Adapting Foundation Models: A Case Study on Remote Sensing Imagery

Qianru SUN

Singapore Management University

1 Nov 2024

Continual Learning Workshop at ACM MM 2024

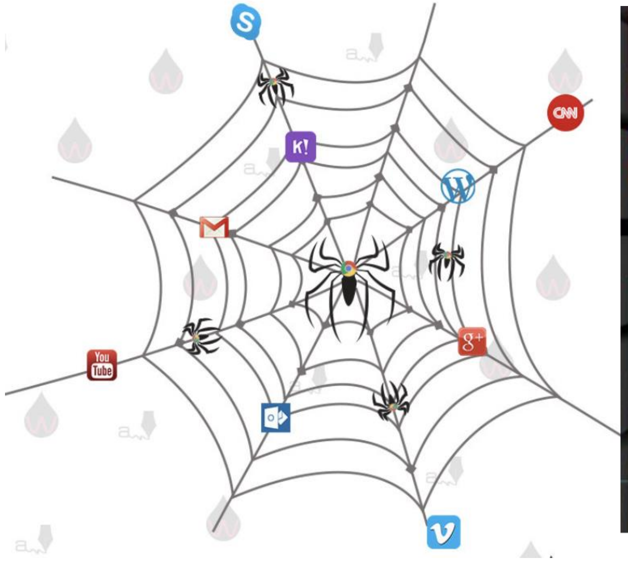
Continual Learning Large Visual Models

- **Two directions**

  Upgrade models with more natural data (images and texts)

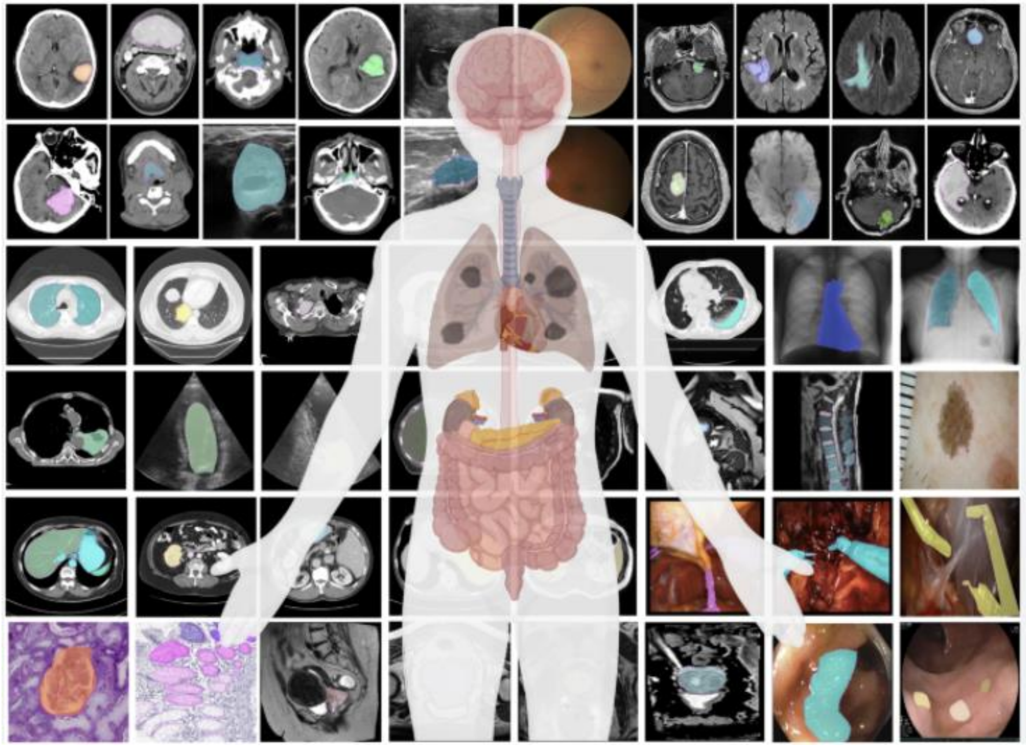
  Adapt models with new domain images, **specifically (e.g., MRI)**

Crawl more natural images for continual pre-training



Daily-life photos

Adapt to new domains with small data



Source: Segment anything in medical images | Nature Communications

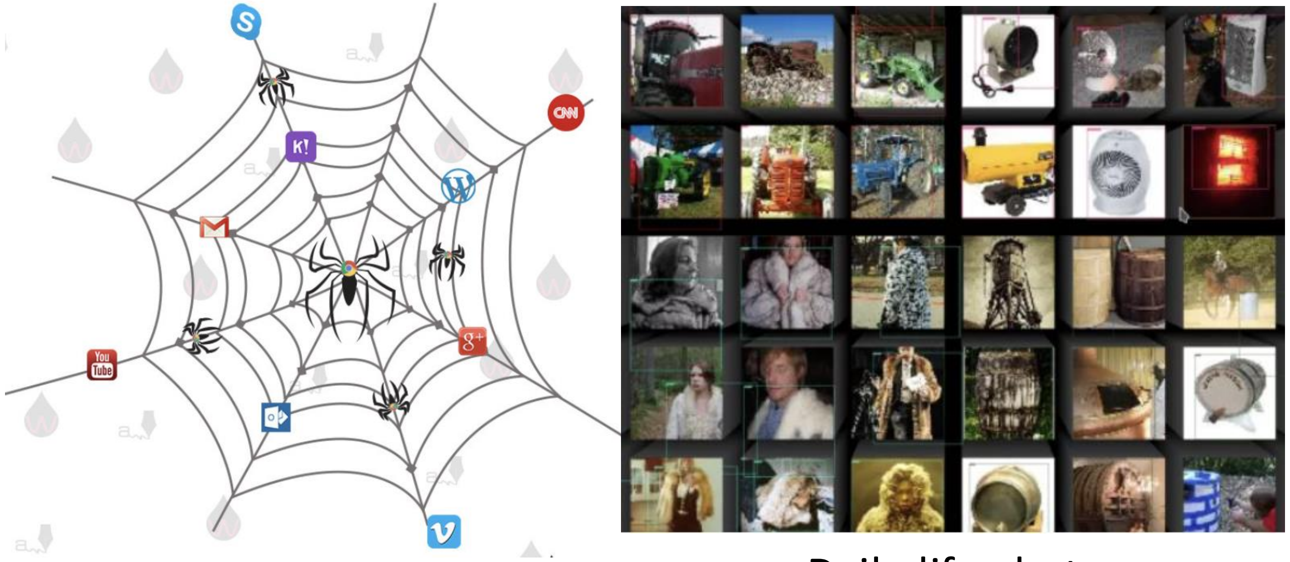
Continual Learning Large Visual Models

- **Two directions**

  Upgrade models with more natural data (images and texts)

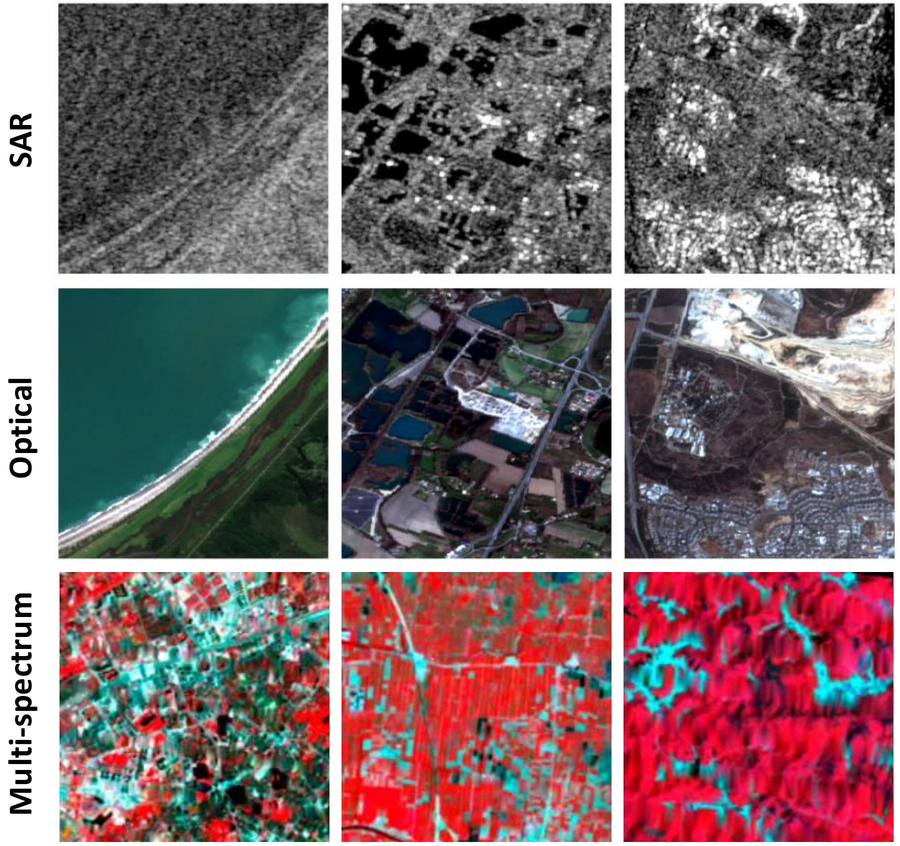
  Adapt models with new domain images, **specifically (e.g., SAR)**

Crawl more natural images for continual pre-training



Daily-life photos

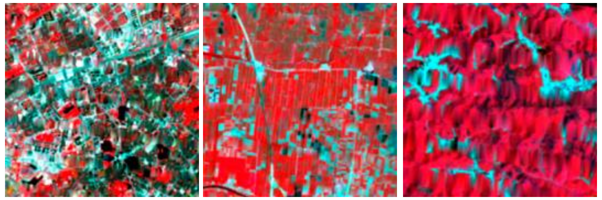
Adapt to new domains with small data



Source: EUSI Database

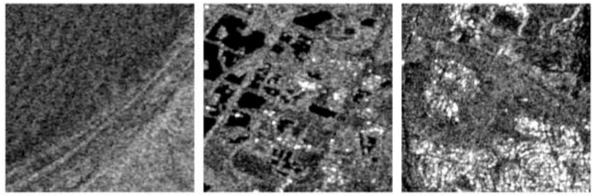
SMU
SINGAPORE MANAGEMENT UNIVERSITY

Multi-spectrum



Optical

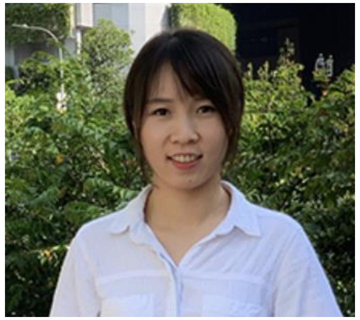
SAR



“Learning De-Biased Representations for Remote-Sensing Imagery”

