

Article

CAP-DBNet: A Class-Adaptive Prototype-Guided Dual-Branch Collaborative Network for Remote Sensing Image Semantic Segmentation

Songtao Shao, Ruilin Li and Ming Yu *

College of Computer Science and Artificial Intelligence, Northeast Forestry University, Harbin 150006, China; 2024112672@nefu.edu.cn (S.S.); 2023122583@nefu.edu.cn (R.L.)

* Correspondence: yuming@nefu.edu.cn

Highlights

What are the main findings?

- Across five repeated experiments with different random seeds on the Vaihingen, Potsdam, and LoveDA datasets, CAP-DBNet improves the average mIoU over the same-backbone LOGCAN++ baseline by 1.43, 1.24, and 1.44 percentage points, respectively.
- Across five-seed evaluations on three benchmark datasets, CAP-DBNet achieves positive average IoU gains for all evaluated foreground categories compared with the same-backbone LOGCAN++ baseline.

What are the implications of the main findings?

- The collaboration of global–local representation learning, class-adaptive supervision, and prototype-guided calibration provides complementary improvements for remote sensing image semantic segmentation.
- Compared with the larger D2LS model, CAP-DBNet achieves a better accuracy–efficiency trade-off with substantially fewer parameters and FLOPs.

Abstract

High-resolution remote sensing image semantic segmentation faces challenges from large-scale variation, fine boundary preservation, and class imbalance. This paper proposes CAP-DBNet, a compact framework that combines a RepViT M2.3 backbone with a global–local collaborative decoder (GLCD), a class-adaptive hybrid loss (CAHL), and a class-adaptive prototype-guided calibration module (CAPC). CAP-DBNet integrates class-center global and window-local modeling through bidirectional feature exchange, mini-batch class-frequency-aware supervision, and prototype-based residual calibration. Rather than introducing an isolated new attention or prototype mechanism, this work focuses on the coordinated design of these components within an efficient segmentation framework. In five repeated experiments with different random seeds, CAP-DBNet achieves average mIoUs of 85.16%, 88.32%, and 55.75% on Vaihingen, Potsdam, and LoveDA, respectively, improving over the same-backbone LOGCAN++ baseline by 1.43, 1.24, and 1.44 percentage points. Compared with D2LS under our evaluation protocol, CAP-DBNet uses 3.55× fewer parameters and requires 8.35× fewer floating-point operations (FLOPs). These results demonstrate an accuracy–efficiency-oriented global–local and class-adaptive framework for remote sensing semantic segmentation across three public benchmarks.

Received: 30 August 2026

Revised: 11 October 2026

Accepted: 15 October 2026

Published: 16 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: semantic segmentation; deep learning; global–local collaboration; class imbalance; prototype learning; class-adaptive learning

1. Introduction

High-resolution remote sensing image semantic segmentation assigns a land-cover category to each pixel. It is a fundamental task for land-use change monitoring, fine-scale urban management, disaster assessment, ecological conservation, desertification and sandy-land boundary monitoring, and territorial spatial planning. As spatial resolution continues to improve, objects such as buildings, roads, water bodies, bare land, forests, agricultural land, and vehicles can be observed at a finer granularity, but category boundaries, local textures, shadow occlusion, and inter-class similarity also become more prominent. This introduction focuses on the modeling challenges; the detailed benchmark descriptions are kept in Section 2.1 to avoid repeating dataset background information across sections.

Existing semantic segmentation methods have broadly evolved from convolutional encoder–decoder networks, atrous convolution, and pyramid pooling to Transformers, state space models, and class-context modeling. A fully convolutional network (FCN) established the basic paradigm of end-to-end pixel prediction [1], and UNet showed the effectiveness of encoder–decoder skip-connection fusion [2]. DeepLabV3+ [3], PSPNet [4], HRNet [5], and SegFormer [6] improved segmentation from the perspectives of multi-scale context, high-resolution representation, and hierarchical attention. For remote sensing scenes, UNetFormer [7], DCSwin [8], PyramidMamba [9], RepViT [10], LOGCAN++ [11], and DOCNet [12] further introduced Transformer structures, efficient backbones, local-global class context, dual-domain optimization, or state space feature fusion, and achieved strong results on public remote sensing datasets. These studies indicate that remote sensing semantic segmentation is no longer a simple pixel classification problem. It is a joint modeling problem that depends on semantic context, local structure, and class discrimination.

Despite the continued improvement in segmentation accuracy, three issues remain challenging when jointly considered within an efficient segmentation framework. First, remote sensing objects exhibit large-scale variation. Large agricultural fields, forests, and building blocks require long-range semantic consistency, whereas roads, vehicles, and small fragments depend on local texture and boundary localization. Convolution-based models may have limited ability to capture long-range dependencies without explicit context modeling [13], whereas global attention mechanisms can increase computational cost and often require additional designs to preserve fine spatial details [14]. Second, remote sensing data are usually class-imbalanced. Minority classes have fewer pixels and more variable shapes, and standard cross-entropy is easily dominated by majority classes. Focal Loss [15], Dice Loss [16], Tversky Loss [17], and Class Balanced Loss [18] can alleviate part of this supervision bias, but fixed weights are difficult to adapt to class-distribution changes across mini-batches and scenes. Third, classes such as tree and low vegetation, bare land and impervious surfaces, and forest and agricultural land can be highly similar in spectrum and texture. Reweighting alone cannot explicitly enlarge their distances in feature space.

CAP-DBNet addresses these challenges by coordinating global–local representation learning, class-frequency-aware supervision, and prototype-guided calibration within a compact RepViT-based framework. The framework is evaluated on three public benchmarks, Vaihingen, Potsdam, and LoveDA, in terms of segmentation accuracy, component contributions, and computational efficiency.

Recent related methods focus on category-guided global–local interaction, dynamic class dictionaries, refined class prototypes, and category-prior-guided frequency enhancement [19–22]. Building on these studies, CAP-DBNet establishes a collaborative learning framework with explicit links between feature interaction and training supervision. Structurally, GLCD uses cross-branch interaction (CBI) to enable bidirectional modulation between global class semantics and local spatial structure. Adaptive branch weighting (ABW) then adjusts their fusion weights according to the current sample and decoder scale, allowing information exchange before fusion. For supervision, CAHL derives smoothed weights from mini-batch class frequencies and applies them to the main segmentation, auxiliary segmentation, and CAPC prototype-discrimination losses. Class-adaptive supervision thus jointly constrains segmentation prediction and prototype discrimination. The methodological contribution lies in collaborative decoding through bidirectional feature interaction and adaptive fusion, together with joint optimization linking class-frequency-aware supervision and prototype-guided feature calibration. These mechanisms establish explicit connections among the components rather than simply placing existing attention, loss, and prototype-learning techniques alongside one another.

The contributions are summarized as follows.

First, we design a global–local collaborative decoder (GLCD). It retains the class-center global modeling mechanism of LOGCAN++ and introduces a window-local branch. Bidirectional cross-branch interaction (CBI) and adaptive branch weighting (ABW) enable collaborative modeling of class-level global semantics and local spatial structure.

Second, we develop a class-adaptive hybrid loss (CAHL), which derives class-adaptive weights from mini-batch pixel frequencies and smooths them using an exponential moving average (EMA). The smoothed weights are applied jointly to cross-entropy (CE) and Dice losses, allowing supervision strength to adapt to the class composition of the current batch.

Third, we design class-adaptive prototype-guided calibration (CAPC), which calibrates pixel features using shared learnable class prototypes and a gated residual mechanism. The smoothed class weights generated by CAHL are also applied to the prototype-discrimination loss. Class-adaptive supervision thus governs both segmentation learning and prototype discrimination, linking supervision optimization with class-based feature calibration.

Beyond the benchmark evaluation, the proposed framework has also been incorporated into a practical remote-sensing analysis workflow for desertification monitoring in Horqin Left Wing Rear Banner, Tongliao City, Inner Mongolia Autonomous Region. In this application, CAP-DBNet is used for pixel-level extraction of desertified areas from Sentinel-2 imagery, providing segmentation outputs for subsequent area statistics and spatiotemporal change analysis. This application context demonstrates the practical relevance of the proposed segmentation framework, while the quantitative experiments reported in this paper remain based on the public benchmark datasets described in Section 2.1.

The following review outlines the related research and methodological differences to further clarify how this design relates to existing work.

1.1. Related Work

1.1.1. Remote Sensing Image Semantic Segmentation and Public Benchmarks

After semantic segmentation was extended from natural images to remote sensing scenes, the input distribution and evaluation focus changed substantially. Natural image segmentation emphasizes semantic separation between objects and backgrounds. Remote sensing imagery, by contrast, often has an overhead viewpoint, large object-scale variation, repeated textures, shadow occlusion, and ambiguous class boundaries. Volpi and

Tuia [23] showed that dense labeling of sub-decimeter aerial imagery requires pixel-level models that preserve geometric detail. LoveDA focuses on land-cover differences between urban and rural scenes [24], whereas Potsdam and Vaihingen emphasize buildings, roads, vegetation, vehicles, and background classes in high-resolution aerial imagery [25]. Together, these benchmarks enable evaluation under urban, rural, and high-resolution aerial conditions.

From the perspective of model development, FCN transformed classification networks into fully convolutional prediction structures [1], and UNet restored spatial details through an encoder–decoder architecture and skip connections [2]. Maggiori et al. [26] further analyzed high-resolution aerial semantic labeling and emphasized the need to combine local and global information efficiently. DeepLabV3+ introduced atrous convolution and an encoder–decoder design [3], PSPNet aggregated multi-scale context through pyramid pooling [4], HRNet maintained high-resolution feature branches [5], and SegFormer improved semantic modeling with a hierarchical Transformer encoder [6]. These methods provide important baselines for remote sensing segmentation. However, under small-target, fine-boundary, and class-imbalanced conditions, they may still produce local discontinuities, fragmented regions, and missed minority classes.

1.1.2. Multi-Scale Context and Global–Local Collaborative Modeling

The same class in remote sensing imagery may appear in very different forms at different scales. Building blocks and agricultural fields often form large continuous regions, whereas roads and vehicles are elongated or small-scale structures. Atrous convolution [3] and pyramid pooling [4] enlarge the receptive field, while feature pyramid networks fuse features at different scales through top-down pathways [27]. For fine-resolution remotely sensed images, A2-FPN [28] shows that attention-enhanced feature pyramids can strengthen multi-scale feature aggregation. Dual-branch structures such as BiSeNet [29] and DDRNet [30] further show the effectiveness of separately modeling spatial details and semantic context. However, many multi-scale fusion strategies rely on predefined aggregation operations and may have limited capability to adaptively model different regional characteristics. Channel-reweighting methods such as squeeze-and-excitation networks [31] further show that pooled descriptors can modulate feature importance; adaptive branch weighting (ABW) instead uses a pooled descriptor to estimate two branch-level weights.

Transformers [13] and window attention [14] provide complementary mechanisms for long-range and local modeling in remote sensing segmentation. UNetFormer [7], DCSwin [8], PyramidMamba [9], and FGNet [32] strengthen multi-scale or global–local representations through Transformer, state-space, or frequency-domain designs. CGGLNet [20] directly studies category-guided global–local feature interaction. D2LS [19] constructs an input-adaptive class dictionary and alternates cross-attention between dictionary embeddings and image features. CRECA-Net [21] refines class prototypes and uses a pixel-difficulty-aware objective, whereas GC2F-Net [22] combines an image-adaptive global category-center prior with frequency-domain enhancement and category-aware local decoding. Recent Hypergraph BiFormer [33] and MSTNet [34] further investigate high-order relational modeling and multi-scale Transformer representations for remote sensing segmentation. Against this background, CAP-DBNet combines lightweight bidirectional global–local representation learning with mini-batch class-frequency-aware supervision and static prototype residual calibration.

