li, S. Zheng, C. Zhang, C. Duan, L. Wang, and P. M. Atkinson, "Abcnet: Attentive bilateral contextual network for efficient semantic segmentation of fine-resolution remotely sensed imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 181, pp. 84–98, 2021. 8
[40] W. Xu, Y. Xu, T. Chang, and Z. Tu, "Co-scale conv-attentional image transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9981–9990. 8
[41] H. Cao, Y. Wang, J. Chen, D. Jiang, X. Zhang, Q. Tian, and M. Wang, "Swin-unet: Unet-like pure transformer for medical image segmentation," in *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*. Springer, 2023, pp. 205–218. 8
[42] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890. 9
[43] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 833–851. 9
[44] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 6399–6408. 9
[45] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022. 9