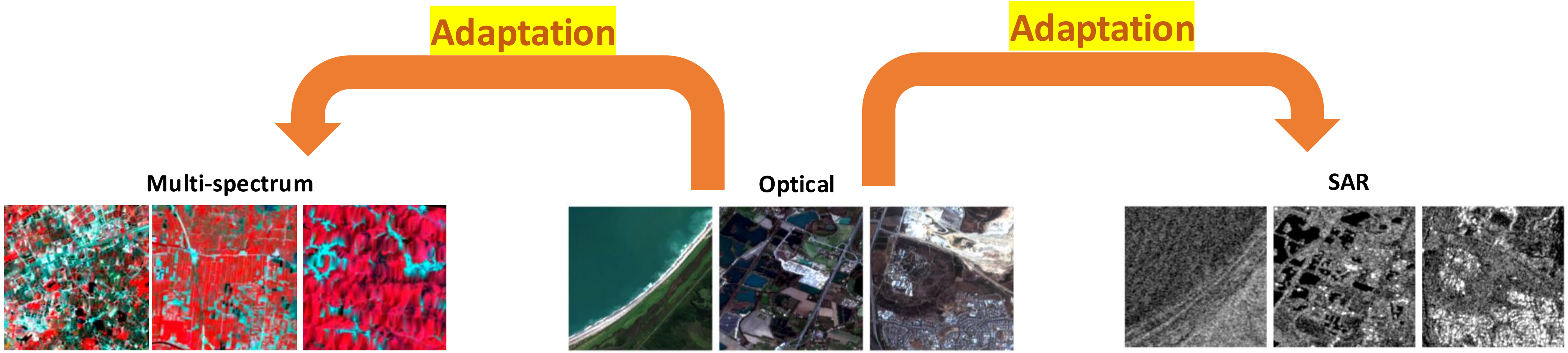
Zichen

Zhaozheng



Qianru

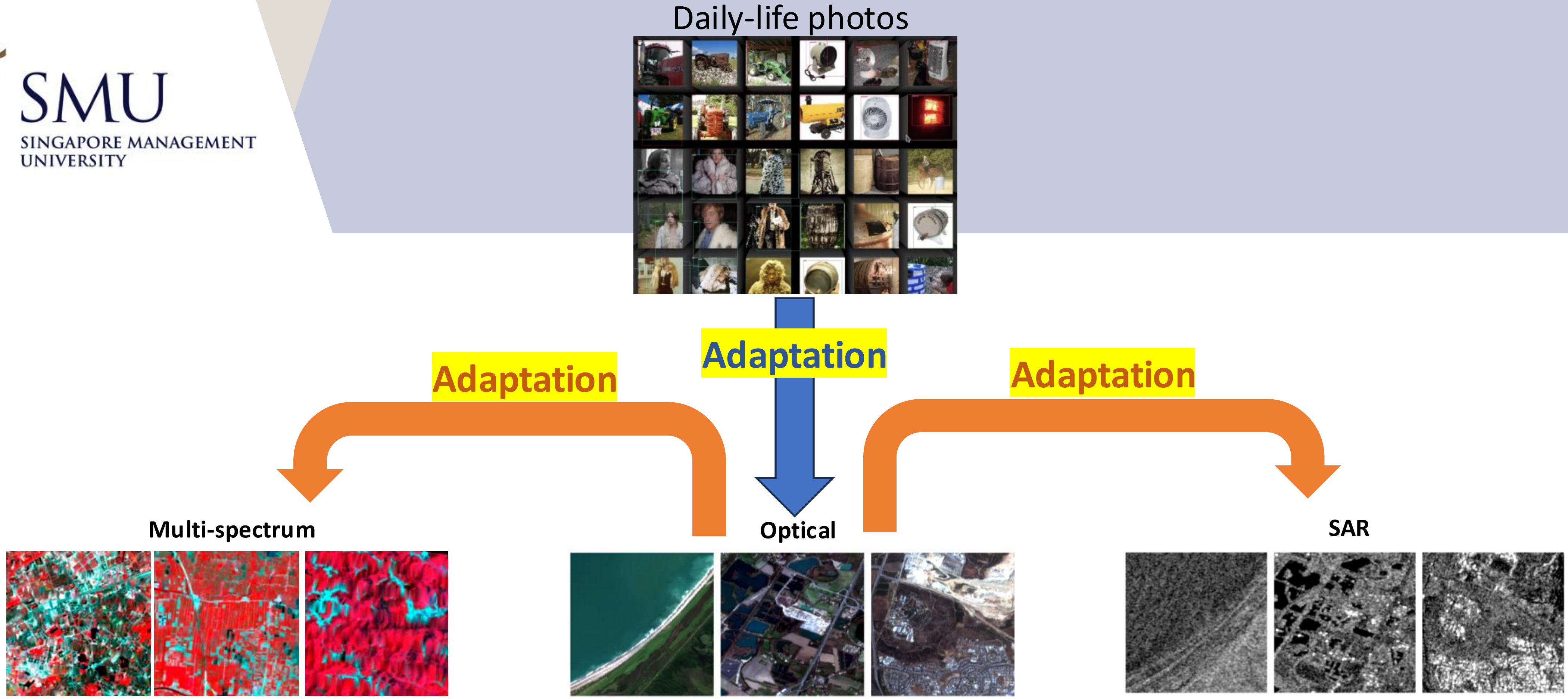
Conference on Neural Information Processing Systems 2024 (NeurIPS’24)



Small Scale Datasets

Large Scale Datasets

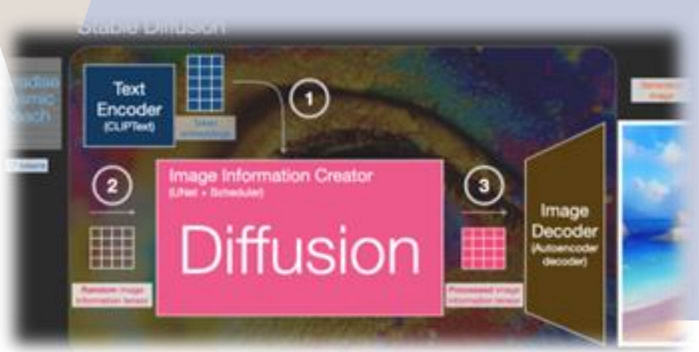
Super Small Scale Datasets



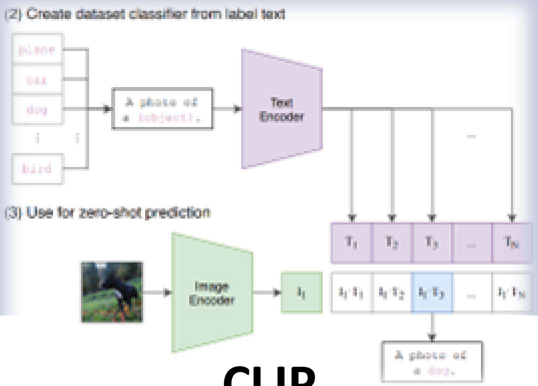
Small Scale Datasets

Large Scale Datasets

Super Small Scale Datasets



SD



CLIP

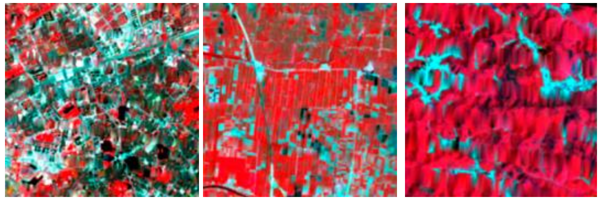
and many other models…

Adaptation

Adaptation

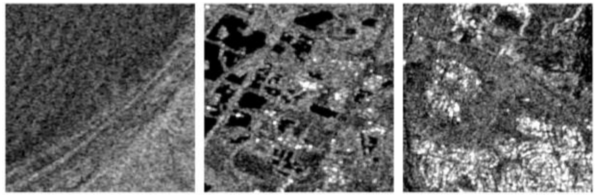
Adaptation

Multi-spectrum



Optical

SAR



Small Scale Datasets

Large Scale Datasets

Super Small Scale Datasets

Learning De-Biased Representations for Remote-Sensing Imagery, Tian et al., NeurIPS 2024

Presentation Outline

Background & Motivation

- Challenges in RS domain
- Limits of Existing Methods
- Motivation of using PEFT

Experimental Results

- Ablation Studies
- Hyperparameter Studies
- Multiple Adaptation Settings
- Multiple Tasks

Insights & Design

- Key Observations of PEFT
- Our Framework
- Core Components

Future Directions

Background & Motivation

Challenges in RS domain • Current Solutions & Limits • Our Key Observations

What is Remote Sensing, and why research in this field is crucial.

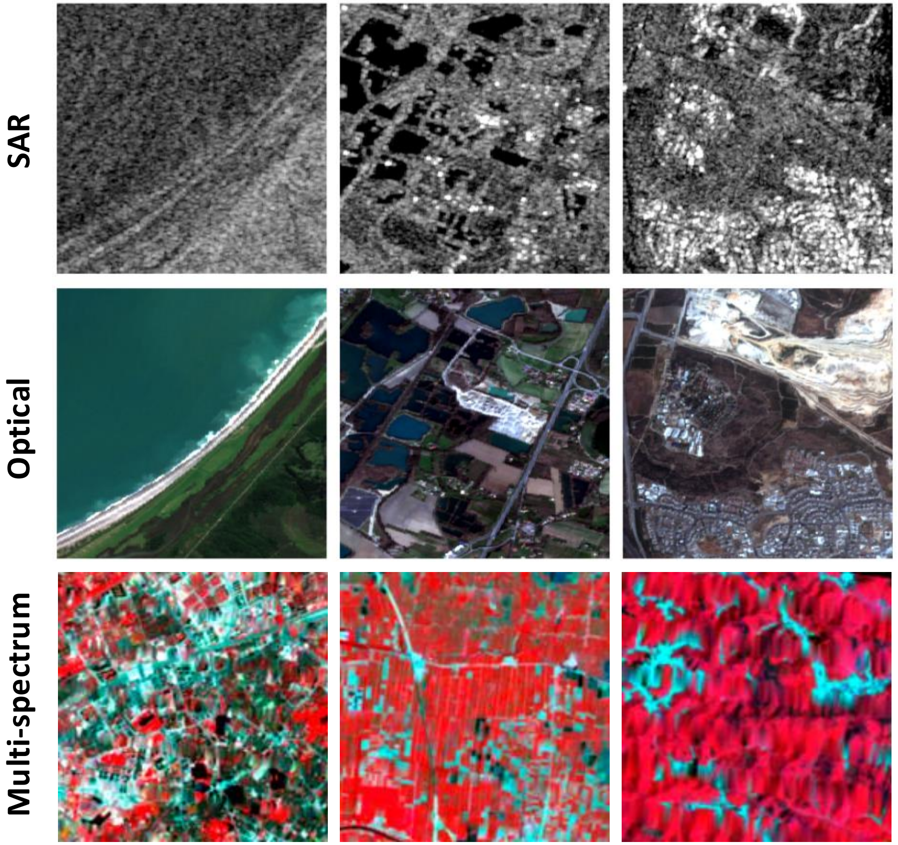
Remote Sensing Domain

- **Definition**

  Remote sensing images are captured from an overhead perspective by spaceborne or airborne sensors, which present unique viewpoints compared to natural images.

- **Multiple Spectrums**
  - Optical RS (ORS): 400-700nm
  - Multi-spectral RS (MSRS): 400-2500nm
  - Synthetic Aperture Radar (SAR): 1mm-1m

- **Key Applications**
  - Environmental monitoring
  - Resource management
  - Disaster response



Source: EUSI Database

What is Remote Sensing, and why research in this field is crucial.

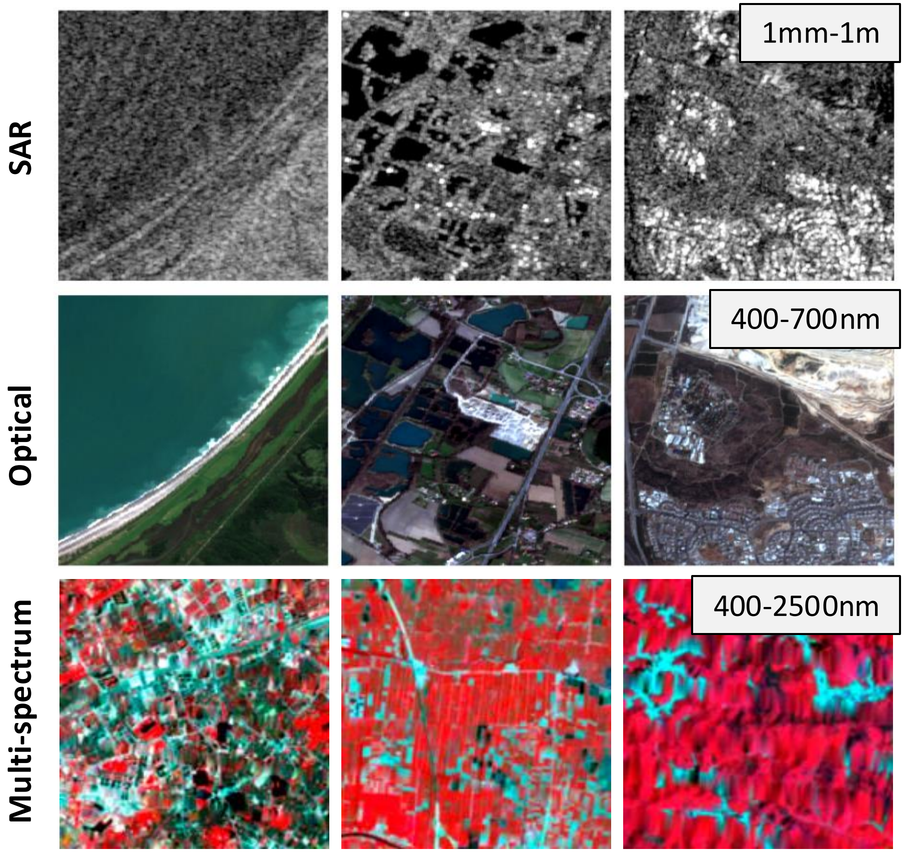
Remote Sensing Domain

- **Definition**

  Remote sensing images are captured from an overhead perspective by spaceborne or airborne sensors, which present unique viewpoints compared to natural images.

- **Multiple Spectrums**
  - Optical RS (ORS): 400-700nm
  - Multi-spectral RS (MSRS): 400-2500nm
  - Synthetic Aperture Radar (SAR): 1mm-1m

- **Key Applications**
  - Environmental monitoring
  - Resource management
  - Disaster response



Source: EUSI Database

What is Remote Sensing, and why research in this field is crucial.

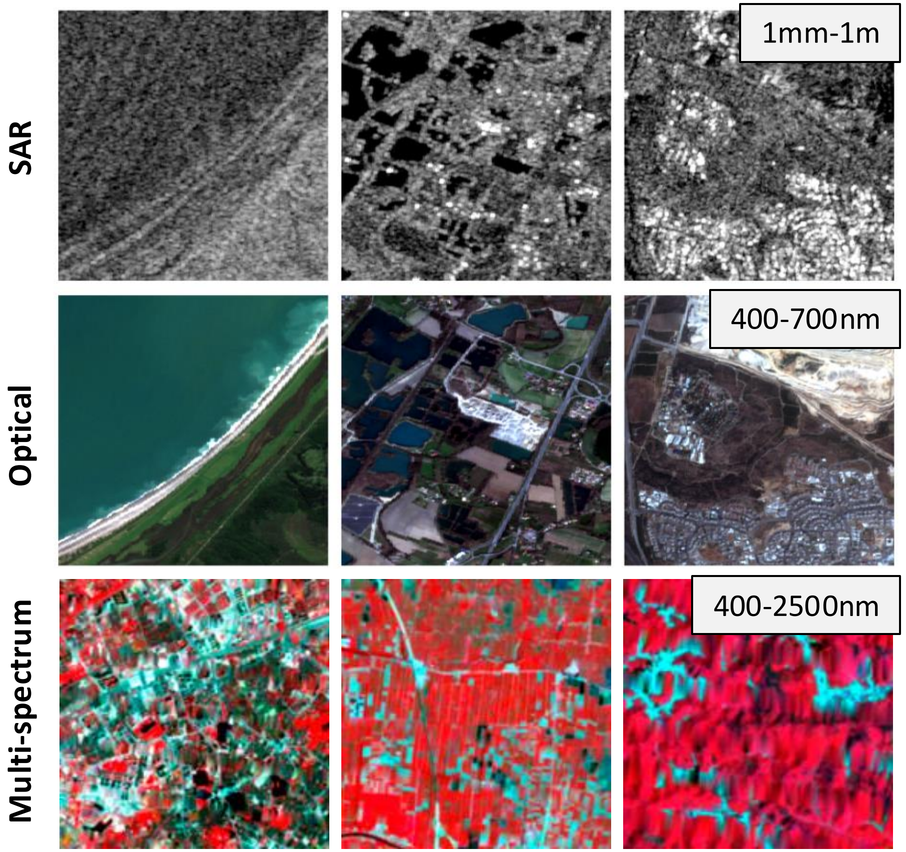
Remote Sensing Domain

- **Definition**

  Remote sensing images are captured from an overhead perspective by spaceborne or airborne sensors, which present unique viewpoints compared to natural images.