1.1.3. Class Imbalance, Loss Reweighting, and Prototype Discrimination

Class imbalance is a common problem in remote sensing semantic segmentation. Large-area classes usually dominate gradient updates, whereas vehicles, bare-land edges,

narrow roads, and other minority or fragmented regions are easily ignored. Focal Loss [15] emphasizes hard samples, Dice Loss [16] and Tversky Loss [17] focus on regional overlap, and Class Balanced Loss [18] adjusts class weights according to the effective number of samples. These methods can reduce supervision bias, but most of them rely on global statistics or fixed weights. They therefore have difficulty reflecting the effective class distribution that appears in a mini-batch. The proposed class-adaptive hybrid loss associates class weights with the effective pixel frequency in the current mini-batch, and uses an exponential moving average (EMA) to reduce weight fluctuation. This design provides a more flexible adjustment of supervision strength during training.

In addition to supervision imbalance, feature-space overlap between confusing classes is another important factor that affects mIoU. Prototype learning represents semantic concepts with class centers [35], and supervised contrastive learning improves discrimination by pulling same-class features together and pushing different-class features apart [36]. ProtoSeg [37] and related prototype-based segmentation methods [38] show that pixel-level prediction can also benefit from class-prototype constraints. Object-contextual representations (OCR) further indicate that class-level semantic summaries help pixels obtain more stable contextual representations [39]. Based on these observations, CAP-DBNet introduces learnable class prototypes at the decoder end. Pixel features are not only used for classification output, but also calibrated through pixel-prototype similarity. This mechanism supports both feature enhancement and discriminative supervision.

Table 1 compares CAP-DBNet with the closest category-guided, global–local, dictionary-based, and prototype-based methods. The entries summarize published mechanisms and design scope; they do not constitute a same-protocol performance ranking.

Table 1. Mechanistic comparison with closely related methods.

Method	Year	Global–Local/Class-Aware Modeling	Class Representation	Adaptive Training Objective	Efficiency/Distinctive Scope
LOG-CAN++ [11]	2025	Local-global class-aware context	Class-level context aggregation	No explicit adaptive reweighting	Context modeling baseline
CGGLNet [20]	2024	Category-guided global–local interaction	Category-guided context	No mini-batch class weighting	Transformer-based interaction
D2LS [19]	2025	Iterative image-dictionary interaction	Input-adaptive class-ID dictionary	Static/dynamic + contrastive losses	ConvNeXt-B in the original work
CRECA-Net [21]	2026	Prototype-guided class-level context	Refined class prototypes	Pixel-difficulty-aware loss	ResNet-50; prototype-quality focus
GC2F-Net [22]	2026	Category-prior-guided spatial-frequency collaboration	Image-adaptive category-center prior	No class-frequency reweighting	ResNet-50; frequency enhancement
CAP-DBNet	This work	Bidirectional class-center global/window-local branches	Static prototypes for residual calibration	Mini-batch class-frequency-aware cross-entropy + Dice	RepViT-based accuracy-efficiency design

D2LS [19] enhances class representations through an input-adaptive class dictionary and iterative image-dictionary interaction. CAP-DBNet instead builds on existing class-center global modeling by introducing a window-local branch. Bidirectional cross-branch interaction (CBI) and adaptive branch weighting (ABW) collaboratively fuse global semantics with local spatial structure. CRECA-Net [21] focuses on class-prototype refinement and pixel-difficulty-aware optimization, whereas CAPC calibrates features using shared learnable class prototypes and a gated residual mechanism. GC2F-Net [22] combines image-adaptive class-center priors with frequency-domain enhancement. In

contrast, CAP-DBNet emphasizes bidirectional collaboration between its class-center global branch and window-local branch.

CAP-DBNet also links class-adaptive supervision with prototype-guided feature calibration through CAHL and CAPC, extending its design beyond feature representation and fusion. CAHL generates class-adaptive supervision weights from mini-batch class frequencies using EMA smoothing. These weights are shared by the main segmentation loss, auxiliary segmentation loss, and CAPC prototype-discrimination loss, establishing consistent class-adaptive weighting for segmentation prediction and prototype discrimination. The unified framework uses GLCD bidirectional collaborative decoding for feature fusion, CAHL class-adaptive supervision for optimization, and CAPC prototype-guided calibration to enhance class representations. Relative to the related methods discussed above, its distinguishing feature is the explicit functional division and training-objective links among the three components. Jointly designed global–local feature interaction, class-frequency-aware supervision, and prototype calibration improve semantic segmentation in complex remote sensing scenes.

2. Materials and Methods

This section describes the datasets and preprocessing, the CAP-DBNet methodology, the optimization objective, and the experimental settings and evaluation metrics. The implementation projects RepViT M2.3 features to 96 channels, uses a local-attention window size of 8, applies smoothed training-time class weights in CAHL, and uses static learnable prototypes with residual prototype-context injection in CAPC.

2.1. Datasets and Preprocessing

LoveDA, Potsdam, and Vaihingen are used as public remote sensing semantic segmentation benchmarks to evaluate the effectiveness of CAP-DBNet under different spatial resolutions, scene complexities, and class systems. Together, the three datasets cover urban, rural, and high-resolution aerial scenes, and enable evaluation under scale variation, class confusion, and class imbalance.

Table 2 reports normalized foreground-class proportions to describe relative class imbalance across the three datasets. These proportions are computed directly from the training-label masks of each dataset after applying the same preprocessing and foreground-class inclusion/exclusion rules used in the experiments. They are normalized over the evaluated foreground classes after excluding non-evaluated labels; split-level absolute pixel counts are not used in the analysis. The exact evaluated classes and ignored labels are specified in Section 2.7 and in Table 3.

Table 2. Foreground class proportions of the three datasets.

Dataset	Class	Proportion (%)
Vaihingen	ImSurf	28.75
	Building	26.39
	LowVeg	20.59
	Tree	22.85
	Car	1.42
	Imbalance ratio (max/min)	20.2×
Potsdam	ImSurf	32.90
	Building	25.91
	LowVeg	21.68
	Tree	17.85
	Car	1.66

LoveDA	Imbalance ratio (max/min)		19.8×
	Agricultural		44.73
	Water		18.33
	Forest		11.53
	Building		11.28
	Road		7.31
	Barren		6.83
	Imbalance ratio (max/min)		6.5×

Table 3. Training settings and evaluation protocol.

Dataset	Split and Metric	Crop	Epochs	Baseline/CAP Train/Val Batch	Training/Evaluation Setting
LoveDA	Official train/val; six-class foreground mIoU	512 × 512	100	16/16	Baseline/CAP: AdamW lr = 1×10^{-4} , wd = 1×10^{-4} ; poly = 0.9; validation-best; five random seeds for the main comparisons and component ablations. D2LS: method-specific optimization (see text).
Potsdam	ISPRS original-tile split; foreground-5 mIoU	512 × 512	100	16/16	Clutter/Background participates in training and confusion-matrix statistics but is excluded from the foreground-five-class mean; Boundary pixels are ignored.
Vaihingen	ISPRS original-tile split; foreground-5 mIoU	512 × 512	100	16/16	Clutter/Background participates in training and confusion-matrix statistics but is excluded from the foreground-five-class mean; Boundary pixels are ignored.

As shown in Table 2, the Car class accounts for only approximately 1.4% and 1.7% of the evaluated foreground pixels in Vaihingen and Potsdam, respectively, whereas the dominant ImSurf class accounts for approximately 29% and 33%. The corresponding inter-class imbalance ratios are 20.2× and 19.8×. In LoveDA, Agricultural accounts for 44.7% of the evaluated foreground pixels, whereas Barren accounts for 6.8%, with an imbalance ratio of 6.5×. These normalized proportions provide a descriptive view of class imbalance. During training, equal class weighting can allow gradients from minority classes such as Car and Barren, as well as small or fragmented objects, to be dominated by large-area classes. CAHL therefore estimates class weights from each training mini-batch rather than relying on the fixed proportions in Table 2.

LoveDA is a large-scale dataset for remote sensing semantic segmentation [24]. Its images are collected from Google Earth in Nanjing, Changzhou, and Wuhan, China. The spatial resolution is 0.3 m, and each image has a size of 1024 × 1024 pixels. The dataset contains urban and rural scenes with seven annotated land-cover classes plus ignore/unlabeled pixels. Background remains one of the seven prediction classes during training and inference; only the reported metrics exclude it from class averaging. This paper evaluates on the official validation set and reports six-class foreground mIoU, consistent with the metric definition used in the comparison tables.

Potsdam is a high-resolution aerial image dataset from the International Society for Photogrammetry and Remote Sensing (ISPRS) 2D Semantic Labeling Contest [25]. It covers the urban area of Potsdam, Germany, with a ground sampling distance (GSD) of 5 cm. The original tile size is 6000 × 6000 pixels. The original tiles are first split by tile ID into training and validation sets, and image patches are then extracted separately from each set. The training set contains 24 original tiles, yielding 3456 patches of 1024 × 1024 pixels; the validation set contains 14 original tiles, yielding 504 patches of 1024 × 1024 pixels.

During training, the patches are randomly rescaled and cropped to 512×512 pixels, with online augmentation including flipping, rotation, and Gaussian blur. Validation uses the complete 1024×1024 patches without random augmentation. The original tile IDs do not overlap between the training and validation sets. The semantic classes include Impervious Surfaces (ImSurf), Building, Low Vegetation (LowVeg), Tree, Car, and Clutter/Background. The main metric is computed over the first five classes, and Clutter/Background is excluded from mIoU. If a Boundary label exists in the preprocessed annotations, it is excluded from evaluation using an ignore index of 6.

Vaihingen is also from the ISPRS 2D Semantic Labeling Contest [25]. It covers the urban area of Vaihingen, Germany, with a GSD of 9 cm. The original tiles are approximately 2500×2000 pixels, with variations across tiles. The split is made at the original-tile level, with 16 training tiles and 17 validation tiles, yielding 1072 and 113 patches of 1024×1024 pixels, respectively. During training, the patches are randomly rescaled, then smart cropping constrained by class proportions produces 512×512 inputs to reduce the occurrence of samples dominated by a single class. Validation uses the complete 1024×1024 patches without random augmentation. The original tile IDs have no intersection between the two sets, preventing patches from the same original tile from appearing in both training and validation. The class definitions are the same as those in Potsdam, including ImSurf, Building, LowVeg, Tree, Car, and Clutter/Background. The main evaluation metric is foreground-five-class mIoU. Clutter/Background participates in model training and confusion-matrix statistics but is excluded from class-IoU averaging. Boundary pixels are treated as ignored labels and excluded from loss computation and evaluation statistics.

Potsdam and Vaihingen use tile IDs consistent with the original training/test splits of the ISPRS 2D Semantic Labeling Contest. Potsdam contains 24 training tiles and 14 held-out evaluation tiles, while Vaihingen contains 16 training tiles and 17 held-out evaluation tiles. The original challenge test tiles are used for local validation and model evaluation, with no original tile shared between training and held-out evaluation. The complete tile IDs and their training/validation assignments are provided in Supplementary File S1. All local experiments use dataset-specific training and validation/evaluation protocols. For LoveDA, $K = 7$ and the network output space contains seven semantic classes including Background, whereas the reported metrics average the six foreground classes (Building, Road, Water, Barren, Forest, and Agricultural); ignore/unlabeled pixels are excluded throughout. For Potsdam and Vaihingen, $K = 6$ and the main metric is foreground-5 mIoU over ImSurf, Building, LowVeg, Tree, and Car. Clutter/Background participates as a valid sixth class in model training and confusion-matrix statistics but is excluded from foreground-five-class mIoU averaging. Boundary pixels use `ignore_index = 6` and are excluded from both loss computation and evaluation statistics.

2.2. Overall Architecture

The overall architecture, module relationships, and training supervision paths of CAP-DBNet are shown in Figure 1. GLCD performs global-local representation learning, CAHL provides class-frequency-aware supervision, and static CAPC supplies auxiliary prototype-guided calibration. Their effects are evaluated through the staged component ablation and hyperparameter sensitivity analyses in Sections 3.3 and 3.5.

CAP-DBNet addresses scale variation, class imbalance, and class confusion in remote sensing image semantic segmentation through a collaborative learning framework that combines feature fusion, supervision optimization, and class-representation calibration. Building on the existing class-center global modeling mechanism, GLCD introduces a window-local branch and dynamically fuses global semantic information with local spatial structure through bidirectional cross-branch interaction and adaptive branch weighting. CAHL adaptively adjusts supervision weights using mini-batch class

frequencies and EMA smoothing, allowing model optimization to respond to the class composition of different training batches. CAPC establishes a link between pixel features and class semantics through shared learnable class prototypes and a gated residual mechanism, providing class-guided feature calibration. Furthermore, the smoothed class weights computed by CAHL are applied to the main segmentation loss, auxiliary segmentation loss, and prototype discrimination loss of CAPC, extending class-adaptive supervision to both segmentation prediction and prototype discrimination. With distinct functional roles and explicit links between their training objectives, the three modules form an integrated framework for jointly optimizing global–local representation learning, class-adaptive supervision, and prototype-guided calibration.

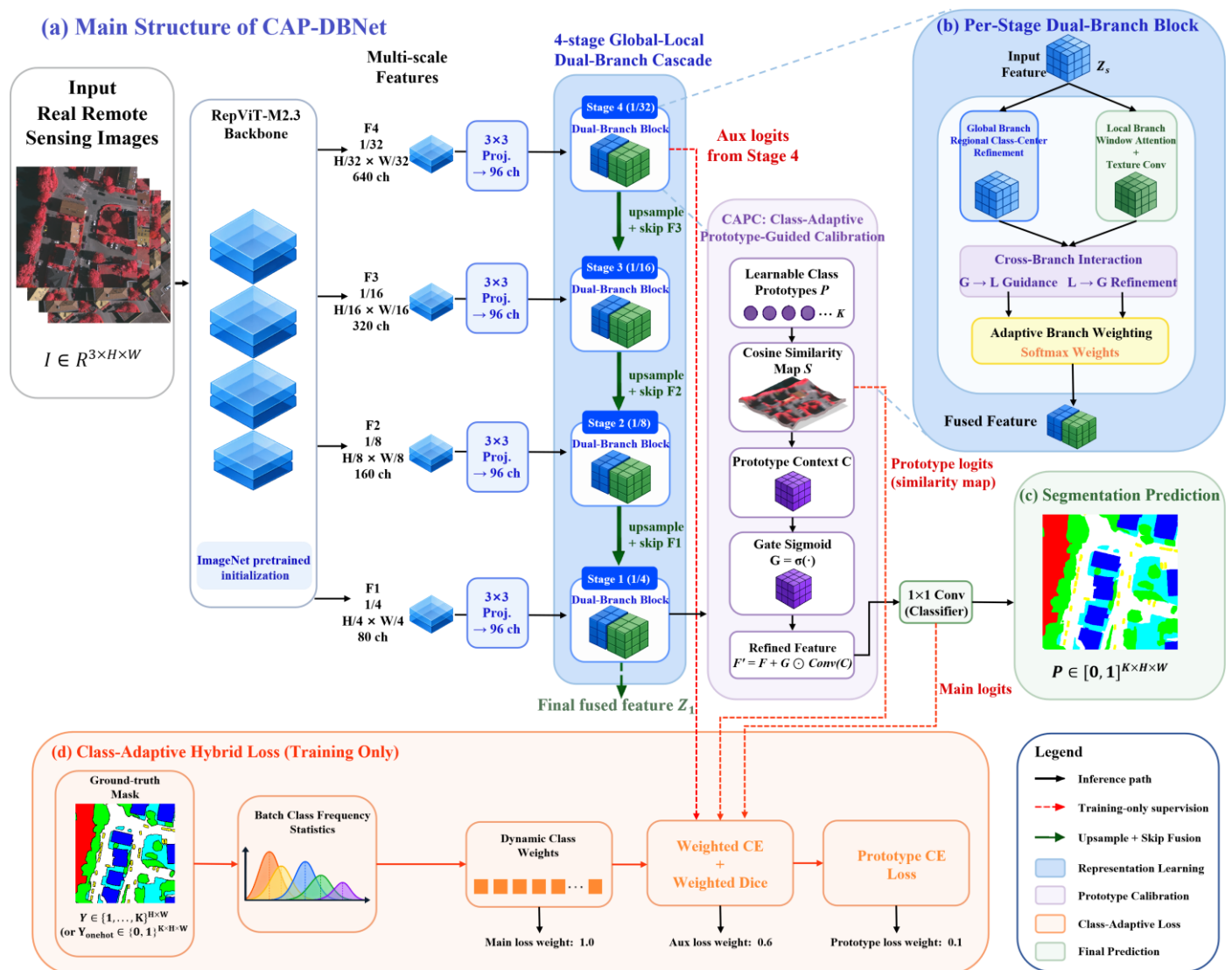


Figure 1. Overall network architecture, module relationships, and training supervision paths of CAP-DBNet. Panels (a–d) show the main architecture, the per-stage dual-branch block, the segmentation prediction, and the training-only class-adaptive hybrid loss, respectively. Black solid arrows denote inference paths, red dashed arrows denote training-only supervision, and green arrows denote upsampling and skip fusion. Blue, purple, orange, and green module backgrounds denote representation learning, prototype calibration, class-adaptive loss, and final prediction, respectively. Cuboids denote feature tensors and circles denote learnable class prototypes.

As shown in Figure 1, the input remote sensing image is fed into an ImageNet-pre-trained RepViT M2.3 backbone [40]. The implementation uses features from backbone

indices 7, 15, 51, and 54, with channel dimensions of 80, 160, 320, and 640, a decoder transform dimension of 96, eight attention heads, the region-partition setting $p = (4,4)$ for the class-center global branch, and a local-attention window size of 8.

Given an input remote sensing image, the backbone extracts four multi-scale feature maps $\{F_1, F_2, F_3, F_4\}$. Their spatial resolutions are approximately $1/4$, $1/8$, $1/16$, and $1/32$ of the input resolution, respectively. The ImageNet-pretrained RepViT M2.3 backbone is used to balance representation capacity and efficiency.

After channel projection, the four backbone features are decoded progressively from Stage 4 to Stage 1. At each scale, a GLCD block forms class-center global and window-local features, exchanges information between the two branches, and adaptively fuses them before upsampling and skip fusion. The Stage-1 fused feature is denoted by Z_1 and is passed directly to CAPC for prototype-guided residual calibration. Figure 1 shows both the inference path and the training-only supervision paths. The main inference chain is $I \rightarrow \text{RepViT} \rightarrow \{F_1, F_2, F_3, F_4\} \rightarrow \text{four-stage GLCD} \rightarrow Z_1 \rightarrow \text{CAPC} \rightarrow P_{\text{main}}$, where $P_{\text{main}} \in \mathbb{R}^{B \times K \times H \times W}$ denotes the main segmentation logits after upsampling. During training, the Stage-4 auxiliary head additionally produces $P_{\text{aux}} \in \mathbb{R}^{B \times K \times H \times W}$, and CAPC produces the pixel-prototype similarity logits $S \in \mathbb{R}^{B \times K \times H' \times W'}$. The ground-truth mask Y provides the class-frequency statistics used to estimate $w_k^{(t)}$, which weights the main, auxiliary, and prototype supervision terms. The red dashed branches are used only for training and are removed during inference.

2.3. Global–Local Collaborative Decoder

Remote sensing images contain both large homogeneous regions and fine structural targets. At decoder scale s , the backbone feature F_s is projected to the unified channel dimension $C = 96$, yielding $Z_s = \phi_s(F_s)$, where $\phi_s(\cdot)$ consists of convolution, normalization, and nonlinear activation. The projected tensor has shape $Z_s \in \mathbb{R}^{B \times C \times H_s \times W_s}$, where B , C , H_s , and W_s denote the mini-batch size, channel dimension, feature height, and feature width, respectively.

The class-center global branch of GLCD inherits the class-aware modeling mechanism of LOGCAN++ [11]. It uses SpatialGatherModule and RVSA_MRAM for class-center aggregation and regionally adaptive attention, respectively. The latter involves learnable sampling offsets, scales, and rotation parameters. These existing operations are not claimed as independent innovations in this work. Instead, the decoder retains the global class-modeling mechanism and adds a window-attention local branch, bidirectional cross-branch interaction (CBI), and adaptive branch weighting (ABW) to jointly model global semantics and local spatial structure.

2.3.1. Class-Center Global Branch

Following the class-aware modeling mechanism of LOGCAN++ [11], the class-center global branch uses SpatialGatherModule and RVSA_MRAM for global class-center aggregation and regional class-center attention, respectively. At decoder scale s , a lightweight 1×1 classification mapping generates preliminary class responses from the projected features. For the projected features $Z_s \in \mathbb{R}^{B \times C \times H_s \times W_s}$, spatial normalization and weighted aggregation of the flattened features yield input-dependent global class centers:

$$A_s = \text{Softmax}_{N_s}(\text{reshape}(W_a Z_s)), \quad C_s = A_s Z_s \quad (1)$$

Here, $N_s = H_s W_s$, $A_s \in \mathbb{R}^{B \times K \times N_s}$, and the flattened features satisfy $Z_s \in \mathbb{R}^{B \times N_s \times C}$. W_a denotes the class-response mapping, and $C_s \in \mathbb{R}^{B \times K \times C}$ contains the K global class centers aggregated from the current input. The global-branch output F_s^g is given by Equation (1d).

Equation (1) aggregates global class centers over the full feature map. Regional class-center refinement follows the RVSA_MRAM of LOGCAN++ [11], further constructing regional class centers at each decoder scale and injecting context through attention. The features Z_s are divided into $p_h \times p_v$ non-overlapping regions ($p_h = p_v = 4$ in this study), each of size $h_s \times w_s$, where $h_s = H_s/p_h$ and $w_s = W_s/p_v$. The features in region r are denoted by $Z_s^{(r)}$. For each region and head m among the M attention heads, the offset $\hat{o}_{r,m} \in \mathbb{R}^2$, scale $\hat{s}_{r,m} \in \mathbb{R}^2$, and rotation angle $\theta_{r,m}$ are predicted to generate the sampling coordinates:

$$\eta_r = W_g \text{LReLU}(\text{GAP}(Z_s^{(r)})) \quad (1a)$$

$$g_{r,m}(\delta) = c_r + R(\theta_{r,m}) \text{diag}(1 + \hat{s}_{r,m}) \delta + \text{diag}(h_s^{-1}, w_s^{-1}) \hat{o}_{r,m}$$

Here, η_r concatenates the offset, scale, and rotation parameters predicted for all attention heads in the region. W_g denotes three parallel 1×1 convolutions, with the offset and scale branches initialized to zero. $R(\theta)$ is a two-dimensional rotation matrix, and c_r is the region center in the $[-1, 1]$ normalized coordinate system. $\delta \in \Lambda_s$ denotes the normalized offset of each of the $h_s \times w_s$ pixels relative to the region center, namely $\delta = \left(\frac{2}{w_s-1} \left(t - \frac{w_s-1}{2} \right), \frac{2}{h_s-1} \left(a - \frac{h_s-1}{2} \right) \right)$, with $a = 0, \dots, h_s - 1$ and $t = 0, \dots, w_s - 1$. When $\hat{o} = \hat{s} = 0$ and $\theta = 0$, the sampling points coincide with the original pixel grid of the region.