- **Multiple Spectrums**
  - Optical RS (ORS): 400-700nm
  - Multi-spectral RS (MSRS): 400-2500nm
  - Synthetic Aperture Radar (SAR): 1mm-1m

- **Key Applications**
  - Environmental monitoring
  - Resource management
  - Disaster response

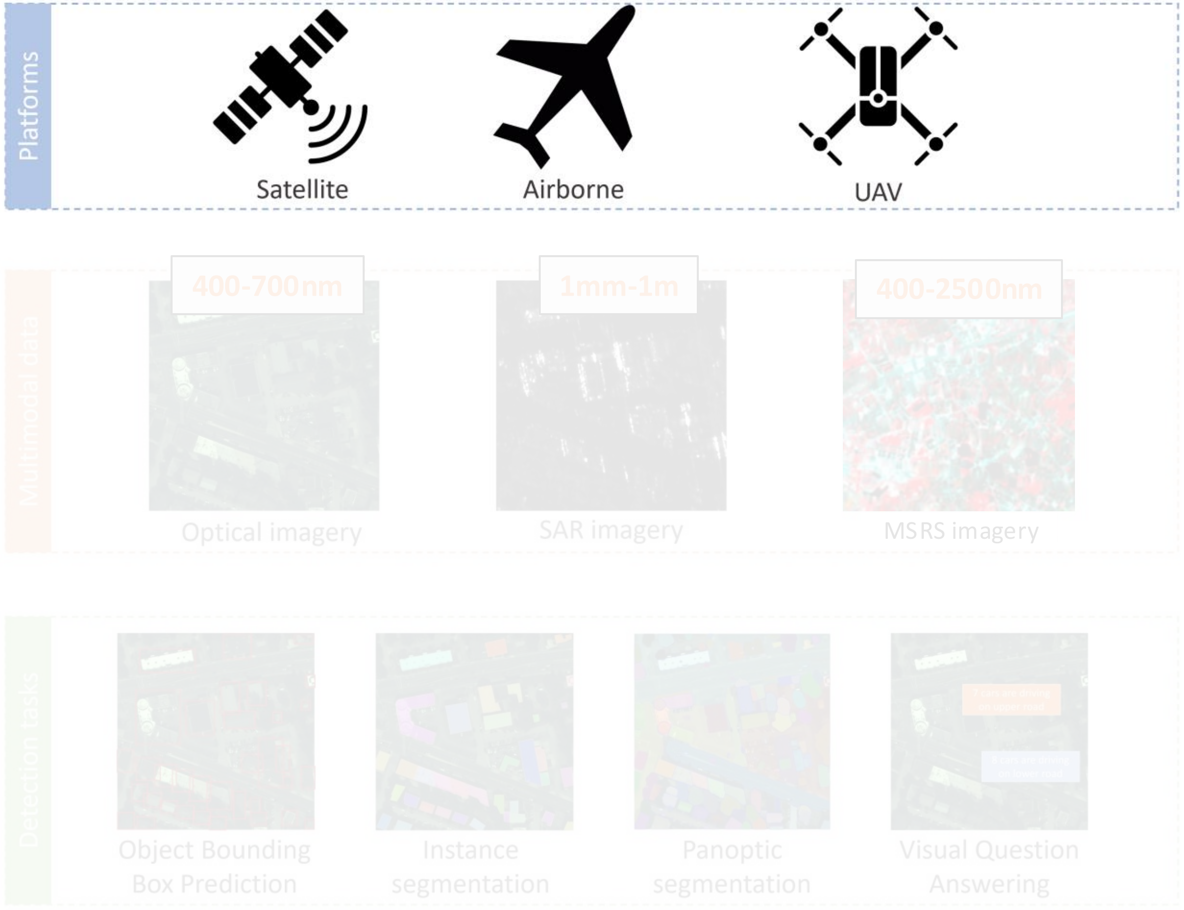


Source: EUSI Database

Remote Sensing data are diverse and complex, requiring heavy processing costs.

Challenges in RS Data

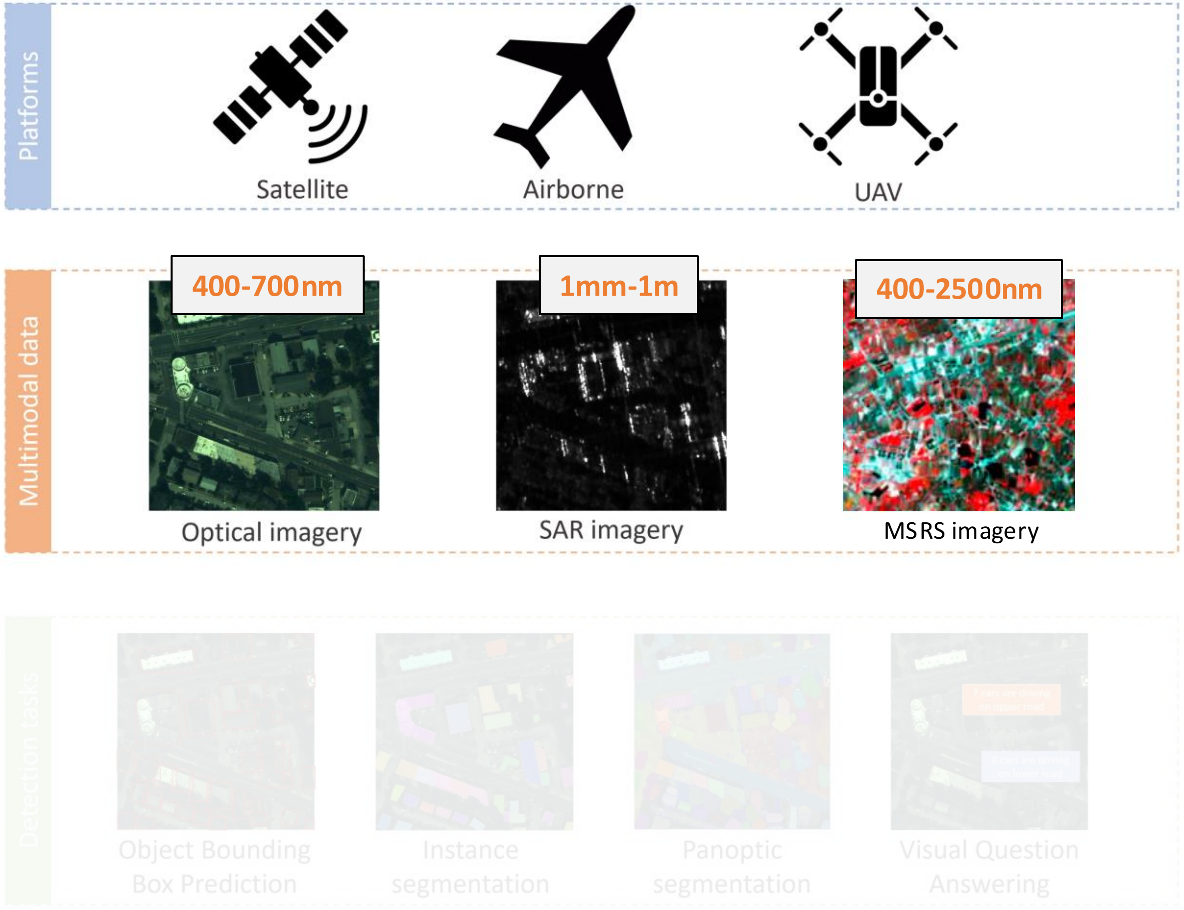
- **RS Data Diversity and Complexity**
  - Various data **source & processing tech**
  - Various spectrums
  - Various downstream tasks



Remote Sensing data are diverse and complex, requiring heavy processing costs.

Challenges in RS Data

- **RS Data Diversity and Complexity**
  - Various data **source & processing tech**
  - Various **spectrums**
  - Various downstream tasks

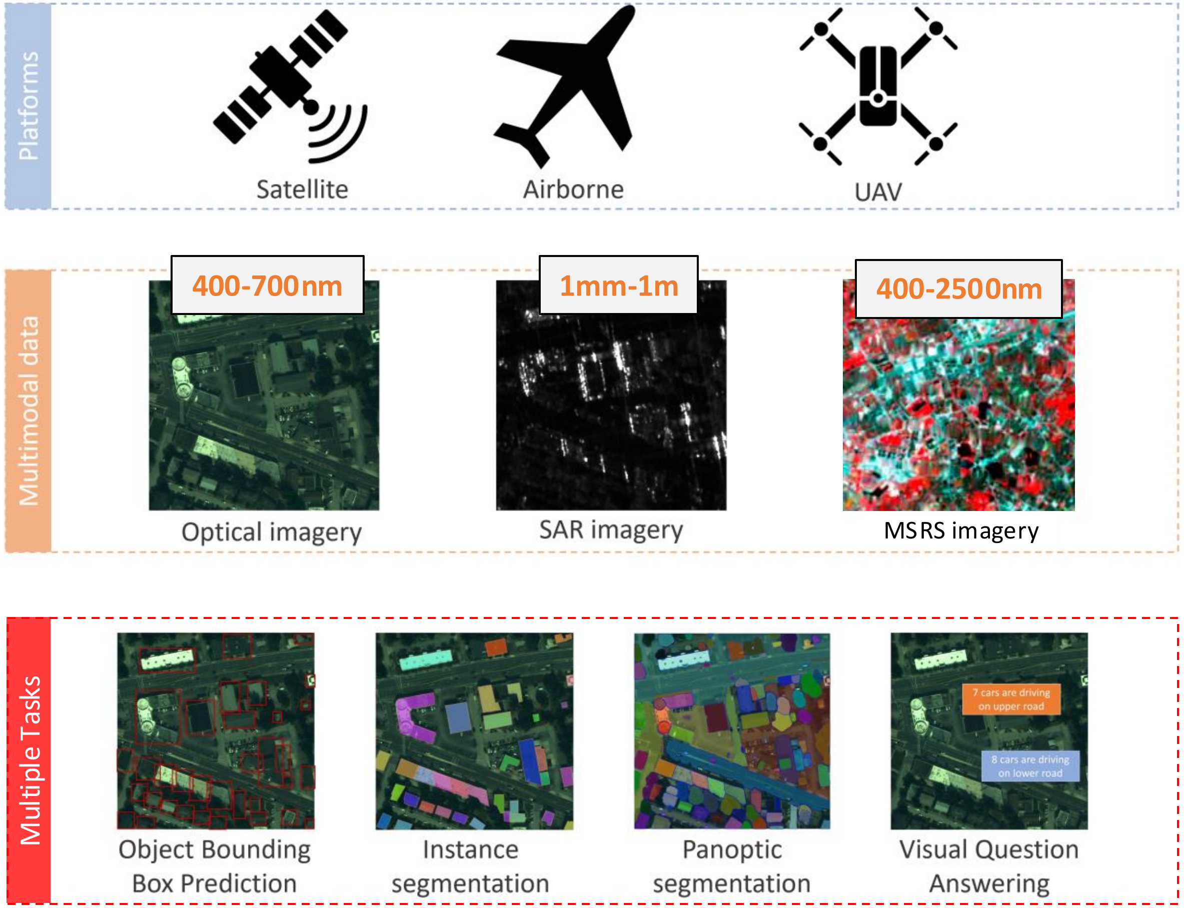


Learning De-Biased Representations for Remote-Sensing Imagery, Tian et al., NeurIPS 2024

Remote Sensing data are diverse and complex, requiring heavy processing costs.

Challenges in RS Data

- **RS Data Diversity and Complexity**
  - Various data **source & processing tech**
  - Various **spectrums**
  - Various downstream **tasks**



Learning De-Biased Representations for Remote-Sensing Imagery, Tian et al., NeurIPS 2024

Remote Sensing data are diverse and complex, requiring heavy processing costs.

Challenges in RS Data

Learning **robust and generic representations** is desirable!

Why not training from scratch?

Parameter Efficient **Transfer Learning**

- **Self-supervised Training from Scratch**
  - Data scarcity in certain spectrums (*e.g.*, **SAR** imagery)
  - Constraints in **model scale** and **data scale**
  - Constraints in **training GPU time**

Year	Dataset Name	Samples	Polarization
2019	AIR-SARShip-1.0/2.0	31,300	-
2019	SEN12MS	180,662	Dual-pol
2019	PolSF	3,000	Full-pol
2019	SAR-Ship	43,819	-
2019	ShipDataset	39,729	HH,VV,VH,HV
2020	HRSID	5,604	HH,HV,VH,VV
2020	So2Sat LCZ42	400,673	Dual-pol
2020	FUSAR-Ship	5,000	-
2020	OpenSARUrban	33,358	Dual-pol
2020	MSAW	48,000	Quad-pol
2022	MSAR	30,158	HH,HV,VH,VV
2022	SADD	883	HH
2023	SAR-AIRcraft	18,888	Uni-polar
2023	OGSOD	18,331	VV/VH
2023	SIVED	1,044	VV/HH
2023	SARDet-100k	116,598	Multiple
	TOTAL	**977,047**	

Table: **High-quality SAR data is scarce.** Only open-sourced datasets released after 2018 are listed. The data acquisition mode (*i.e.*, polarization) vary greatly among datasets.

Learning De-Biased Representations for Remote-Sensing Imagery, Tian et al., NeurIPS 2024

Why not training from scratch?

Parameter Efficient **Transfer Learning**

- **Self-supervised Training from Scratch**
  - Data scarcity in certain spectrums (*e.g.*, **SAR** imagery)
  - Constraints in **model scale** and **data scale**
  - Constraints in **training GPU time**

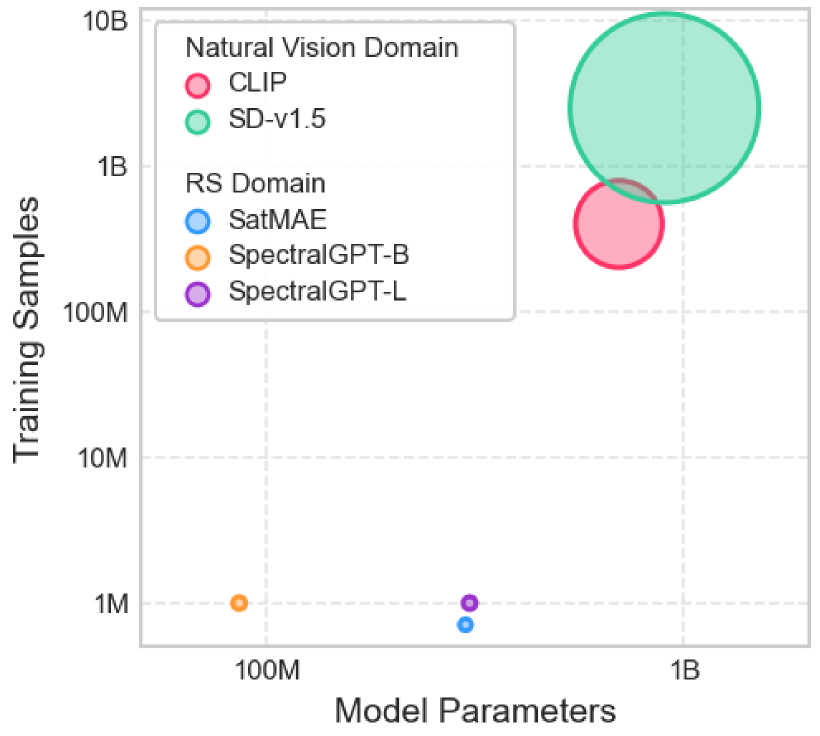


Figure: **Compare foundation models.** The bubble figure shows model scale, data scale and training time of five representative foundation models. Bubble size indicates training GPU-hour. Models from RS domain are much smaller in both model and data scale compared to natural vision domain.

Learning De-Biased Representations for Remote-Sensing Imagery, Tian et al., NeurIPS 2024

Why not training from scratch?

Parameter Efficient **Transfer Learning**

- **Self-supervised Training from Scratch**
    - Data scarcity in certain spectrums (*e.g.*, **SAR** imagery)
    - Constraints in **model scale** and **data scale**
    - Constraints in **training GPU time**

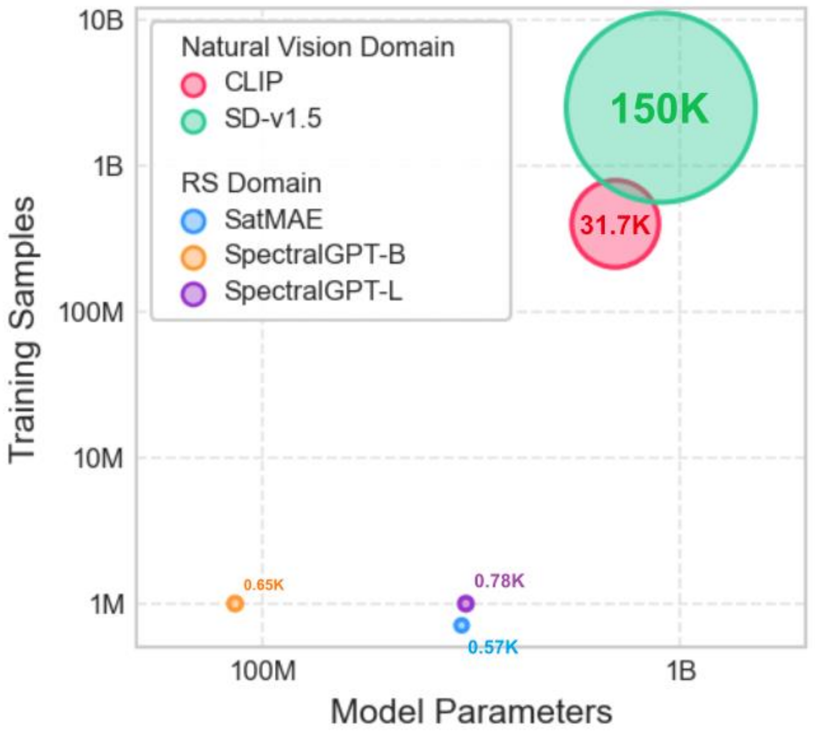
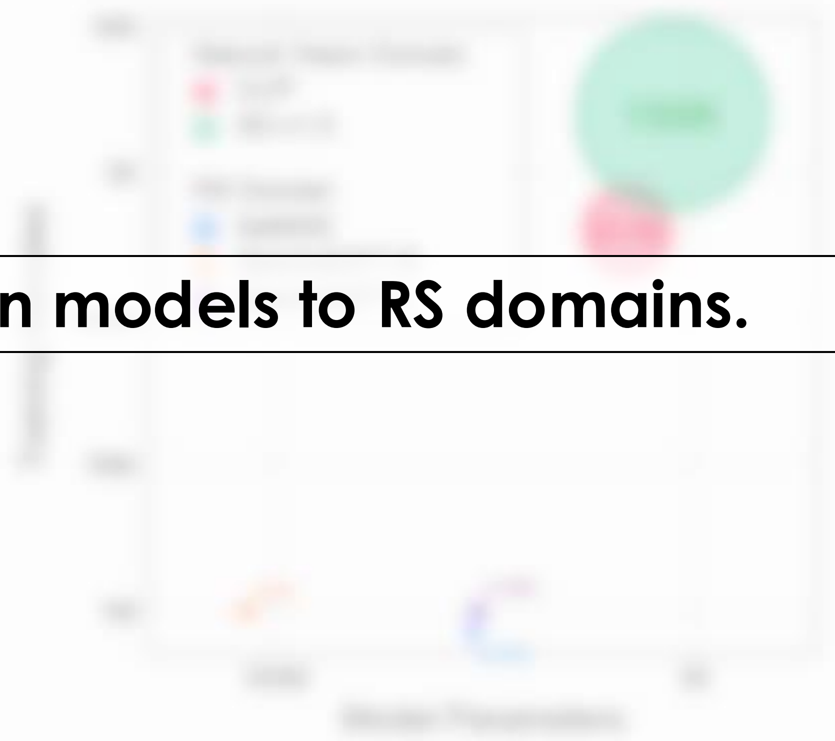


Figure: **Compare foundation models.** The bubble figure shows model scale, data scale and training time of five representative foundation models. Numbers near to bubbles are training GPU-hour. Models from RS domain uses less training GPU-hours compared with natural vision domain.

Why not training from scratch?

Parameter Efficient **Transfer Learning**



We propose to **transfer existing foundation models to RS domains.**

Why do we need parameter efficient?

Parameter Efficient Transfer Learning

- **Transfer Learning Setups**
  - Adaptation **from natural vision domain to RS domain**
  - Adaptation **between RS spectrums**
- **Zero-Shot and Fine-tuning**
  - Fine-tuning suffers from 1) catastrophic forgetting, 2) long training time, and 3) high VRAM usage.
  - Even zero-shot outperforms fine-tuning.
- **Parameter Efficient Transfer Learning (PEFT)**
  - **LoRA - Low Rank Adaptation**
  - Both fine-tuning, zero-shot and PEFT suffers from long-tailed distribution issue.

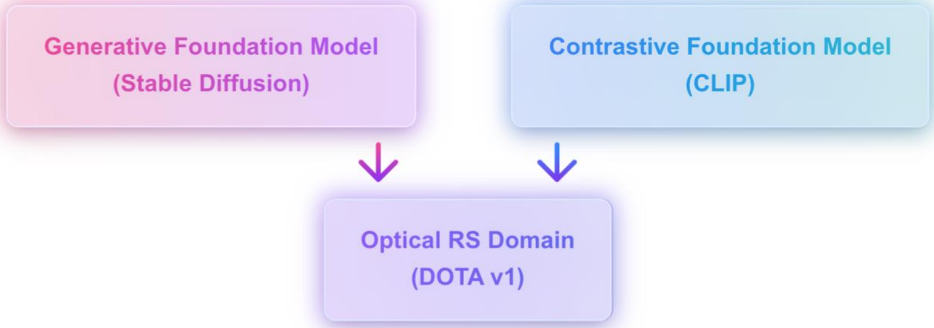


Figure: **Adaptation settings from natural vision domain to RS domain.**

We select two representative models in generative and contrastive arch (*i.e.*, Stable Diffusion v1.5 and CLIP) as source model, and transfer to optical RS domain (*i.e.*, target dataset DOTA v1).

Why do we need parameter efficient?

Parameter Efficient Transfer Learning

- **Transfer Learning Setups**
  - Adaptation **from natural vision domain to RS domain**
  - Adaptation **between RS spectrums**
- **Zero-Shot and Fine-tuning**
  - Fine-tuning suffers from 1) catastrophic forgetting, 2) long training time, etc.
  - Even zero-shot outperforms fine-tuning.
- **Parameter Efficient Transfer Learning (PEFT)**
  - **LoRA - Low Rank Adaptation**
  - Both fine-tuning, zero-shot and PEFT suffers from long-tailed distribution issue.

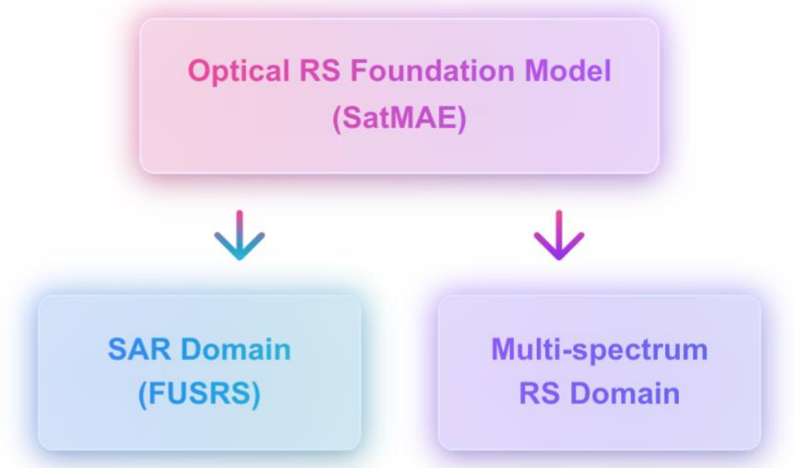


Figure: **Adaptation settings between RS spectrums.**

We transfer knowledge from optical RS foundation model (*i.e.*, SatMAE) to two data-scarce domains: SAR and MSRS imagery.

Why do we need parameter efficient?

Parameter Efficient Transfer Learning

- **Transfer Learning Setups**
  - Adaptation **from natural vision domain to RS domain**
  - Adaptation **between RS spectrums**
- **Zero-Shot and Fine-tuning**
  - Fine-tuning suffers from 1) catastrophic forgetting, 2) long training time, etc.
  - Even zero-shot outperforms fine-tuning.
- **Parameter Efficient Transfer Learning (PEFT)**
  - **LoRA - Low Rank Adaptation**
  - Both fine-tuning, zero-shot and PEFT suffers from long-tailed distribution issue.