Let $\mathcal{S}(\cdot, g)$ denote bilinear interpolation at the normalized coordinates g , with zero padding outside the valid range. At the sampled positions, class responses are used to weight and sum the key features, yielding the regional class centers for region r and head m :

$$C_{r,m}^{\text{loc}} = \sum_{\delta \in \Lambda_s} \mathcal{S}(P_s, g_{r,m}(\delta)) \mathcal{S}(Z_s^{k,(m)}, g_{r,m}(\delta))^T \in \mathbb{R}^{K \times d} \quad (1b)$$

Here, $P_s = \text{Softmax}_{N_s}(W_p Z_s) \in \mathbb{R}^{B \times K \times H_s \times W_s}$ represents the spatial class responses internal to the module, and $Z_s^k = W_k Z_s$ is the key projection. The superscript (m) indicates the channel slice of dimension $d = C/M$ associated with head m . Regional pixels then serve as queries, regional class centers as keys, and global class centers as values:

$$O_r^{(m)} = \text{Softmax}_K \left(\frac{(Z_s^{(r)} W_q^T)^{(m)} (C_{r,m}^{\text{loc}})^T}{\sqrt{d}} \right) (C_s W_v^T)^{(m)} \in \mathbb{R}^{h_s w_s \times d} \quad (1c)$$

In this expression, $Z_s^{(r)}$ is flattened in row-major order into a $h_s w_s \times C$ matrix. The outputs of all heads are concatenated and linearly projected, restored to their original spatial arrangement, and fused with the input features by concatenation to obtain the global-branch output:

$$F_s^g = \phi_{\text{cat}}([Z_s; \text{Merge}_r([O_r^{(1)}, \dots, O_r^{(M)}] W_{\text{out}}^T)]) \in \mathbb{R}^{B \times C \times H_s \times W_s} \quad (1d)$$

Here, Merge_r restores each regional result to its original spatial position, while ϕ_{cat} consists of two blocks of 3×3 convolution, BN, ReLU, and Dropout. The linear mappings include biases, which are omitted from the equations. This study uses $M = 8$ and $C = 96$. Equations (1)–(1d) describe existing computations of LOGCAN++ and are not claimed as innovations in this work. At each decoder scale, C_s is computed independently from the features at that scale, and F_s^g and the window-local branch output F_s^l are passed to CBI and ABW in Section 2.3.3.

2.3.2. Window Local Branch

In parallel, the window-local branch applies multi-head self-attention with a window size of 8 and depthwise separable convolution to Z_s , producing $F_s^l \in \mathbb{R}^{B \times C \times H_s \times W_s}$ for local texture and boundary modeling. Unlike p , the window size specifies the pixel extent of each local-attention window. The two branch outputs are then processed by the cross-branch interaction (CBI) and ABW operations shown in Figure 2.

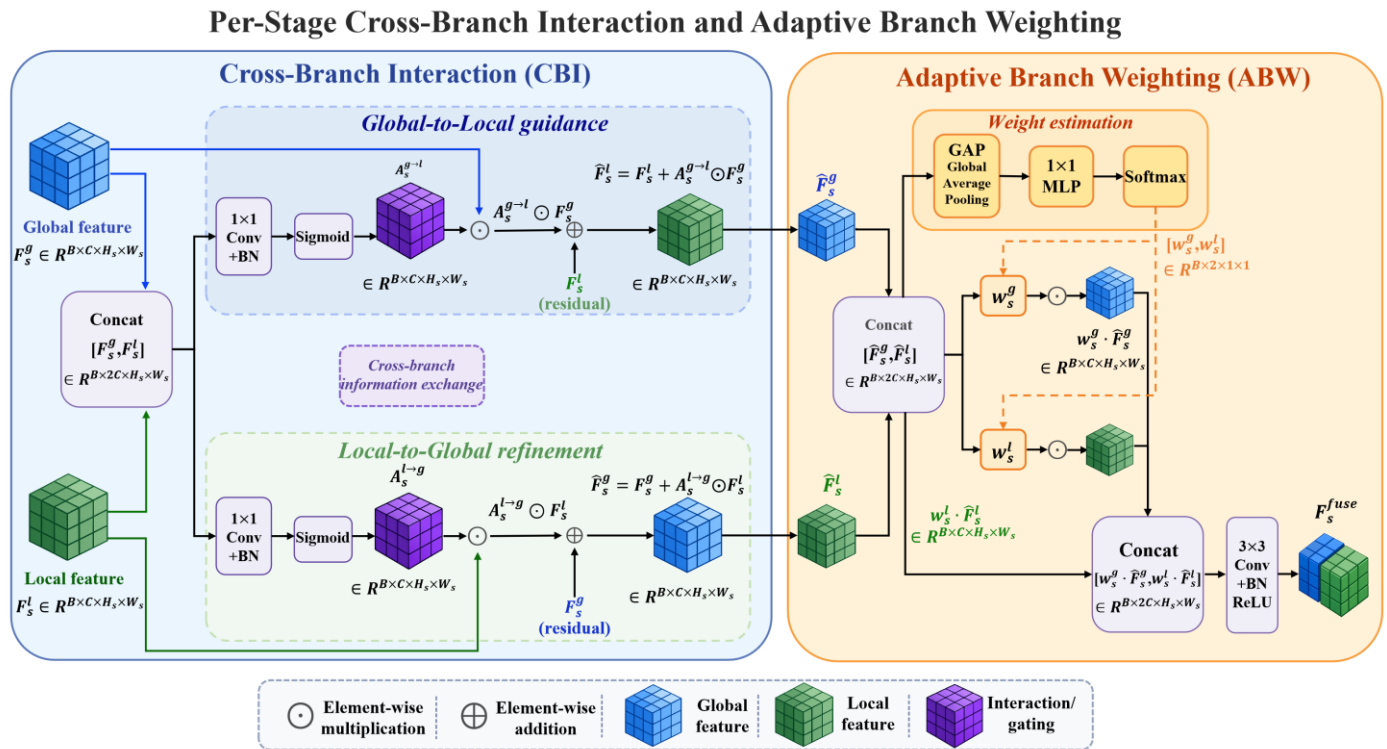


Figure 2. Computation flow of cross-branb interaction and adaptive branb weighting in the global–local collaborative decoder. Blue and green denote global and local features, respectively, and purple denotes cross-branb interaction or gating. Solid arrows indicate feature flow, while orange dashed arrows propagate the adaptive weights. The circled dot and circled plus denote element-wise multiplication and addition, respectively.

2.3.3. Cross-Branh Interaction and Adaptive Branch Weighting

Given F_s^g and F_s^l at scale s , CBI concatenates them as $U_s = [F_s^g; F_s^l] \in \mathbb{R}^{B \times 2C \times H_s \times W_s}$ and learns two directional guidance maps:

$$A_s^{g \rightarrow l} = \sigma(\psi^{g \rightarrow l}(U_s)), \quad A_s^{l \rightarrow g} = \sigma(\psi^{l \rightarrow g}(U_s)) \quad (2)$$

Here, $\psi^{g \rightarrow l}$ and $\psi^{l \rightarrow g}$ are lightweight 1×1 convolution and normalization mappings, $\sigma(\cdot)$ is the Sigmoid function, and $A_s^{g \rightarrow l}$ and $A_s^{l \rightarrow g} \in \mathbb{R}^{B \times C \times H_s \times W_s}$ are channel-spatial guidance maps. They provide global-to-local semantic guidance and local-to-global structural refinement, respectively. The interacted features are computed by residual modulation:

$$\hat{F}_s^l = F_s^l + A_s^{g \rightarrow l} \odot F_s^g, \quad \hat{F}_s^g = F_s^g + A_s^{l \rightarrow g} \odot F_s^l \quad (3)$$

The residual terms preserve the original branb information while injecting complementary information from the other branb. Both $\hat{F}_s^g$ and $\hat{F}_s^l$ retain shape $\mathbb{R}^{B \times C \times H_s \times W_s}$. ABW then concatenates them, applies global average pooling (GAP), and feeds the descriptor to a lightweight multilayer perceptron (MLP).

A Softmax operation produces sample- and scale-dependent global and local branch weights:

$$v_s = \text{GAP}([\hat{F}_s^g; \hat{F}_s^l]), [w_s^g, w_s^l] = \text{Softmax}(\text{MLP}(v_s)), w_s^g + w_s^l = 1 \quad (4)$$

The two scalar branch weights have shape $\mathbb{R}^{B \times 1 \times 1 \times 1}$ and are broadcast over channels and spatial positions. The final fused output at scale s is:

$$F_s^{\text{fuse}} = \phi_f([w_s^g \cdot \hat{F}_s^g; w_s^l \cdot \hat{F}_s^l]) \in \mathbb{R}^{B \times C \times H_s \times W_s} \quad (5)$$

Here, $[\cdot; \cdot]$ denotes channel concatenation and $\phi_f(\cdot)$ is a 3×3 convolution followed by batch normalization (BN) and a rectified linear unit (ReLU). Thus, CBI performs directional information exchange before ABW controls the relative global/local contribution for the current sample and decoder scale.

Figure 2 visualizes Equations (2)–(5). Starting from F_s^g and F_s^l , CBI generates $A_s^{g \rightarrow l}$ and $A_s^{l \rightarrow g}$, produces the interacted features $\hat{F}_s^l$ and $\hat{F}_s^g$ through residual modulation, and passes them to ABW. GAP and the MLP estimate w_s^g and w_s^l ; the weighted features are concatenated and convolved to produce F_s^{fuse} . This operation adjusts branch contributions at the sample and scale levels without explicit pixel-level region selection.

The collaborative design of GLCD lies in the consecutive computation of bidirectional information exchange and adaptive fusion. Unlike directly concatenating or adding independently computed global and local features, CBI first establishes guidance in both the global-to-local and local-to-global directions. Residual modulation allows each branch to receive information from the other branch before fusion. ABW then estimates sample- and scale-dependent branch weights from the interacted joint features, adaptively adjusting the relative contributions of global semantic information and local spatial structure. Thus, GLCD combines two complementary modeling paths through explicit bidirectional information flow and a dynamic fusion rule, forming a unified collaborative decoding process.

2.4. Class-Adaptive Hybrid Loss

Loss computation uses a unified valid-pixel masking mechanism across datasets. For Vaihingen and Potsdam, the model predicts six semantic classes. Clutter/Background, labeled 5, participates in training as the valid sixth class, whereas Boundary is mapped to ignore_index = 6 and excluded from loss computation. For LoveDA, the model predicts seven semantic classes including Background, which is labeled 0 and participates in training; unlabeled pixels are mapped to ignore_index = 7 and excluded from supervision. CAHL estimates class frequencies from valid pixels only. Cross-entropy excludes invalid pixels through ignore_index, Dice Loss masks both predicted probabilities and labels, and prototype discrimination uses the same ignore rules. The main segmentation head, auxiliary segmentation head, and prototype supervision share the same valid-pixel definition and class-adaptive weights, preventing ignored regions from directly affecting loss computation.

Class imbalance biases training toward majority classes and weakens gradients from minority or fragmented regions such as vehicles, narrow roads, and vegetation boundaries. For each mini-batch, n_k denotes the number of valid pixels assigned to class k ; ignored pixels are excluded. The mini-batch class frequency is:

$$f_k = \frac{n_k + \varepsilon}{\sum_{j=1}^K n_j + K\varepsilon} \quad (6)$$

Here, K denotes the number of semantic classes in the model prediction and supervision space, n_j is the valid-pixel count of class j in the current mini-batch, and ε prevents division by zero. The corresponding raw inverse-frequency weight is:

$$a_k = \left(\frac{1}{f_k + \varepsilon} \right)^\gamma \quad (7)$$

In Equation (7), a_k is the raw weight, ε is a smoothing term, and γ controls the strength of frequency modulation; $\varepsilon = 10^{-6}$ and $\gamma = 0.4$ in the reported configuration. If class k is absent from the current mini-batch, its current batch weight is set to zero before clipping and normalization. The remaining positive weights are upper-clipped at $w_{\max} = 3.0$ and divided by their mean, yielding $\bar{w}_k^{(t)}$; absent classes are therefore excluded from the current batch normalization mean.

An exponential moving average (EMA) is then used only during training to reduce mini-batch fluctuation:

$$w_k^{(t)} = \mu w_k^{(t-1)} + (1 - \mu) \bar{w}_k^{(t)} \quad (8)$$

where μ is the EMA momentum and $\mu = 0.85$ in all reported experiments. The batch weight $\bar{w}_k^{(t)}$ is zero for an absent class, so its historical EMA weight decays by μ rather than being reset abruptly. Such a class has no target pixel in cross-entropy, while its decayed weight may still penalize false-positive predictions through the Dice term. This EMA operates only on training loss weights and is unrelated to prototype memory or dynamic prototype updating.