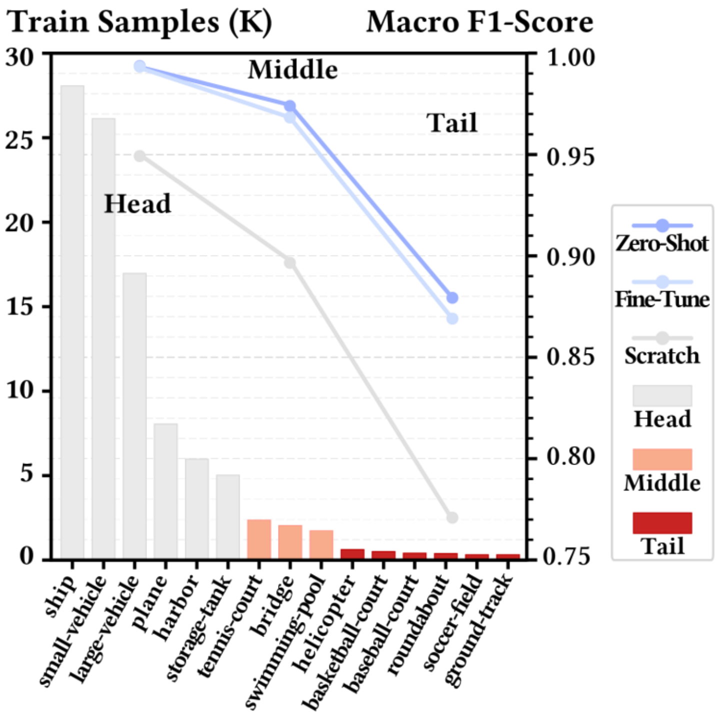


Figure: **Performance of SD to ORS adaptation.** Representations evaluated by linear probing. “Scratch” means supervised learning from scratch.

By comparing zero-shot and fine-tuning, we could conclude that fine-tuning suffers from catastrophic forgetting issue.

Why do we need parameter efficient?

Parameter Efficient Transfer Learning

- **Transfer Learning Setups**
  - Adaptation **from natural vision domain to RS domain**
  - Adaptation **between RS spectrums**
- **Zero-Shot and Fine-tuning**
  - Fine-tuning suffers from 1) catastrophic forgetting, 2) long training time, etc.
  - Even zero-shot outperforms fine-tuning.
- **Parameter Efficient Transfer Learning (PEFT)**
  - **LoRA[1] - Low Rank Adaptation**
  - Both fine-tuning, zero-shot and PEFT suffers from long-tailed distribution issue.

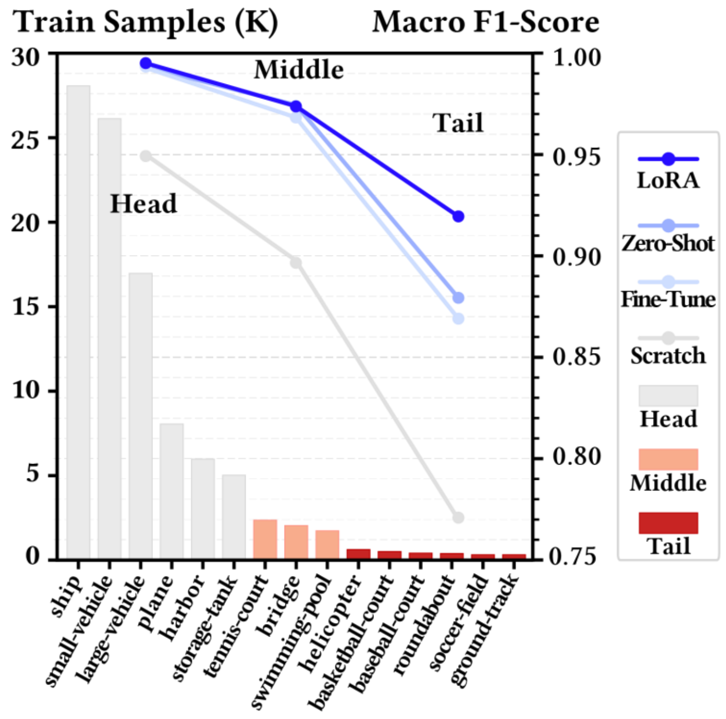


Figure: **Performance of Natural to ORS adaptation setting.**

LoRA achieves the best performance, especially on tail classes.

[1] Lora: Low-rank adaptation of large language models, Hu, Edward J., et al., ArXiv (2021)

Why do we need parameter efficient?

LoRA – Low Rank Adaptation

- **Rank** – The linearly independent column numbers (or row numbers) in a matrix.
- **Low-Rank Matrix in Neural Network** – For a given neural layer with params matrix $\boldsymbol{\theta}_{n\times k}$, the rank of this matrix can be considered "the dimensions of representation space". It is updated by an updating matrix $\Delta\boldsymbol{\theta}_{n\times k}$:

$$\boldsymbol{\theta}_{n\times k} + \Delta\boldsymbol{\theta}_{n\times k} \,,$$

- **Low-Rank Decomposition** – Generally, this updating matrix $\Delta\boldsymbol{W}_{n\times k}$ is sparse. Thus, instead of updating the whole $n \times k$ matrix, we could decompose $\Delta\boldsymbol{W}_{n\times k}$ into two low-rank dense matrixes $\boldsymbol{A}$ and $\boldsymbol{B}$:

$$\Delta\boldsymbol{W}_{n\times k} = \boldsymbol{B}_{n\times r}\boldsymbol{A}_{r\times k}$$

Such sparse matrix $\Delta\boldsymbol{\theta}_{n\times k}$ is called "a LoRA module", and r is its rank. Multiple LoRA modules could be weighted combined.

[1] *Lora: Low-rank adaptation of large language models, Hu, Edward J., et al., ArXiv (2021)*

Why do we need parameter efficient?

LoRA – Low Rank Adaptation

- **Rank** – The linearly independent column numbers (or row numbers) in a matrix.
- **Low-Rank Matrix in Neural Network** – For a given neural layer with params matrix $\boldsymbol{\theta}_{n\times k}$, the rank of this matrix can be considered "the dimensions of representation space". It is updated by an updating matrix $\Delta\boldsymbol{\theta}_{n\times k}$:

$$\boldsymbol{\theta}_{n\times k} + \Delta\boldsymbol{\theta}_{n\times k}\ ,$$

- **Low-Rank Decomposition** – Generally, this updating matrix $\Delta\boldsymbol{W}_{n\times k}$ is sparse. Thus, instead of updating the whole $n \times k$ matrix, we could decompose $\Delta\boldsymbol{W}_{n\times k}$ into two low-rank dense matrixes $\boldsymbol{A}$ and $\boldsymbol{B}$:

$$\Delta\boldsymbol{W}_{n\times k} = \boldsymbol{B}_{n\times r}\boldsymbol{A}_{r\times k}$$

Such sparse matrix $\Delta\boldsymbol{\theta}_{n\times k}$ is called "a LoRA module", and r is its rank. Multiple LoRA modules could be weighted combined.

[1] Lora: Low-rank adaptation of large language models, Hu, Edward J., et al., ArXiv (2021)

Why do we need parameter efficient?

P...

LoRA – Low Rank Adaptation

- **Rank** – The linearly independent column numbers (or row numbers) in a matrix.
- **Low-Rank Matrix in Neural Network** – For a given neural layer with params matrix $\boldsymbol{\theta}_{n\times k}$, the rank of this matrix can be considered "the dimensions of representation space". It is updated by an updating matrix $\Delta\boldsymbol{\theta}_{n\times k}$:

$$\boldsymbol{\theta}_{n\times k} + \Delta\boldsymbol{\theta}_{n\times k}\ ,$$

- **Low-Rank Decomposition** – Generally, this updating matrix $\Delta\boldsymbol{W}_{n\times k}$ is sparse. Thus, instead of updating the whole $n \times k$ matrix, we could decompose $\Delta\boldsymbol{W}_{n\times k}$ into two low-rank dense matrixes $\boldsymbol{A}$ and $\boldsymbol{B}$:

$$\Delta\boldsymbol{W}_{n\times k} = \boldsymbol{B}_{n\times r}\boldsymbol{A}_{r\times k}$$

Such sparse matrix $\Delta\boldsymbol{\theta}_{n\times k}$ is called "a LoRA module", and r is its rank. Multiple LoRA modules could be weighted combined.

Why do we need parameter efficient?

P...

LoRA – Low Rank Adaptation

- **Rank** – The linearly independent column numbers (or row numbers) in a matrix.
- **Low-Rank Matrix in Neural Network** – For a given neural layer with params matrix $\boldsymbol{\theta}_{n\times k}$, the rank of this matrix can be considered "the dimensions of representation space". It is updated by an updating matrix $\Delta\boldsymbol{\theta}_{n\times k}$:

$$\boldsymbol{\theta}_{n\times k} + \Delta\boldsymbol{\theta}_{n\times k}\ ,$$

- **Low-Rank Decomposition** – Generally, this updating matrix $\Delta\boldsymbol{W}_{n\times k}$ is sparse. Thus, instead of updating the whole $n \times k$ matrix, we could decompose $\Delta\boldsymbol{W}_{n\times k}$ into two low-rank dense matrixes $\boldsymbol{A}$ and $\boldsymbol{B}$:

$$\Delta\boldsymbol{W}_{n\times k} = \boldsymbol{B}_{n\times r}\boldsymbol{A}_{r\times k}$$

Such sparse matrix $\Delta\boldsymbol{\theta}_{n\times k}$ is called "a LoRA module", and r is its rank. Multiple LoRA modules could be weighted combined.

Why do we need parameter efficient?

Parameter Efficient Transfer Learning

- **Transfer Learning Setups**
  - Adaptation **from natural vision domain to RS domain**
  - Adaptation **between RS spectrums**
- **Zero-Shot and Fine-tuning**
  - Fine-tuning suffers from 1) catastrophic forgetting, 2) long training time, etc.
  - Even zero-shot outperforms fine-tuning.
- **Parameter Efficient Fine-Tuning (PEFT)**
  - **LoRA - Low Rank Adaptation**
  - Both fine-tuning, zero-shot and PEFT suffers from **long-tailed** distribution issue.

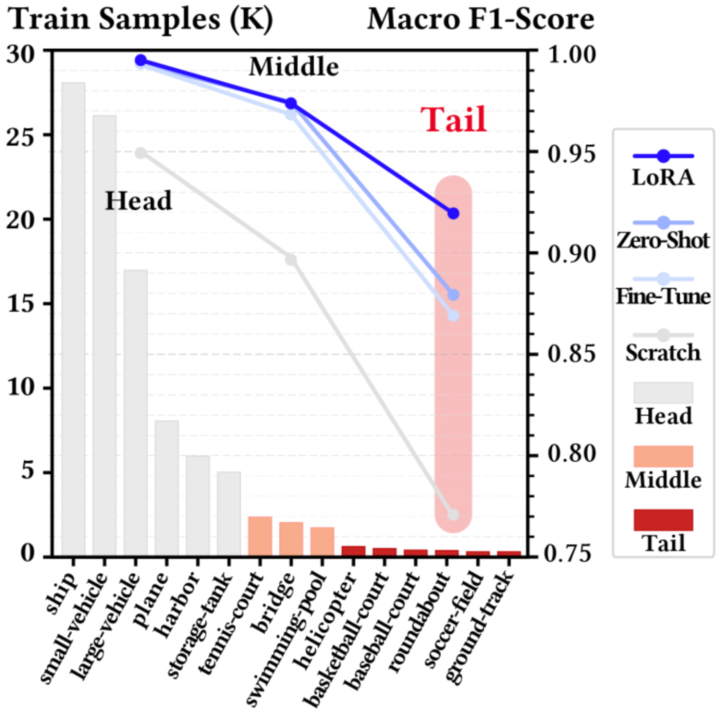


Figure: **Performance of Natural to ORS adaptation setting.** The debLoRA achieves highest performance, especially for tail class.

Insights & Design

Key Observations • Framework • Core Components • Algorithm Explanation

SMU SINGAPORE MANAGEMENT UNIVERSITY

We observe that representation space learnt by PEFT methods are biased.

Key Observation – Biased Representation Space

- **Biased Representation Space**
  - When learnt on long-tailed data, LoRA's adapted **feature space** of LoRA **is biased[2]**.
  - Validation samples of **head class** are mostly **correctly** classified.
  - Validation samples of **tail class** are **wrongly** classified as head class.
  - Key issue: **Train/Val distribution mismatch** for tail classes.

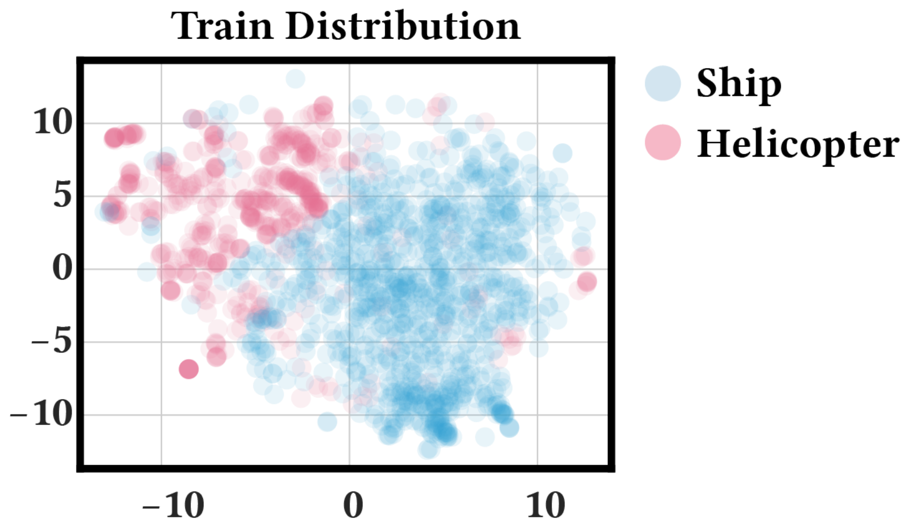


Figure: **Feature distribution of training samples.** For clearer visualization, we pick representative head class "Helicopter" and tail class "Ship" from DOTA v1 dataset as an example.