Using the smoothed weights, CAHL combines weighted cross-entropy and weighted Dice losses:

$$L_{CE} = - \left(\frac{1}{N} \right) \sum_{i=1}^N w_{y_i}^{(t)} \log(p_{i,y_i}) \quad (9)$$

$$Dice_k = \frac{2 \sum_{i=1}^N p_{i,k} I(y_i = k) + \varepsilon}{\sum_{i=1}^N p_{i,k} + \sum_{i=1}^N I(y_i = k) + \varepsilon} \quad (10)$$

$$L_{Dice} = \frac{\sum_{k=1}^K w_k^{(t)} (1 - Dice_k)}{\sum_{k=1}^K w_k^{(t)} + \varepsilon} \quad (11)$$

Here, N is the number of valid pixels, $p_{i,k}$ is the predicted probability that pixel i belongs to class k , y_i is its ground-truth label, $I(\cdot)$ is the indicator function, and $Dice_k$ is the class- k overlap score. The same smoothed weight $w_k^{(t)}$ is used in weighted cross-entropy, weighted Dice, and the prototype loss defined in Equation (18). Each segmentation output is optimized with:

$$L_{seg} = \lambda_{CE} L_{CE} + \lambda_{Dice} L_{Dice} \quad (12)$$

where λ_{CE} and λ_{Dice} control the cross-entropy and Dice contributions, respectively. During training, the main output and auxiliary output are both supervised. In the complete CAP-DBNet, the prototype logits generated by CAPC also receive prototype-discrimination supervision. These supervision branches are used only during training and do not change the main inference path or introduce additional inference cost.

Figure 3 summarizes CAHL and its role in the complete training objective. Valid-pixel counts from the current mini-batch are transformed by inverse-frequency weighting, clipping, normalization, and EMA smoothing. The resulting $w_k^{(t)}$ values are shared by the main and auxiliary segmentation losses and by prototype supervision. The coefficients 1.0, 0.6, and 0.1 weight the main, auxiliary, and prototype losses, respectively. These branches affect training only and do not modify the inference graph.

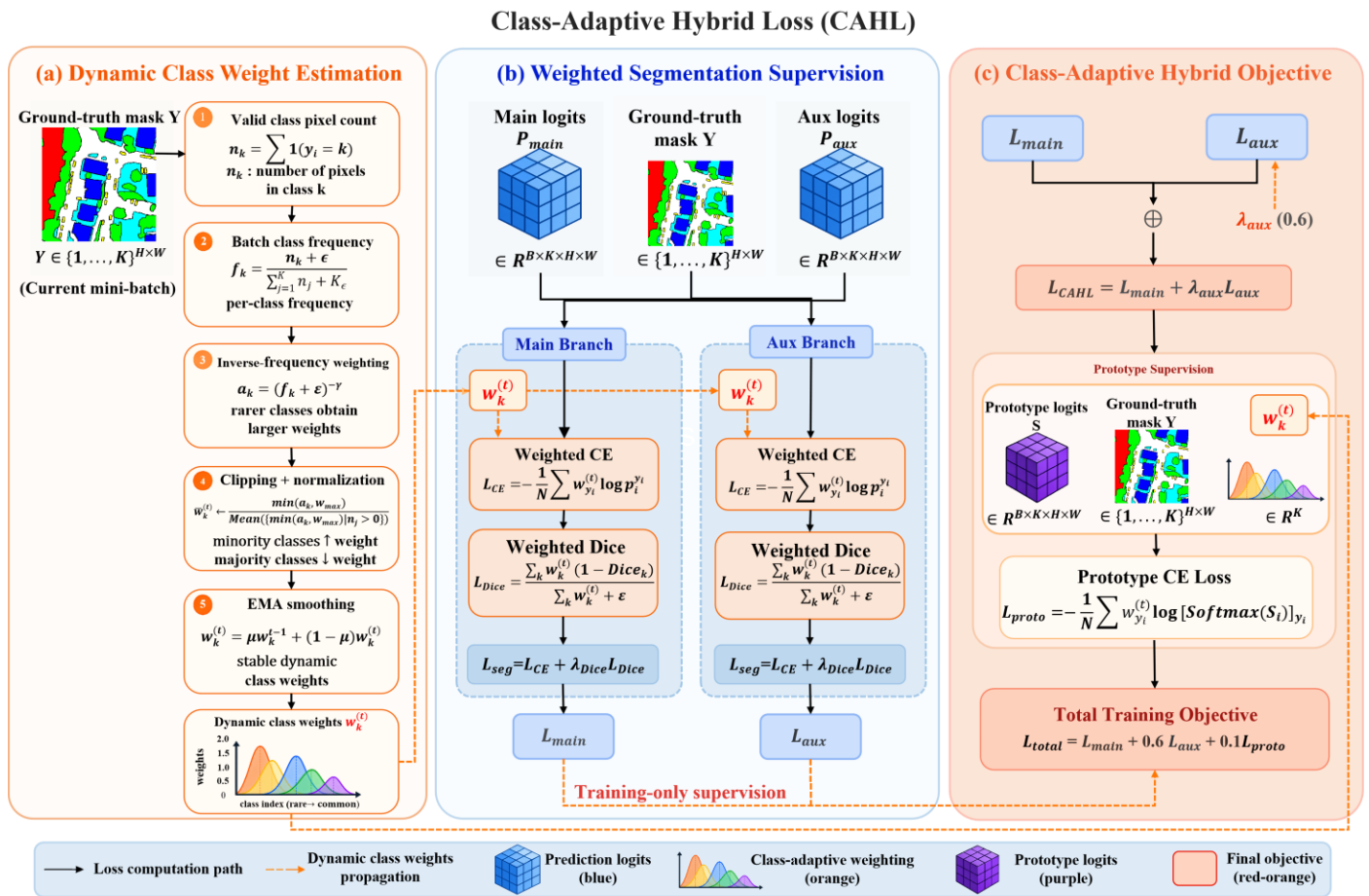


Figure 3. Class-adaptive hybrid loss and total training objective. The left, middle, and right panels show dynamic class-weight estimation, weighted segmentation supervision, and the class-adaptive hybrid objective with prototype supervision, respectively. Solid arrows indicate loss computation, and orange dashed arrows propagate the class weights. Blue denotes segmentation logits or segmentation-loss terms, purple denotes prototype logits, and orange denotes class-adaptive weighting. The circled plus denotes addition.

2.5. Class-Adaptive Prototype-Guided Calibration

CAHL changes supervision weights but does not impose a feature-space reference. CAPC therefore adds prototype-guided residual calibration and prototype supervision, as shown in Figure 4. Its K class prototypes are sample-independent learnable parameters: they are optimized during training, are not generated or updated from each input image, and remain fixed during inference. This definition distinguishes static CAPC prototypes from the input-dependent class centers C_s formed by the GLCD global branch.

Figure 4 gives the CAPC computation. Its input $\hat{F}$ is exactly the Stage-1 decoder output Z_1 shown in Figure 1, i.e., $\hat{F} = Z_1 \in \mathbb{R}^{B \times C \times H' \times W'}$. For pixel i , $f_i \in \mathbb{R}^C$ is the projected feature vector, and $p_k \in \mathbb{R}^C$ is the learnable prototype of class k . Temperature-scaled pixel-prototype similarity is used both to retrieve prototype context and to provide prototype supervision during training.

The decoder feature $\hat{F} = Z_1$ is distinct from the four encoder features $\{F_1, F_2, F_3, F_4\}$. The scalar f_k in CAHL denotes class frequency, whereas f_i in CAPC denotes a pixel feature vector; their subscripts and dimensionalities distinguish the two quantities.

The prototype bank $P = [p_1, \dots, p_K] \in \mathbb{R}^{K \times C}$ is shared across all samples. It is learned by gradient descent but is not an input-conditioned memory or a test-time prototype update mechanism.

After L2 normalization, the scaled similarity between pixel feature f_i and class prototype p_k is computed as follows, where τ is the temperature coefficient applied only to the pixel-prototype similarity branch:

$$s_{i,k} = \frac{\text{normalize}(f_i)^T \text{normalize}(p_k)}{\tau} \quad (13)$$

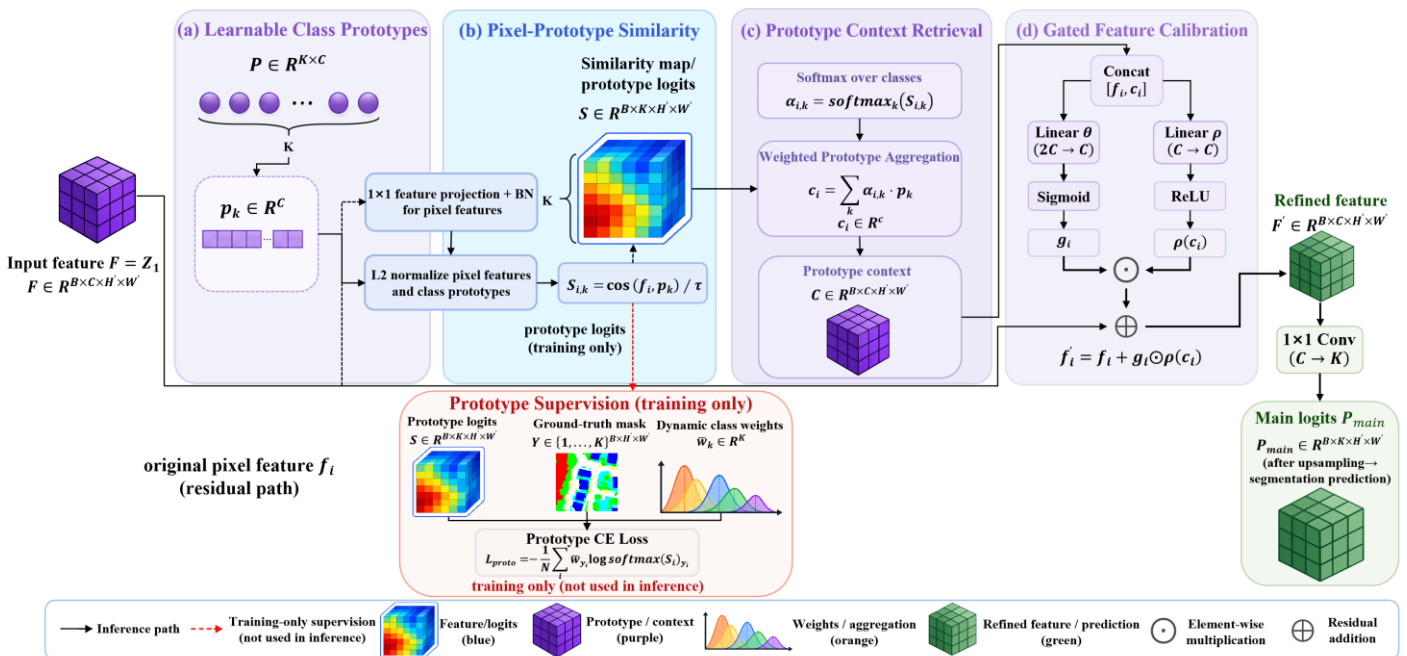


Figure 4. Class-adaptive prototype-guided calibration module and prototype supervision path. From left to right, the four panels show learnable class prototypes, pixel-prototype similarity, prototype-context retrieval, and gated feature calibration. Black solid arrows denote inference paths, and red dashed arrows denote training-only supervision. Purple denotes prototypes or prototype context, orange curves denote class weights, and green denotes refined features or predictions. The circled dot denotes element-wise multiplication, and the circled plus denotes residual addition.

A smaller τ produces a more concentrated prototype-attention distribution. Collecting all $s_{i,k}$ values yields the prototype similarity map S . A Softmax operation over its K class similarities converts the scaled scores into prototype attention weights:

$$a_{i,k} = \frac{\exp(s_{i,k})}{\sum_{j=1}^K \exp(s_{i,j})} \quad (14)$$

where $a_{i,k}$ is the attention weight from pixel i to prototype k . The resulting prototype context is:

$$c_i = \sum_{k=1}^K a_{(i,k)} p_k \quad (15)$$

where c_i is the C -dimensional prototype context retrieved for pixel i . The gate and calibrated feature are then defined as:

$$g_i = \sigma(W_{\text{gate}}[f_i; c_i]) \quad (16)$$

$$f'_i = f_i + g_i \odot \rho(c_i) \quad (17)$$

where W_{gate} is the gate mapping, $[\cdot; \cdot]$ denotes channel concatenation, $\rho(\cdot)$ projects prototype context into the feature space, and $\odot$ is element-wise multiplication. The residual form injects prototype context without replacing the original pixel feature.