[2] We define feature space Z as biased if $Vol(Z_h) \gg Vol(Z_t)$, and $\exists\, z_t \in Z_t: P(z_t \in Z_h) > P(z_t \in Z_t)$, where Z_h and Z_t denotes the feature spaces of head and tail classes respectively, $Vol(\cdot)$ denotes feature space volume, and $P(\cdot)$ denotes the probability distribution predicted by the model.

We observe that representation space learnt by PEFT methods are biased.

Key Observation – Biased Representation Space

- **Biased Representation Space**
  - When learnt on long-tailed data, LoRA's adapted **feature space** of LoRA **is biased**[2].
  - Validation samples of **head class** are mostly **correctly** classified.
  - Validation samples of **tail class** are **wrongly** classified as head class.
  - Key issue: **Train/Val distribution mismatch** for tail classes.

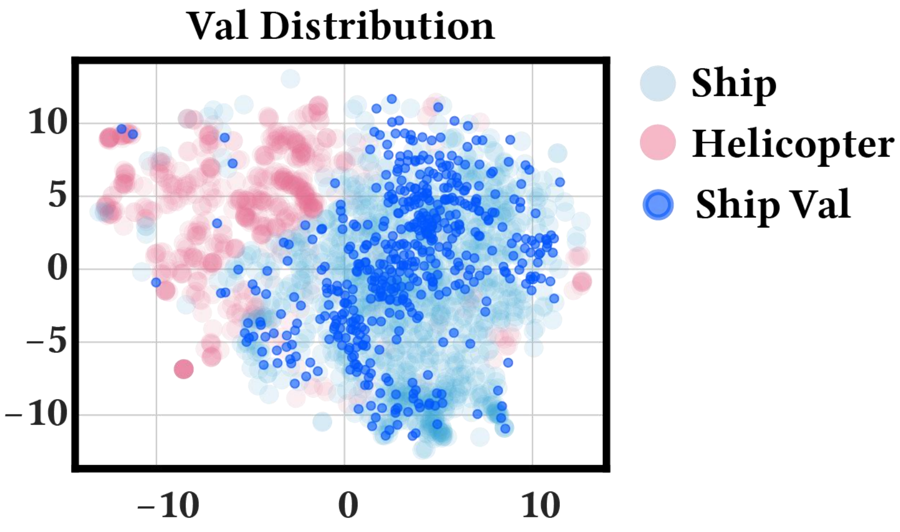


Figure: **Feature distribution of training samples.** For clearer visualization, we pick representative head class "Helicopter" and tail class "Ship" from DOTA v1 dataset as an example.

[2] We define feature space Z as biased if $Vol(Z_h) \gg Vol(Z_t)$, and $\exists\, z_t \in Z_t: P(z_t \in Z_h) > P(z_t \in Z_t)$, where Z_h and Z_t denotes the feature spaces of head and tail classes respectively, $Vol(\cdot)$ denotes feature space volume, and $P(\cdot)$ denotes the probability distribution predicted by the model.

We observe that representation space learnt by PEFT methods are biased.

Key Observation – Biased Representation Space

- **Biased Representation Space**
  - When learnt on long-tailed data, LoRA's adapted **feature space** of LoRA **is biased[2]**.
  - Validation samples of **head class** are mostly **correctly** classified.
  - Validation samples of **tail class** are **wrongly** classified as head class.
  - Key Challenge: **Train/Val distribution mismatch** for tail classes.

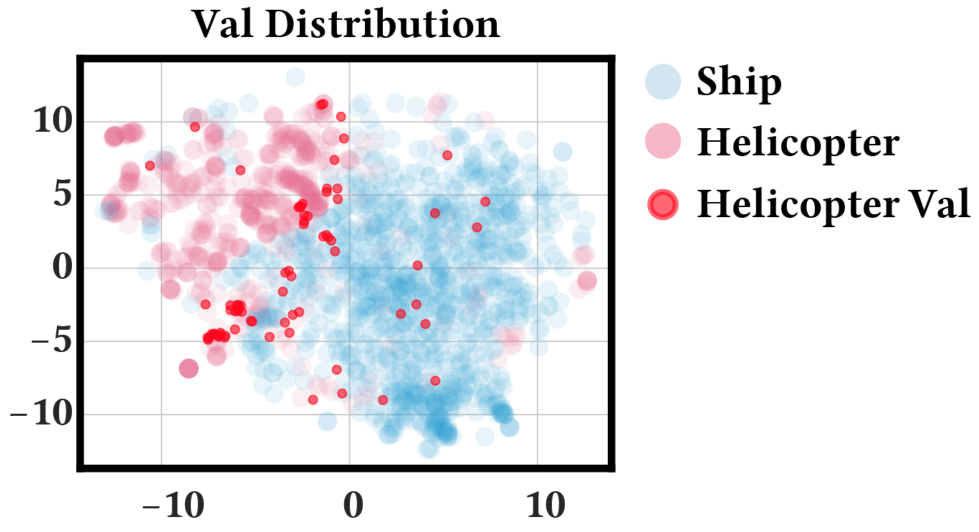


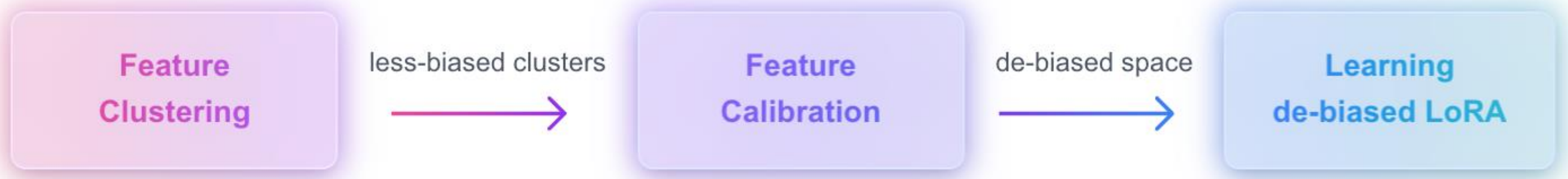
Figure: **Feature distribution of training samples.** For clearer visualization, we pick representative head class "Helicopter" and tail class "Ship" from DOTA v1 dataset as an example.

[2] We define feature space Z as biased if $Vol(Z_h) \gg Vol(Z_t)$, and $\exists\, z_t \in Z_t$: $P(z_t \in Z_h) > P(z_t \in Z_t)$, where Z_h and Z_t denotes the feature spaces of head and tail classes respectively, $Vol(\cdot)$ denotes feature space volume, and $P(\cdot)$ denotes the probability distribution predicted by the model.

Our Framework involves three core components.

Framework of Our Approach

- **Three key components**
  - **Feature clustering** – Unsupervised clustering to find less biased prototypes.
  - **Feature calibration** – Use less-biased prototypes to calibrate tail class features.
  - **debLoRA learning** – Learn a LoRA module to capture this de-bias mapping.



We first found balanced prototypes within feature space.

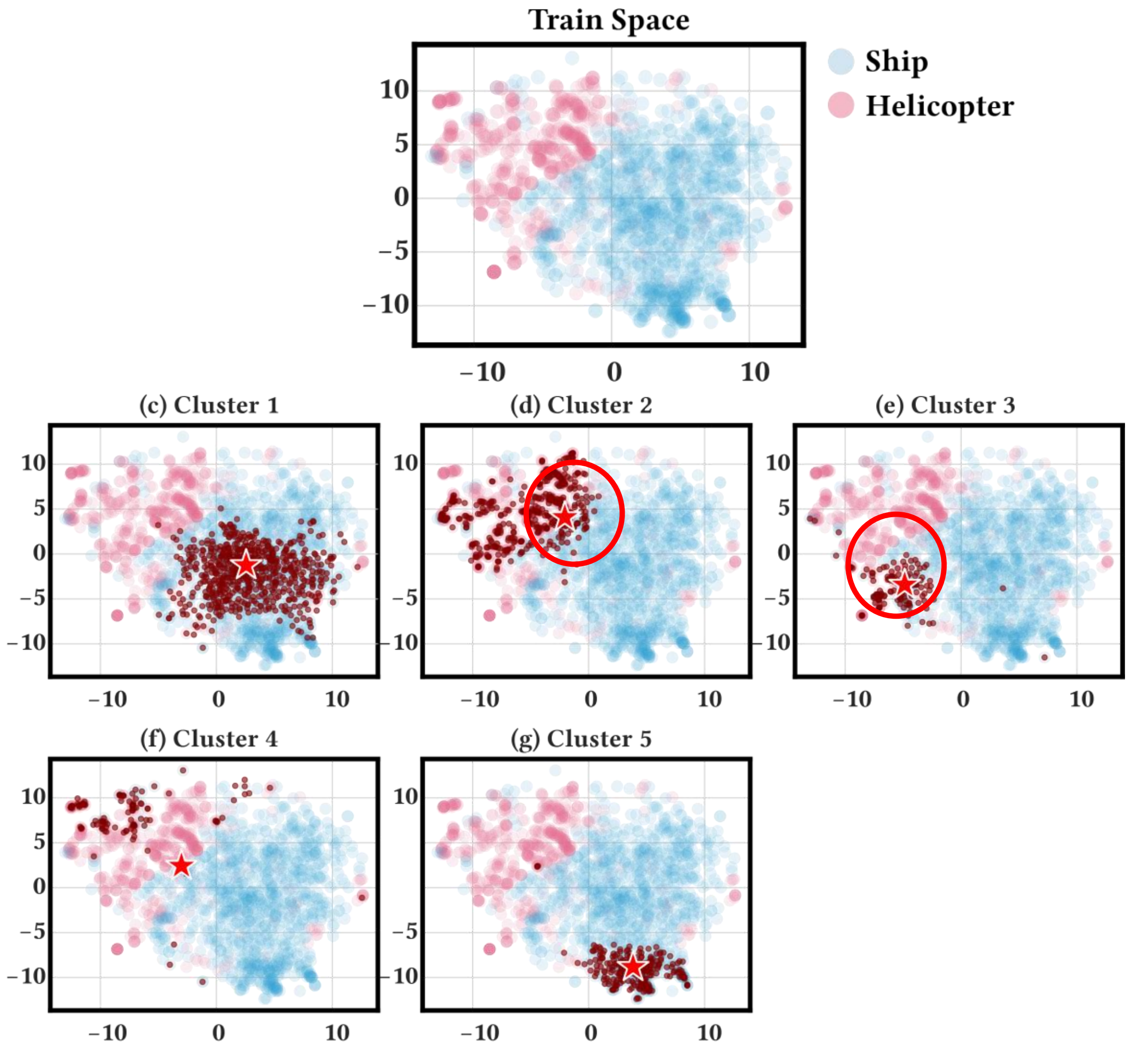
Feature Clustering

- **Feature clustering**
  - We conduct K-Means clustering over training samples' feature space.

$$\min_{\mu_k} \sum_{i=1}^{N} \min_{k} \|z_i - \mu_k\|^2, s.t. \forall k, n_k \geq \frac{N}{K \cdot \rho},$$

where μ_k and n_k denote the center and size of the k-th cluster, respectively.

  - Some cluster centers are contributed by both head and tail classes, and hence is less biased (*e.g.*, clusters 2 and 3).



We secondly construct less biased centers and calibrate features.

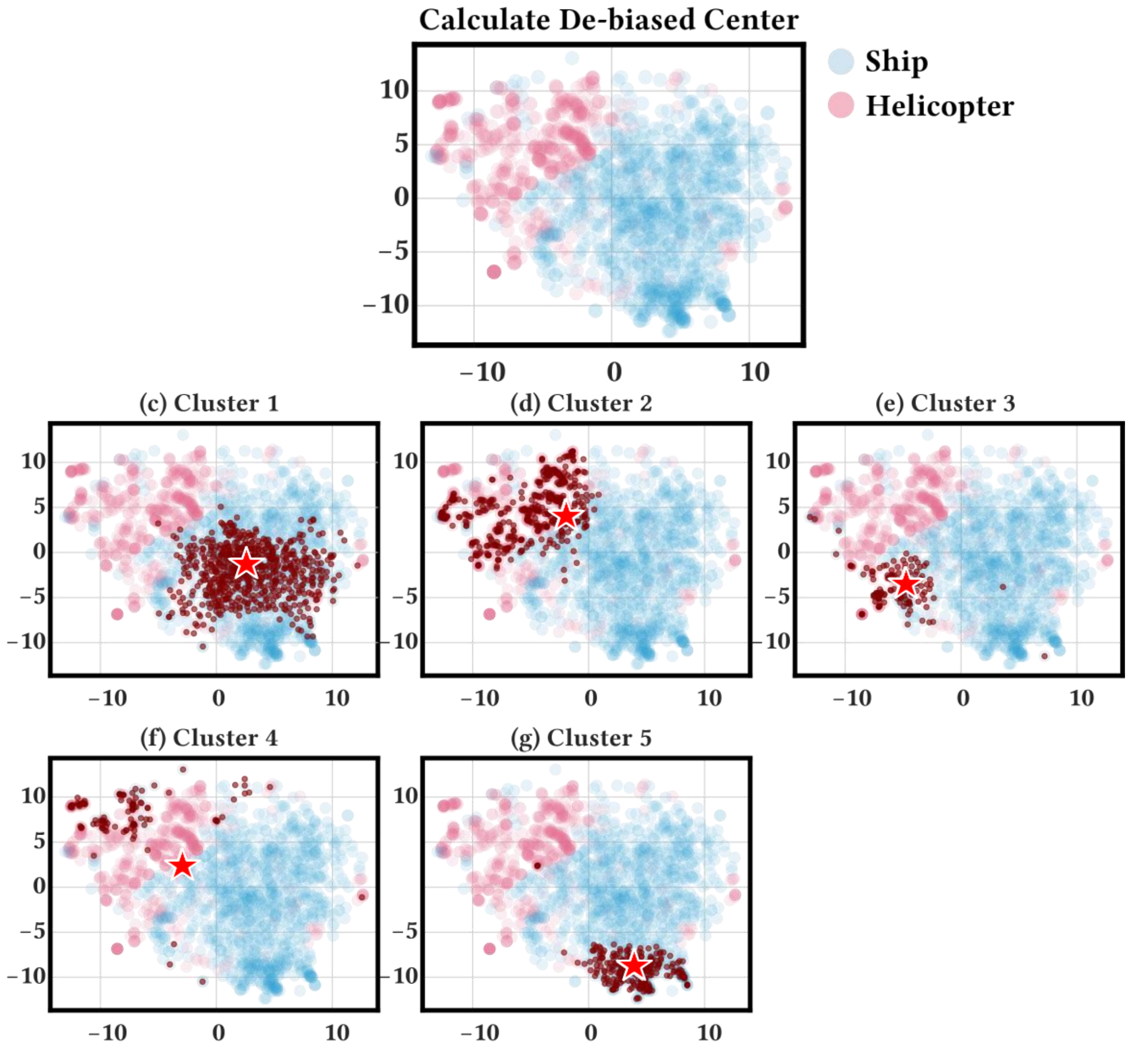
Construct De-Biased Center

- **De-Biased Center**
  - We calculate de-biased representation center for each tail class:

$$\hat{\mu}_c = \sum_k w_k \cdot \mu_k \, , w_k = \frac{n_k}{n_c},$$

here weight w_k proportion to the fraction of class c samples in k-th cluster.

  - This ensures that the de-biased center $\hat{\mu}_c$ is not dominated by head classes



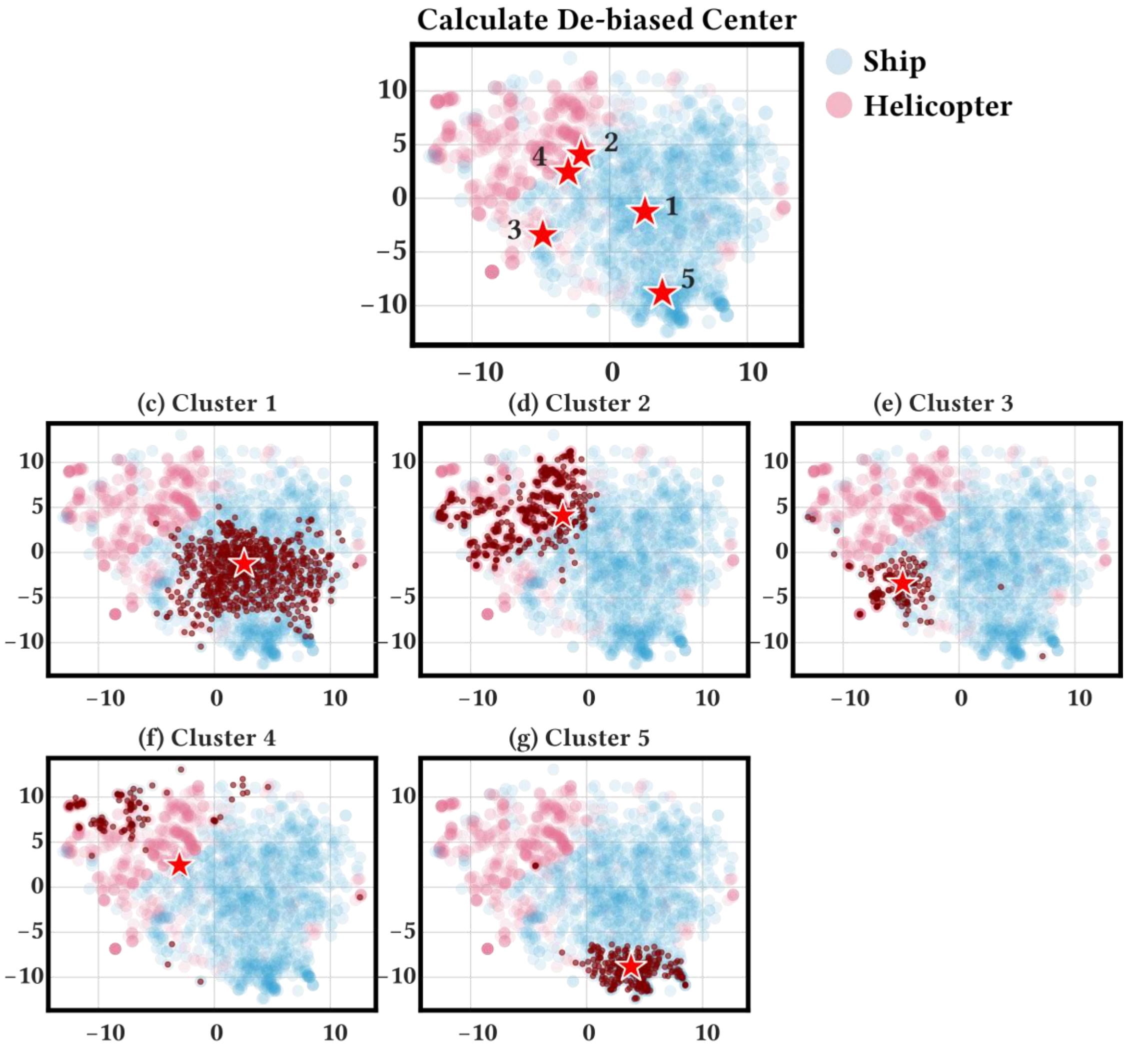
We secondly construct less biased centers and calibrate features.

Construct De-Biased Center

- **De-Biased Center**
  - We calculate de-biased representation center for each tail class:

$$\hat{\mu}_c = \sum_k w_k \cdot \mu_k \, , w_k = \frac{n_k}{n_c},$$

here weight w_k proportion to the fraction of class c samples in k-th cluster.
  - This ensures that the de-biased center $\hat{\mu}_c$ is not dominated by head classes



We secondly construct less biased centers and calibrate features.

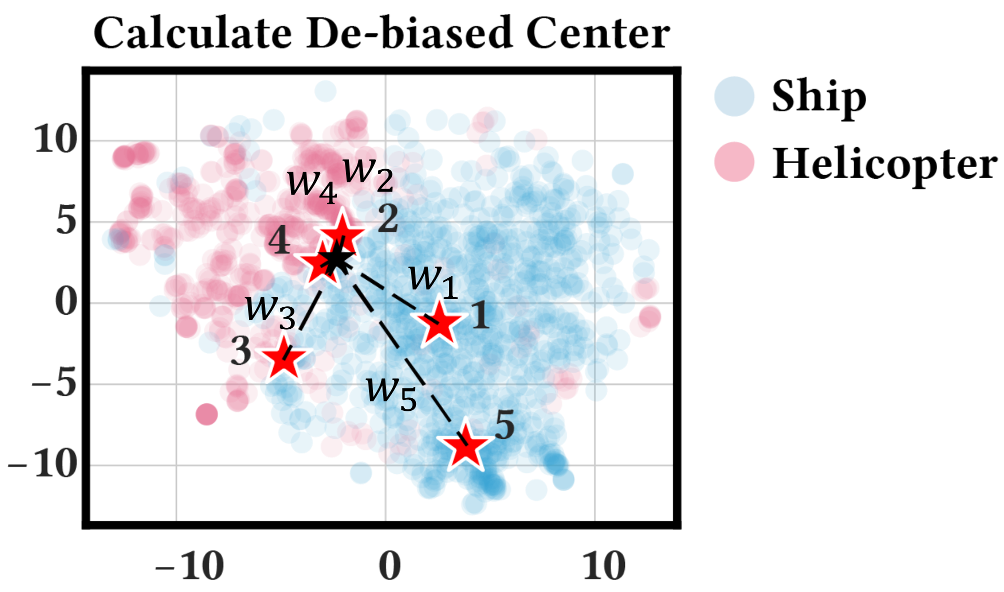
Construct De-Biased Center

- **De-Biased Center**
  - We calculate de-biased representation center for each tail class:

$$\hat{\mu}_c = \sum_k w_k \cdot \mu_k \,, w_k = \frac{n_k}{n_c},$$

here weight w_k proportion to the fraction of class c samples in k-th cluster.

  - This ensures that the de-biased center $\hat{\mu}_c$ is not dominated by head classes



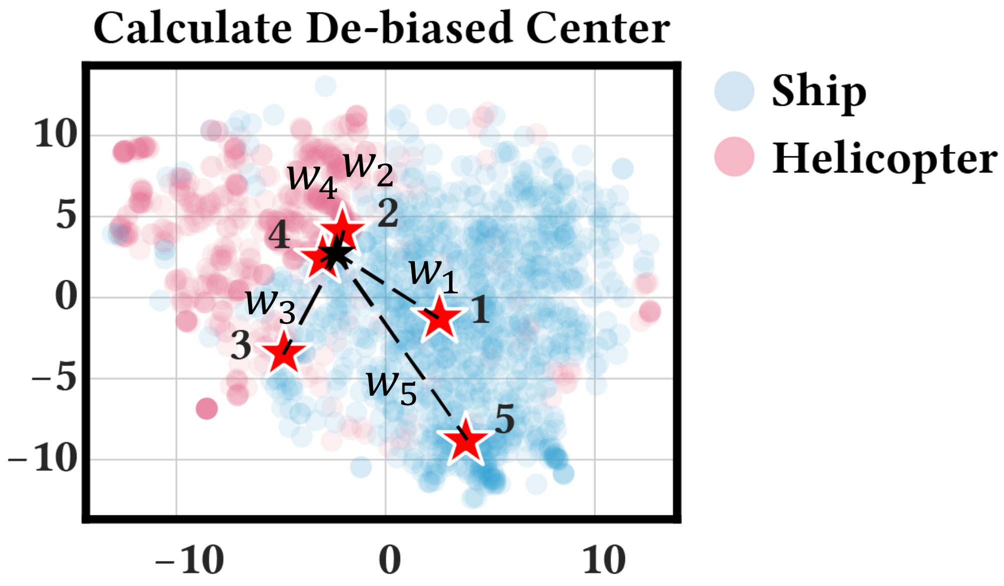
We secondly construct less biased centers and calibrate features.

Construct De-Biased Center

- **De-Biased Center**
  - We calculate de-biased representation center for each tail class:

$$\hat{\mu}_c = \sum_k w_k \cdot \mu_k \,, w_k = \frac{n_k}{n_c},$$

    here weight w_k proportion to the fraction of class c samples in k-th cluster.
  - This ensures that the de-biased center $\hat{\mu}_c$ is not dominated by head classes



We utilize LoRA to capture the de-bias mapping.

Feature Calibration

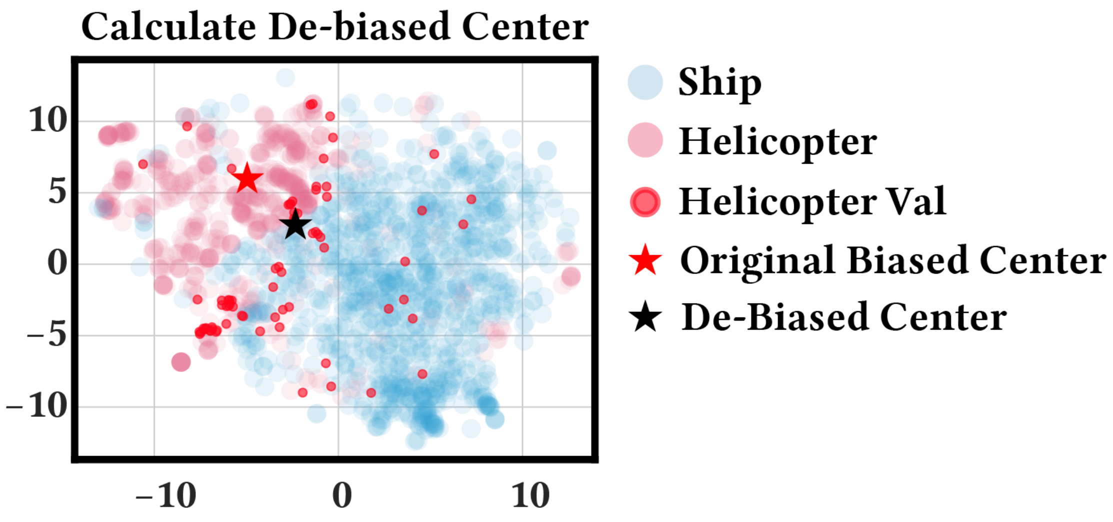
- **Tail Class Calibration**
  - De-Biased Center are **closer to validation** samples.
  - We calibrate tail class features z by moving them close to de-biased center $\hat{\mu}$:

$$\tilde{z} = \alpha z + (1 - \alpha)\hat{\mu},$$

where $\alpha = \min(1, \frac{10}{ir})$ empirically.

- **Learning debLoRA**
  - We learn an LoRA module with training objective

$$\min_{\phi} \frac{1}{D_t} \sum_{x \in D_t} \left\| g_\phi(f_\theta(x)) - \tilde{z} \right\|^2$$



We utilize LoRA to capture the de-bias mapping.