The same similarity logits S are also used for auxiliary prototype discrimination:

$$L_{\text{proto}} = -\frac{1}{N} \sum_{i=1}^N w_{y_i}^{(t)} \log[\text{Softmax}(S_i)]_{y_i} \quad (18)$$

Here, $S_i \in \mathbb{R}^K$ is the similarity-logit vector at pixel i , y_i is its ground-truth class, N is the number of valid pixels, and $w_{y_i}^{(t)}$ is the same smoothed CAHL weight assigned to class y_i . This term encourages higher similarity to the correct class prototype while giving low-frequency classes the same training-time reweighting used by the segmentation losses.

CAHL and CAPC act at the levels of supervision optimization and class-representation calibration, respectively, and are linked by a unified class-adaptive supervision mechanism. CAHL adapts supervision weights to the mini-batch class distribution, allowing the optimization contribution of each class to vary with the composition of the training data. CAPC establishes pixel-class associations using shared learnable class prototypes, injects prototype context through a gated residual mechanism, and strengthens class-discrimination supervision through an auxiliary prototype-discrimination loss.

Their collaboration is further reflected in the training objective. The prototype-discrimination loss computed from the prototype-similarity logits generated by CAPC uses the same smoothed class weights as the main and auxiliary segmentation losses in CAHL. Class-frequency information therefore contributes to both segmentation optimization and prototype-discrimination learning, subjecting both supervision paths to a consistent class-adaptive weighting rule. CAHL dynamically adjusts class-supervision strength, while CAPC provides explicit class-semantic references and pixel-feature calibration. Shared class weights and joint optimization link these functions, forming a collaborative learning mechanism in which supervision optimization and feature representation complement each other.

2.6. Optimization Objective

The complete training objective combines the main segmentation loss, auxiliary segmentation loss, and prototype discrimination loss:

$$L_{\text{total}} = L_{\text{seg}}^{\text{main}} + \lambda_{\text{aux}} L_{\text{seg}}^{\text{aux}} + \lambda_{\text{proto}} L_{\text{proto}} \quad (19)$$

where both segmentation terms use CAHL. The reported configuration sets the auxiliary-loss coefficient to 0.6, the prototype-loss coefficient to 0.1, and τ to 0.1; τ is used only for CAPC pixel-prototype similarity and not for the final segmentation classifier.

The remaining reported settings are $\gamma = 0.4$, $\varepsilon = 10^{-6}$, $w_{\text{max}} = 3.0$, EMA momentum $\mu = 0.85$, $\lambda_{\text{CE}} = 1.0$ for all datasets, and $\lambda_{\text{Dice}} = 0.7$ for Vaihingen/Potsdam and 0.5 for LoveDA. These values are unchanged from the experimental configurations summarized in Section 2.7.

The complete training procedure of CAP-DBNet is summarized in Algorithm 1.

Algorithm 1. Training procedure of CAP-DBNet

Input: Training dataset $D = \{(x, y)\}$, network parameters θ including the learnable prototype bank P , and training hyperparameters.

Output: Optimized network parameters θ^* .

1. Initialize the RepViT M2.3 backbone with ImageNet-pretrained weights and initialize the remaining network parameters, including the learnable prototype bank P .
 2. For each training iteration $t = 1, 2, \dots$ do
-

-
3. Sample a mini-batch $\{(x_i, y_i)\}_{i=1}^B$ from D .
 4. Extract four-scale backbone features $\{F_s\}_{s=1}^4$ using RepViT M2.3 and project them into a unified channel dimension $C = 96$, obtaining $\{Z_s\}_{s=1}^4$.
 5. For $s = 4, 3, 2, 1$ do
 6. Perform top-down decoding. For $s < 4$, upsample the previous-stage fused feature, concatenate it with the current-scale feature, and apply convolutional fusion.
 7. Generate global contextual features through class-center aggregation and region-adaptive attention with regional sampling (Equations (1)–(1d)).
 8. Generate local features through window-based multi-head self-attention and depthwise separable convolution.
 9. Apply CBI (Equations (2) and (3)) and ABW (Equations (4) and (5)) to obtain F_s^{fuse} .
 10. If $s = 4$, generate auxiliary segmentation logits from F_4^{fuse} for deep supervision.
 11. End For
 12. Apply CAPC to the Stage-1 fused feature. Compute the temperature-scaled pixel–prototype similarity logits S between projected pixel features and the learnable prototypes P (Equation (13)).
 13. Compute prototype attention weights and retrieve prototype context features (Equations (14) and (15)).
 14. Obtain calibrated features through gated residual calibration (Equations (16) and (17)) and generate main segmentation logits using the classifier. Upsample the main and auxiliary logits to the target segmentation resolution.
 15. Estimate mini-batch class frequencies from valid labeled pixels and compute CAHL weights using inverse-frequency weighting, clipping, normalization, and EMA smoothing (Equations (6)–(8)).
 16. Compute the main and auxiliary segmentation losses using weighted cross-entropy and Dice Loss (Equations (9)–(12)).
 17. Compute the prototype-discrimination loss L_{proto} using the same smoothed CAHL class weights as the segmentation losses (Equation (18)).
 18. Calculate the total loss according to Equation (19).
 19. Backpropagate the total loss and update the network parameters θ , including the learnable prototype bank P , using the optimizer.
 20. End For
 21. Return θ^* .
-

Note: The auxiliary segmentation branch, CAHL weight estimation, and prototype-discrimination loss are used only during training and are disabled during inference. These training-only computations add no inference cost. CAPC feature calibration remains active during inference.

Use of Generative AI: Generative AI was used only for language polishing and sentence-level refinement of text originally written by the authors. It was not used to generate research content, experimental data, numerical results, figures, analyses, or scientific conclusions. All AI-assisted revisions were reviewed and approved by the authors.

2.7. Experimental Settings and Evaluation Metrics

All comparison models are locally implemented in PyTorch (version 2.0.0+cu118) and evaluated using validation-best checkpoints. The RepViT-based Baseline and CAP-DBNet initialize the RepViT M2.3 backbone from ImageNet-pretrained weights [40] and use AdamW [41] with a learning rate of 1×10^{-4} , weight decay of 1×10^{-4} , a polynomial schedule with power 0.9, 100 epochs, 32-bit floating-point precision (FP32) training, and

default train/validation batch sizes of 16/16. In the main comparison experiments, LOGCAN++, DOCNet, D2LS, and CAP-DBNet are each trained and evaluated using five different random seeds on Vaihingen, Potsdam, and LoveDA. Each method maintains the same data split, model configuration, training budget, and evaluation protocol across seeds. The remaining comparison methods report single-run results with a fixed random seed.

To improve comparability, the existing comparison experiments use the same dataset-specific splits, label mappings, and evaluation protocols, together with 512×512 training crops, a 100-epoch training budget, a fixed validation frequency, and FP32 precision. Method-specific optimization strategies, effective batch sizes, and backbone initialization are retained. LOGCAN++ and CAP-DBNet use the same RepViT M2.3 backbone initialization, batch sizes, and optimization settings, forming the main controlled comparison for assessing the model improvements. D2LS retains its method-specific configuration: an ImageNet-pretrained ConvNeXt-B backbone, AdamW with Lookahead, cosine annealing, and effective batch sizes of 4 on Vaihingen/Potsdam and 8 on LoveDA. D2LS is evaluated over five random seeds.

To further assess different segmentation architectures under the same backbone, an adapted UPerNet [42] with RepViT M2.3 is included as a supplementary comparison. This version retains the main UPerNet decoder architecture and replaces its encoder with RepViT M2.3. It is trained and evaluated with one fixed random seed on each of Vaihingen, Potsdam, and LoveDA. The data splits, label mappings, and evaluation metrics match those used for the main models. Its results are used only for descriptive same-backbone comparisons and are excluded from the five-seed paired significance tests. The adapted version uses the same ImageNet-pretrained initialization, number of training epochs, optimizer, batch sizes, and learning-rate settings as CAP-DBNet.

Common data splits and evaluation protocols do not imply identical optimization conditions across methods. D2LS and other methods still differ in their optimization strategies, effective batch sizes, and backbone initialization, and an equal number of epochs does not imply an equal number of parameter updates. These results are therefore local reproduction comparisons under common data and evaluation protocols, rather than strictly controlled comparisons with identical optimization conditions. LOGCAN++ and CAP-DBNet form the main controlled comparison. For the main comparison methods, two-sided paired t-tests use corresponding random seeds, with Holm correction across the nine main comparisons over the three datasets. The nine key leave-one-component-out comparisons undergo a separate Holm correction.

Table 3 summarizes the dataset splits, evaluated classes, crop size, epoch budget, and core Baseline/CAP-DBNet training configuration. LOGCAN++, DOCNet, D2LS, and CAP-DBNet are evaluated over five random seeds in the main comparison experiments, with means and sample standard deviations calculated for each metric to assess variation across runs. Other comparison methods retain single-run results, which are interpreted descriptively. The statistical settings for component ablation experiments are described separately in Sections 3.3 and 3.4.

Computational complexity is directly profiled at 1024×1024 input resolution with batch size 1 under a one-operation-per-multiple-accumulate convention. For LoveDA, D2LS checkpoints are selected using validation mIoU over all seven classes, while final performance is reported using the six foreground classes.

The evaluation metrics are class-level Intersection over Union (IoU), mean IoU (mIoU), mean accuracy (mACC), and mean class F1 (mF1). mACC and mF1 average per-class accuracy and F1, respectively, over the same foreground classes and excluded labels used for mIoU.

To specify the metric calculations, a confusion matrix is constructed from valid evaluation pixels. For each evaluated class k , IoU, mIoU, mean class accuracy (mACC), and mean class F1 (mF1) are defined as follows:

$$\text{IoU}_k = \frac{TP_k}{TP_k + FP_k + FN_k} \quad (20)$$

$$\text{mIoU} = \frac{1}{|C|} \times \sum_{k \in C} \text{IoU}_k \quad (21)$$

$$\text{mACC} = \frac{1}{|C|} \times \sum_{k \in C} \frac{TP_k}{TP_k + FN_k} \quad (22)$$

$$\text{mF1} = \frac{1}{|C|} \times \sum_{k \in C} \frac{2TP_k}{2TP_k + FP_k + FN_k} \quad (23)$$

Here, TP_k , FP_k , and FN_k denote true positives, false positives, and false negatives for class k , respectively. C is the set of evaluated classes and $|C|$ is its cardinality. mACC is the arithmetic mean of class recalls, and mF1 is the arithmetic mean of class F1 scores. The valid-pixel and ignored-label rules follow Section 2.7.

Specifically, Vaihingen and Potsdam average metrics over the five foreground classes ImSurf, Building, LowVeg, Tree, and Car. Clutter/Background is a valid class in confusion-matrix statistics but is excluded from the five-class average; Boundary pixels are ignored. LoveDA averages metrics over the six foreground classes: Building, Road, Water, Barren, Forest, and Agricultural. Background participates in model prediction and confusion-matrix statistics but is excluded from the foreground average. Unlabeled and ignored pixels are excluded from evaluation.

The following analysis uses mIoU as the primary metric and reports the other metrics under a consistent class set. Foreground-five-class mIoU and all-six-class mIoU including Clutter/Background use different averaging sets and cannot be directly compared numerically. To improve evaluation transparency, Supplementary File S2 provides single-run foreground-five-class mIoU, Clutter/Background IoU, and all-six-class mIoU for Vaihingen and Potsdam. This supplementary analysis is excluded from the significance tests for the five-seed experiments in the main text.