Feature Calibration

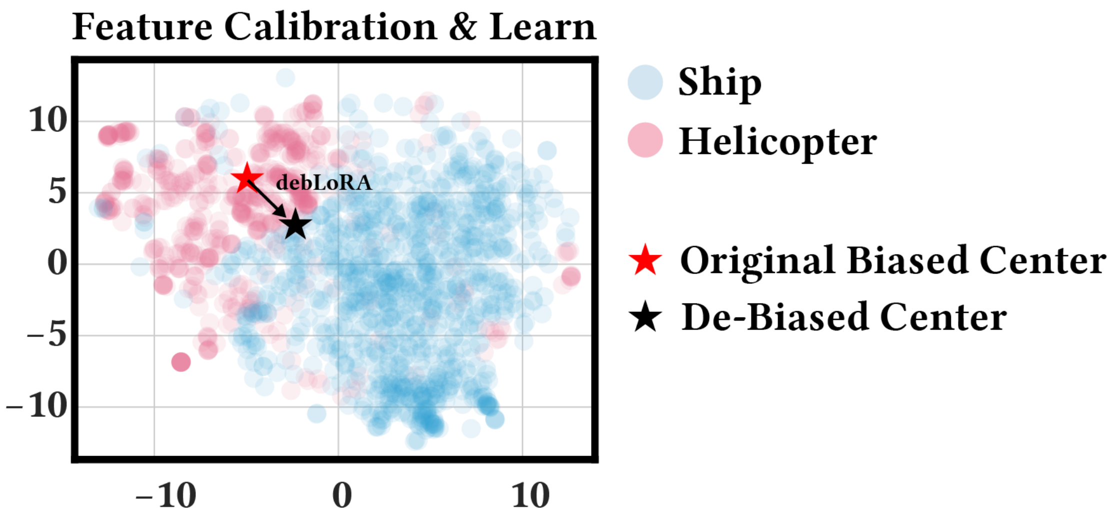
- **Tail Class Calibration**
  - De-Biased Center are **closer to validation** samples.
  - We calibrate tail class features z by moving them close to de-biased center $\hat{\mu}$:

$$\tilde{z} = \alpha z + (1 - \alpha)\hat{\mu},$$

where $\alpha = \min(1, \frac{10}{ir})$ empirically.

- **Learning debLoRA**
  - We learn an LoRA module with training objective

$$\min_{\phi} \frac{1}{D_t} \sum_{x \in D_t} \left\| g_{\phi}(f_{\theta}(x)) - \tilde{z} \right\|^2$$



Experimental Results

Feature Distribution Study • Ablation Studies • Hyperparameter Studies • Validation on Multiple Domains • Validation on Multiple Tasks

Compare Intra- and Inter-class distance.

Feature Distribution Analysis

- **Inter-class distance**
  - debLoRA achieves **higher inter-class distances** for **both head and tail** classes, indicating improved head-tail separability.
- **Intra-Class Distance for Tail**
  - debLoRA maintains **lower and more consistent intra-class distances** for tail classes, suggesting more compact and **generalizable** features for tail.

Table R5: **Quantitative feature analysis on the DOTA dataset.** Inter-class distance is measured as the average cosine distance between class centers, while intra-class distance is the average cosine distance between samples and their corresponding class centers.

Method	**Inter-class**		**Intra-class**
	Head-Tail	Tail-Tail	Tail
Fine-tuning	0.674	0.621	0.170
LoRA	0.702	0.607	0.182
w/ debLoRA	0.719	0.632	0.146

Evaluate the sensitive of our method to hyperparameters: rank and clustering.

Sensitivity to Clustering Hyper-params

- **K-Means Clustering**
  - Our method is **non-sensitive to cluster number K** in K-Means Clustering. Recommended cluster number is between 32 and 64.
- **DBSCAN Clustering**
  - Our method shows comparable performance as mini batch K-Means, and **non-sensitive to eps** hyper-param.

Table R6: **Ablation study on the number of clusters (K) in debLoRA.** Experiments were conducted on the SD → DOTA adaptation. Our default value are marked in gray.

K	Macro F1 Score (%)		
	Head	Middle	Tail
16	99.1	96.9	90.4
32	99.3	97.7	95.1
64	99.3	97.4	94.8

Table: **Compare with DBSCAN (eps).** The DBSCAN results is close to K-Means.

Eps	Macro F1 Score (%)		
	Head	Middle	Tail
32	99.3	97.7	95.1
0.1 (K=43)	99.3	97.3	94.2
0.2 (K=37)	99.3	97.8	95.3

Generalization across Different Adaptation Settings

- **Natural to Natural**
- **Natural to Remote Sensing**
- **Optical Remote Sensing to SAR**

Table 2: **State-of-the-art comparison under different adaptation settings.** The experiments are conducted on two RS adaptation settings: 1) Natural→ORS, where we adopt Stable Diffusion (SD) and OpenCLIP as foundation models and DOTA as the target dataset. 2) ORS→SAR, where we adopt SatMAE as the foundation model and FUSRS (SAR imagery dataset) as the target dataset. Results are evaluated by linear probing and reported in macro F1-Score (%). The highest result in each position is highlighted by **bold**. Our results are marked in gray.

Method	SD → DOTA			OpenCLIP → DOTA			SatMAE → FUSRS		Mean		
	Head	Middle	Tail	Head	Middle	Tail	Head	Tail	Head	Middle	Tail
Zero-Shot	99.2	97.3	87.9	93.1	92.7	91.7	78.3	67.8	90.2	95.0	82.5
Fine-Tune	99.1	96.7	86.8	93.1	91.1	89.2	88.6	73.6	93.6	93.9	83.2
cLoRA	99.1	94.3	89.3	97.3	93.3	92.2	89.9	82.0	95.5	93.8	87.9
w/ debLoRA	99.3	97.5	93.5	**97.6**	**95.8**	**95.0**	**92.5**	**86.1**	**96.5**	**96.7**	**91.5**
LoRA	99.4	97.2	91.8	96.6	92.7	91.6	87.1	76.3	94.4	95.0	86.6
w/ ResLT [8]	**99.4**	97.7	93.0	97.7	94.1	93.8	86.6	75.4	94.6	95.9	87.4
w/ SADE [65]	99.1	97.3	92.4	97.3	93.0	92.5	89.6	78.4	95.3	95.2	87.8
w/ debLoRA	99.3	**97.7**	**95.1**	97.2	95.6	94.8	90.1	81.0	95.5	**96.7**	90.3

How we evaluate our method?

Results on Oriented Object Detection

- **Oriented Object Detection**
  - Our method consistently outperforms state-of-the-art, especially for the tail classes.

Table R1: **Evaluation of oriented object detection on the DOTA dataset.** We report mAP (%) for head, middle, and tail classes. "From-Scratch" refers to training both the feature extractor and FCOS detector head from scratch. All methods use FCOS as the detector head. The "Params (M)" column shows the number of parameters in the feature extractor.

	Method	**mAP (%)**			**Avg. (%)**	**Backbone**
		Head	Middle	Tail		
1	From-Scratch	75.8	83.7	62.1	73.9	ResNet-101
2		76.1	84.2	62.8	74.4	SD U-Net
3	Zero-Shot	71.0	73.7	55.9	66.9	SD U-Net
4	Fine-Tune	76.3	84.9	64.3	75.2	SD U-Net
5	LoRA	77.5	86.3	66.5	76.7	SD U-Net
6	debLoRA (Ours)	**79.4**	**88.5**	**73.2**	**80.4**	SD U-Net

Please access our paper and code using following links.

Thank You!

- **Future Directions**
  - Explore how to align train/val mismatch in PEFT
  - Explore non-linear optimization in PEFT
- **Supplementary Links**
  - Here is our paper’s ArXiv Link
  - Here is our paper’s GitHub Repo

Appendix

Full Dataset Details • Additional Experimental Results

Data availability of ORS large-scale pre-training?

Datasets – Available ORS Pre-training Data

Year	Dataset Name	Classes	Image Size (pixel)	Image (Instance)	Annotation Format	Image Source	Resolution (m)
2014	NWPU VHR-10	-	956 x 554 ~ 1073 x 704	715 / 85	Object classification	Google Earth / Vaihingen	0.5-2 / 0.08
2014	NWPU-RESISC45	-	256 x 256	31,500 (700 each class)	Scene classification	Google Earth	0.2-30
2016	HRSC 2016	-	300 x 300 ~ 1500 x 900	1061	Ship detection and classification	Google Earth	0.4-2
2016	Airbus Ship Detection	-	768 x 768	208,162	Ship detection	-	-
2018	xView	60	2772 x 2678 ~ 5121 x 3023	1413	Object Class	WorldView-3	0.3
2019	HRRSD	-	493 x 402 ~ 2077 x 2606	21,761 / 4961	Object Class	Google Earth / Baidu Map	0.15-1.2 / 0.6-1.2
2020		20	800 x 800	23,463 (192,518)	OBB	Google Earth	0.5-30
2020	FGSD	43	930 x 930	2,612 (5,634)	HBB, OBB	Google Earth	0.12-1.93

Table: **Open-sourced ORS datasets (1/2).** Only datasets released after 2013 are listed.

Data availability of ORS large-scale pre-training?

Datasets – Available ORS Pre-training Data

Year	Dataset Name	Classes	Image Size (pixel)	Image (Instance)	Annotation Format	Image Source	Resolution (m)
2020	FGSC-23	23	40 x 40 ~ 800 x 800	4,080	Class	Google Earth	0.4-2
2021	FGSCR-42	42	50 x 50 ~ 1500 x 1500	9,320	Class	Google Earth	-
2021	ShipRSImageNet	50	930 x 930 ~ 1,400 x 1,400	3,435 (17,573)	HBB, OBB	Multi-sources	-
2022	DOTA v2.0	18	800 x 800 ~ 20K x 20K	11,268 (1,793,658)	OBB	Google Earth / JiLin-1 / GaoFen-2	0.1 ~ 4.5
2022	VHRShips	35	280 x 720	5,312 (11,179)	HBB	Google Earth	0.43
2022	FAIR1M	37	1,000 x 1,000 ~ 10,000 x 10,000	42,796 (1.02M)	OBB	Google Earth / GaoFen	0.3-0.8
2023	UOW-Vessel	10	8192 x 4320 ~ 8192 x 6881	3,500 (35,598)	Polygon	Google Earth	-
	TOTAL			**4.7M**			

Table: **Open-sourced ORS datasets (2/2).** Only datasets released after 2013 are listed.

Data availability of SAR large-scale pre-training?

Datasets – Available SAR Pre-training Data

Year	Dataset Name	Samples	Sensor	Polarization	Resolution	Classes	Annotation Format
2017	SSDD	1,160	Sentinel-1, RadarSat-2, TerraSAR-X	HH,VV,VH,HV	1~15m	1 (Ship)	Object detection
2017	OpenSARShip 2.0	34,528	Sentinel-1	-	-	-	Object detection, scene classification
2018	SEN1-2	282,384	Sentinel-1/2	Single-pol	-	-	Image matching, data fusion
2018	SARptical	10,108	TerraSAR-X	-	<1m	-	Image matching
2019	AIR-SARShip-1.0/2.0	31/300	Gaofen-3	-	1m/3m	10+	Object detection
2019	SEN12MS	180,662	Sentinel-1/2	Dual-pol	10m	-	Image classification, semantic segmentation, data fusion
2019	PolSF	3,000	Various	Full-pol	-	5-6	Image classification, semantic segmentation, data fusion
2019	SAR-Ship	43,819	Gaofen-3/Sentinel-1	-	-	-	Object detection, scene classification
2019	ShipDataset	39,729	Sentinel-1, Gaofen-3	HH,VV,VH,HV	3~25m	1 (Ship)	Object detection
2020	HRSID	5,604	Sentinel-1B, TerraSAR-X, TanDEM-X	HH,HV,VH,VV	0.5~3m	1 (Ship)	Object detection

Table: **Open-sourced SAR datasets (1/2).** Only datasets released after 2016 are listed. Majorly involve object detection, scene classification, segmentation tasks.

Data availability of SAR large-scale pre-training?

Datasets – Available SAR Pre-training Data

Year	Dataset Name	Samples	Sensor	Polarization	Resolution	Classes	Annotation Format
2020	So2Sat LCZ42	400,673	Sentinel-1/2	Dual-pol	-	17	Image classification, data fusion, uncertainty quantification
2020	FUSAR-Ship	5,000+	Gaofen-3	-	-	-	Object detection
2020	OpenSARUrban	33,358	Sentinel-1	Dual-pol	10m	10	Image classification
2020	MSAW	48,000	Capella-X	Quad-pol	0.5m	-	Semantic segmentation
2022	MSAR	30,158*	HISEA-1	HH,HV,VH,VV	≤1m	4	Object detection
2022	SADD	883	TerraSAR-X	HH	0.5~3m	1 (Aircraft)	Object detection
2023	SAR-AIRcraft	18,888*	Gaofen-3	Uni-polar	1m	1 (Aircraft)	Object detection
2023	OGSOD	18,331	Gaofen-3	VV/VH	3m	3	Object detection
2023	SIVED	1,044	Airborne SAR synthetic slice	VV/HH	0.1,0.3m	1 (Car)	Object detection
2023	SARDet-100k	116,598	Multiple	Multiple	0.1~25m	6	Multi-class object detection
	TOTAL	**1.273M**					

Table: **Open-sourced SAR datasets (2/2).** Only datasets released after 2016 are listed. Only 1.27M available SAR data, while there are more then TB-level unlabeled SAR data available[2]. Majorly involve object detection, scene classification, segmentation tasks.

[2] [Datasets | Resources | ICEYE]