To reduce the influence of random variation in individual training runs, the four main comparison methods are evaluated using five different random seeds on each of the three datasets. Foreground class IoU, mIoU, mACC, and mF1 are calculated for each independent run. Arithmetic means and sample standard deviations over the five runs (Mean $\pm$ SD) describe average performance and variation across runs. The sample standard deviation uses the denominator $n - 1$, where $n = 5$. In the lower portions of Tables 4–6, class-level IoUs are five-run means, while mIoU, mACC, and mF1 are reported as Mean $\pm$ SD. The repeated-seed experiments retain the evaluation classes and ignored-label rules of the original experiments.

Table 4. Class-level IoU and overall performance comparison on the Vaihingen dataset.

Method	Backbone	ImSurf	Building	LowVeg	Tree	Car	mIoU	mACC	mF1	FLOPs (G)	Params (M)
DeepLabV3+ [3]	ResNet 50	84.55	89.19	69.83	79.87	63.31	77.35	86.89	86.90	59.85	39.76
UNetFormer [7]	ResNet 18	85.39	88.86	70.04	80.70	72.26	79.45	89.20	88.36	11.74	11.68
DCSwin [8]	Swin B	87.52	91.67	72.97	81.93	76.78	82.17	90.57	90.06	70.15	66.90

Pyramid-Mamba [9]	ResNet 18	79.52	80.84	67.11	79.36	59.25	73.22	83.81	84.24	102.30	115.12
OCRNet [39]	HRNetV2 W32	85.61	91.37	69.84	79.37	77.77	80.79	87.82	89.19	95.44	32.94
UPerNet [42]	RepViT M2.3	87.75	92.21	72.64	81.43	78.63	82.53	90.80	90.28	238.27	64.11
LOGCAN++ [11]	RepViT M2.3	87.67	91.99	74.18	82.68	82.13	83.73 ± 0.36	91.36 ± 0.17	91.03 ± 0.21	37.30	25.19
DOCNet [12]	HRNetV2 W32	87.76	91.36	74.45	82.3	80.66	83.31 ± 0.24	91.11 ± 0.18	90.78 ± 0.15	197.21	39.12
D2LS [19]	ConvNeXt-B	87.07	91.33	72.9	82.15	80.9	82.87 ± 0.29	91.02 ± 0.12	90.50 ± 0.17	374.24	94.03
CAP-DBNet	RepViT M2.3	88.98	92.74	75.63	83.84	84.61	85.16 ± 0.19	92.06 ± 0.12	91.88 ± 0.11	44.82	26.51

Note: The upper portion reports single-run results, and the lower portion reports five-random-seed results. Class IoUs and overall metrics in the upper portion are single-run values. In the lower portion, class IoUs are means, while mIoU, mACC, and mF1 are means ± sample standard deviations (Mean ± SD).

Table 5. Class-level IoU and overall performance comparison on the Potsdam dataset.

Method	Backbone	ImSurf	Building	LowVeg	Tree	Car	mIoU	mACC	mF1
DeepLabV3+ [3]	ResNet 50	86.01	92.87	76.52	78.84	88.32	84.51	84.95	91.49
UNetFormer [7]	ResNet 18	86.40	92.32	77.17	79.53	90.03	85.09	92.01	91.83
DCSwin [8]	Swin B	88.38	94.30	78.72	81.25	89.94	86.52	92.79	92.67
PyramidMamba [9]	ResNet 18	87.52	93.70	78.21	78.87	91.97	86.05	92.16	92.38
OCRNet [39]	HRNetV2 W32	86.01	91.66	75.91	78.49	91.00	84.61	85.00	91.54
UPerNet [42]	RepViT M2.3	88.06	94.34	78.18	79.6	91.55	86.35	93.4	92.54
LOGCAN++ [11]	RepViT M2.3	88.41	94.33	78.58	80.89	93.2	87.08 ± 0.22	93.68 ± 0.16	92.97 ± 0.12
DOCNet [12]	HRNetV2 W32	88.71	94.76	79.6	81.17	93.4	87.53 ± 0.23	93.66 ± 0.22	93.23 ± 0.13
D2LS [19]	ConvNeXt-B	88.16	94.83	78.24	80.77	93.23	87.04 ± 0.21	93.73 ± 0.23	92.94 ± 0.12
CAP-DBNet	RepViT M2.3	89.51	95.07	80.36	82.44	94.22	88.32 ± 0.26	93.99 ± 0.22	93.69 ± 0.15

Note: The upper portion reports single-run results, and the lower portion reports five-random-seed results. Class IoUs and overall metrics in the upper portion are single-run values. In the lower portion, class IoUs are means, while mIoU, mACC, and mF1 are means ± sample standard deviations (Mean ± SD).

Table 6. Six-foreground-class IoU and overall performance comparison on the LoveDA dataset.

Method	Backbone	Agricultural	Building	Road	Water	Barren	Forest	mIoU	mACC	mF1
DeepLabV3+ [3]	ResNet 50	52.93	57.23	53.16	65.65	36.09	41.76	51.14	67.21	67.11
UNetFormer [7]	ResNet 18	54.54	57.13	52.97	64.11	34.06	41.90	50.79	64.17	66.76
DCSwin [8]	Swin B	52.50	60.82	57.25	70.72	37.55	45.15	54.00	69.19	69.49
PyramidMamba [9]	ResNet 18	50.38	53.87	56.74	71.19	20.91	40.11	48.87	63.79	64.07
OCRNet [39]	ResNet 50	46.15	58.75	57.18	80.63	13.27	46.94	50.49	64.26	63.95
UPerNet [42]	RepViT M2.3	53.41	64.37	55.72	69.89	35.56	38.86	52.97	66.98	67.69
LOGCAN++ [11]	RepViT M2.3	54.74	65.22	54.86	71.95	37.71	41.43	54.31 ± 0.26	68.95 ± 0.36	69.59 ± 0.22
DOCNet [12]	HRNetV2 W32	56.66	67.03	57.78	71.93	31.42	43.83	54.78 ± 0.31	67.30 ± 0.25	69.71 ± 0.27
D2LS [19]	ConvNeXt-B	53.85	66.2	57	71.84	31.19	43.55	53.94 ± 0.29	65.70 ± 0.35	69.02 ± 0.24
CAP-DBNet	RepViT M2.3	55.3	67.6	58.1	72.47	38.07	42.97	55.75 ± 0.34	69.30 ± 0.32	70.78 ± 0.28

Note: The upper portion reports single-run results, and the lower portion reports five-random-seed results. Class IoUs and overall metrics in the upper portion are single-run values. In the lower

portion, class IoUs are means, while mIoU, mACC, and mF1 are means $\pm$ sample standard deviations (Mean $\pm$ SD).

Additional experiments assess the collaborative mechanisms and the rationale for key training parameters through collaborative-mechanism ablation, batch-size robustness analysis, and local-attention-window sensitivity analysis. Collaborative-mechanism ablation uses five random seeds on Vaihingen, Potsdam, and LoveDA and reports mIoU as Mean $\pm$ SD. Batch-size and local-window sensitivity analyses use single runs with a fixed random seed to describe performance changes across configurations. Except for the parameter or mechanism under study, all settings retain the default configuration of the corresponding dataset. The statistical scope of each experiment is stated in its table.

3. Results

3.1. Comparison with State-of-the-Art Methods

To evaluate CAP-DBNet, comparisons are made with representative segmentation methods, including DeepLabV3+, UNetFormer, DCSwin, OCRNet, DOCNet, Pyramid-Mamba, LOGCAN++, D2LS, and UPerNet [42]. The results in Tables 4–6 are obtained from local implementations using the data splits and evaluation metrics in Sections 2.1 and 2.7. The same-backbone baseline is an adaptation of LOGCAN++ [11] with RepViT M2.3, rather than the earlier LoG-CAN model. It uses the same backbone initialization and training protocol as CAP-DBNet to assess the improvements introduced by the additional modules. Other methods retain their respective training configurations, with the scope of comparability described in Section 2.7. Because LOGCAN++ and CAP-DBNet share the RepViT M2.3 backbone, pretrained initialization, optimization settings, and evaluation protocol, their performance differences more directly reflect the overall contribution of GLCD, CAHL, and CAPC. DOCNet and D2LS use different backbones and some different training settings, providing supplementary comparisons across architectures and computational scales; their performance differences are not attributed entirely to the added modules.

LOGCAN++ and CAP-DBNet form the main controlled same-backbone comparison over five random seeds. UPerNet with RepViT M2.3 provides a supplementary same-backbone comparison using one random seed and is excluded from the five-seed paired significance tests.

Table 4 reports the Vaihingen segmentation results. Over five random seeds, CAP-DBNet achieves a mean mIoU of $(85.16 \pm 0.19)\%$, improving on the same-backbone LOGCAN++ result of $(83.73 \pm 0.36)\%$ by 1.43 percentage points, with positive mean IoU gains in all five foreground classes. Car IoU increases from 82.13% to 84.61%, a gain of 2.48 percentage points, the largest among the five classes. Car accounts for only approximately 1.42% of the evaluated foreground pixels and consists of small, sparsely distributed targets, making local structure and class supervision particularly relevant. The window-local branch of GLCD extracts fine spatial features and integrates global class semantics through bidirectional cross-branch interaction and adaptive weighting, while CAHL adjusts supervision weights according to mini-batch class frequencies. The Car improvement is consistent with these structural-modeling and class-adaptive-supervision objectives, indicating the framework's ability to accommodate small-scale classes.

For LowVeg, IoU increases from 74.18% to 75.63%, a gain of 1.45 percentage points. Tree, which shares some spectral and texture similarities with LowVeg, also improves from 82.68% to 83.84%, a gain of 1.16 percentage points. LowVeg often has irregular spatial distributions and complex transitions at boundaries with neighboring vegetation classes, requiring both regional semantic context and local texture information. GLCD combines features through global–local modeling, while CAPC provides class-semantic

references using learnable prototypes and calibrates pixel features through gated residuals. The joint improvements in LowVeg and Tree are consistent with the intended effect of this design. CAP-DBNet also exceeds DOCNet and D2LS in mean mIoU by approximately 1.85 and 2.29 percentage points, respectively. Its parameter count and FLOPs are approximately 1/3.55 and 1/8.35 of those of D2LS, indicating a favorable segmentation-accuracy-computational-complexity trade-off.

Table 5 reports the Potsdam segmentation results. Over five random seeds, CAP-DBNet achieves a mean mIoU of $(88.32 \pm 0.26)\%$, approximately 1.24 percentage points above the same-backbone LOGCAN++ result of $(87.08 \pm 0.22)\%$, with positive mean IoU gains in all five foreground classes. LowVeg IoU increases from 78.58% to 80.36%, a gain of 1.78 percentage points, the largest in this dataset; Tree improves from 80.89% to 82.44%, a gain of 1.55 percentage points. These vegetation classes share spectral and texture characteristics and are prone to semantic confusion at complex transitions between land-cover classes. GLCD combines class-level global semantics with local spatial structure, and CAPC supplies prototype-guided feature calibration to strengthen class representations. The improvements in both LowVeg and Tree illustrate the framework's application to vegetation segmentation.

For the small-scale Car class, IoU improves from 93.20% to 94.22%, a gain of 1.02 percentage points, despite the already high Car IoU of the same-backbone LOGCAN++. Car accounts for only approximately 1.66% of the evaluated Potsdam foreground pixels, making this improvement consistent with the local-structure modeling of GLCD and class-adaptive supervision of CAHL. Together with the 2.48-percentage-point Car gain on Vaihingen, these results show positive mean IoU gains for small vehicle targets in two aerial-image datasets with different spatial resolutions. Mean IoUs for ImSurf and Building also increase by 1.10 and 0.74 percentage points, respectively, indicating improvements across small targets, vegetation, and large-area land-cover classes. CAP-DBNet exceeds DOCNet and D2LS in mean mIoU by approximately 0.79 and 1.28 percentage points, respectively, further supporting its effectiveness for high-resolution remote sensing image segmentation.

Table 6 reports the six-foreground-class LoveDA results. Over five random seeds, CAP-DBNet achieves an mIoU of $(55.75 \pm 0.34)\%$, compared with $(54.31 \pm 0.26)\%$ for the same-backbone LOGCAN++, a mean improvement of approximately 1.44 percentage points. Mean IoU gains for Agricultural, Building, Road, Water, Barren, and Forest are 0.56, 2.38, 3.24, 0.52, 0.36, and 1.54 percentage points, respectively, with positive gains in all six foreground classes. The larger improvements in Building and Road indicate favorable performance on built structures and narrow linear features. Compared with D2LS, CAP-DBNet improves mean mIoU by approximately 1.81 percentage points and has higher mean IoUs for Agricultural, Building, Road, Water, and Barren, while D2LS retains a slight advantage for Forest.

Across the five-seed results in Tables 4–6, CAP-DBNet improves mean mIoU over the same-backbone LOGCAN++ by 1.43, 1.24, and 1.44 percentage points on Vaihingen, Potsdam, and LoveDA, respectively. Positive mean IoU gains in every evaluated foreground class indicate that the improvement is not confined to a single dataset or a dominant class. CAP-DBNet also exceeds DOCNet and D2LS in mean mIoU on all three datasets, providing consistent average performance gains. The five-seed overall results are summarized in Table 7, and the corresponding statistical tests are reported in Table 8.

To further control backbone differences, an adapted UPerNet with RepViT M2.3 is included as a supplementary comparison. Its single-run mIoUs on Vaihingen, Potsdam, and LoveDA are 82.53%, 86.35%, and 52.97%, respectively, compared with CAP-DBNet's five-seed means of 85.16%, 88.32%, and 55.75%. The corresponding differences are 2.63, 1.97, and 2.78 percentage points. Under the data splits and evaluation protocol used here,

CAP-DBNet therefore also provides favorable segmentation accuracy relative to another same-backbone architecture.

3.2. Robustness Analysis with Multiple Random Seeds

Table 7 summarizes the five-random-seed overall results of LOGCAN++, DOCNet, D2LS, and CAP-DBNet. Means and sample standard deviations describe average segmentation performance and variation across independent runs under the dataset-specific evaluation class sets. The statistics correspond to the repeated-seed results in Tables 4–6.

Table 7. Five-random-seed performance of the main comparison methods (Mean $\pm$ SD, %).

Dataset	Method	mIoU (%)	mACC (%)	mF1 (%)
Vaihingen	LOGCAN++	83.73 $\pm$ 0.36	91.36 $\pm$ 0.17	91.03 $\pm$ 0.21
Vaihingen	DOCNet	83.31 $\pm$ 0.24	91.11 $\pm$ 0.18	90.78 $\pm$ 0.15
Vaihingen	D2LS	82.87 $\pm$ 0.29	91.02 $\pm$ 0.12	90.50 $\pm$ 0.17
Vaihingen	CAP-DBNet	85.16 $\pm$ 0.19	92.06 $\pm$ 0.12	91.88 $\pm$ 0.11
Potsdam	LOGCAN++	87.08 $\pm$ 0.22	93.68 $\pm$ 0.16	92.97 $\pm$ 0.12
Potsdam	DOCNet	87.53 $\pm$ 0.23	93.66 $\pm$ 0.22	93.23 $\pm$ 0.13
Potsdam	D2LS	87.04 $\pm$ 0.21	93.73 $\pm$ 0.23	92.94 $\pm$ 0.12
Potsdam	CAP-DBNet	88.32 $\pm$ 0.26	93.99 $\pm$ 0.22	93.69 $\pm$ 0.15
LoveDA	LOGCAN++	54.31 $\pm$ 0.26	68.95 $\pm$ 0.36	69.59 $\pm$ 0.22
LoveDA	DOCNet	54.78 $\pm$ 0.31	67.30 $\pm$ 0.25	69.71 $\pm$ 0.27
LoveDA	D2LS	53.94 $\pm$ 0.29	65.70 $\pm$ 0.35	69.02 $\pm$ 0.24
LoveDA	CAP-DBNet	55.75 $\pm$ 0.34	69.30 $\pm$ 0.32	70.78 $\pm$ 0.28

Note: Results are means $\pm$ sample standard deviations over five random seeds. The evaluated foreground class sets follow the corresponding dataset protocols.

To assess whether performance differences between the main comparison methods are affected by random training variation, two-sided paired t-tests use the results obtained with five corresponding random seeds. The 95% confidence intervals are calculated for paired mean mIoU differences. Holm correction is applied to the nine comparisons across the three datasets, as reported in Table 8.

Table 8. Paired statistical comparisons of CAP-DBNet with the main comparison methods.

Dataset	Comparison	Mean mIoU Gain (pp)	95% Confidence Interval	Holm-Adjusted <i>p</i>
Vaihingen	CAP-DBNet vs. LOGCAN++	+1.43	[1.15, 1.71]	0.000276
Vaihingen	CAP-DBNet vs. DOCNet	+1.85	[1.78, 1.92]	0.00000219
Vaihingen	CAP-DBNet vs. D2LS	+2.29	[2.12, 2.46]	0.0000113
Potsdam	CAP-DBNet vs. LOGCAN++	+1.24	[1.16, 1.31]	0.00000838
Potsdam	CAP-DBNet vs. DOCNet	+0.79	[0.74, 0.84]	0.00000971
Potsdam	CAP-DBNet vs. D2LS	+1.28	[1.20, 1.35]	0.00000838
LoveDA	CAP-DBNet vs. LOGCAN++	+1.44	[0.99, 1.88]	0.000848
LoveDA	CAP-DBNet vs. DOCNet	+0.97	[0.94, 1.02]	0.00000224
LoveDA	CAP-DBNet vs. D2LS	+1.81	[1.56, 2.07]	0.000122

Note: Two-sided paired t-tests use five corresponding random seeds. Confidence intervals refer to paired mean differences. Holm correction is applied to the nine comparisons. Reported mean mIoU gains are calculated by subtracting the two-decimal mean mIoUs displayed in the main comparison tables; confidence intervals and *p*-values remain based on the original five-seed paired statistics.

Table 8 shows positive mIoU differences between CAP-DBNet and LOGCAN++, DOCNet, and D2LS on all three datasets. None of the 95% confidence intervals include

zero, and all nine Holm-adjusted p -values are below 0.001, indicating statistically significant differences under the tests used here.

3.3. Component Ablation Studies

The baseline is the RepViT M2.3 adaptation of LOGCAN++ [11]. The + GLCD configuration retains the existing LOGCAN++ class-center global modeling mechanism and introduces the window-local branch, bidirectional cross-branch interaction, and adaptive branch weighting. Subsequent configurations add CAHL and CAPC. Stage-wise and leave-one-component-out ablations are conducted on Vaihingen, Potsdam, and LoveDA. The stage-wise study adds modules in the order Baseline, + GLCD, + GLCD + CAHL, and complete CAP-DBNet, as shown in Table 9. The batch-size analysis is presented in Table 10. The leave-one-component-out configurations retain two modules to assess the contribution of the third, as shown in Table 11. Every configuration is evaluated using five random seeds.

Table 9. Ablation study on component contributions.

Setting	GLCD	CAHL	CAPC	Vaihingen mIoU	Vaihingen mACC	Potsdam mIoU	Potsdam mACC	LoveDA mIoU	LoveDA mACC
Baseline	X	X	X	83.73 ± 0.36	91.36 ± 0.17	87.08 ± 0.22	93.68 ± 0.16	54.31 ± 0.26	68.95 ± 0.36
+ GLCD	✓	X	X	84.38 ± 0.30	91.52 ± 0.16	87.52 ± 0.21	93.80 ± 0.15	54.68 ± 0.29	68.92 ± 0.35
+ GLCD + CAHL	✓	✓	X	84.94 ± 0.16	91.72 ± 0.11	87.88 ± 0.22	93.91 ± 0.17	55.08 ± 0.30	69.18 ± 0.29
CAP-DBNet	✓	✓	✓	85.16 ± 0.19	92.06 ± 0.12	88.32 ± 0.26	93.99 ± 0.22	55.75 ± 0.34	69.30 ± 0.32

Note: All configurations use five random seeds and report mIoU and mACC as Mean ± SD. ✓ and X denote enabled and disabled components, respectively.

Table 9 reports the five-seed component-ablation results across the three datasets. Adding GLCD increases mean mIoU by 0.65, 0.44, and 0.37 percentage points on Vaihingen, Potsdam, and LoveDA, respectively. Adding CAHL provides further gains of 0.56, 0.36, and 0.40 percentage points, and the final addition of CAPC contributes 0.22, 0.44, and 0.67 percentage points. Thus, each successive module provides a positive mean mIoU gain under the stage-wise configurations tested.

Differences in module gains across datasets may be associated with their scene characteristics and segmentation requirements. Vaihingen and Potsdam contain high-spatial-resolution aerial images with ground sampling distances of approximately 9 cm and 5 cm, respectively, and land-cover classes such as vehicles, buildings, low vegetation, and trees that differ in scale and shape. The mean mIoU gain from GLCD is 0.65 percentage points on Vaihingen, compared with 0.44 on Potsdam. Bidirectional interaction and adaptive fusion between the class-center global branch and window-local branch support regional semantic consistency and local structure, consistent with the demands of high-resolution aerial images for small targets, complex land-cover boundaries, and multi-scale feature fusion. On Potsdam, the subsequent additions of CAHL and CAPC yield gains of 0.36 and 0.44 percentage points, suggesting complementary improvements through feature fusion, class-adaptive supervision, and prototype calibration.

LoveDA contains urban and rural scenes with more diverse class compositions and spatial distributions, including large contiguous agricultural and water regions as well as roads, buildings, and confusable land-cover classes. According to the training-set foreground proportions in Table 2, Agricultural accounts for 44.73%, whereas Barren accounts for only 6.83%, indicating class-distribution imbalance. GLCD initially provides a mean mIoU gain of 0.37 percentage points; CAHL and CAPC then add 0.40 and 0.67 percentage points, with CAPC providing the largest stage-wise gain. This pattern is consistent with the class-distribution differences and semantic discrimination demands of LoveDA.

CAHL adapts supervision weights to mini-batch class frequencies, while CAPC supplies class-semantic references and feature calibration through shared learnable prototypes and gated residuals; shared class-adaptive weights link their training objectives. Across the three datasets, GLCD, CAHL, and CAPC provide complementary support for spatial structure, supervision allocation, and class representation, with stage-wise contributions that vary with scene characteristics and class composition.

Because CAHL estimates supervision weights from class-pixel frequencies in the current training mini-batch, batch size can affect both frequency estimation and optimization. To examine sensitivity to this parameter, CAP-DBNet is evaluated with batch sizes of 8, 16, and 32 on Vaihingen, Potsdam, and LoveDA, as shown in Table 10. The data splits, model architecture, optimizer, and number of training epochs are held consistent. A batch size of 16 is the default configuration.

Table 10. Effect of batch size on CAP-DBNet.

Batch Size	Vaihingen mIoU (%)	Potsdam mIoU (%)	LoveDA mIoU (%)
8	85.07	88.04	55.38
16 (default)	85.24	88.31	55.58
32	85.11	88.22	55.46

Note: Results are single runs with a fixed random seed. Batch size denotes the actual mini-batch used for CAHL class-frequency estimation; 16 is the default configuration.

Across the tested batch sizes, the mIoU ranges are 0.17, 0.27, and 0.20 percentage points on Vaihingen, Potsdam, and LoveDA, respectively, indicating relatively small variations. The default batch size of 16 achieves the highest mIoUs among the tested settings: 85.24%, 88.31%, and 55.58%, respectively. Batch sizes of 8 and 32 produce slightly lower performance without marked degradation.

These results indicate some robustness of the complete CAP-DBNet, including CAHL, to training batch size within the range tested. The EMA smoothing in CAHL is designed to reduce batch-to-batch variation in class weights. However, batch size also affects gradient estimation and the number of parameter updates, so the observed performance changes cannot be attributed entirely to the class-weight computation in CAHL.

3.4. Interaction Analysis of Components

The leave-one-component-out experiments further assess the contributions of GLCD, CAHL, and CAPC within the complete framework (Table 11). Compared with configurations that remove GLCD, CAHL, or CAPC, respectively, complete CAP-DBNet increases mean mIoU by 0.46, 0.34, and 0.22 percentage points on Vaihingen; 0.56, 0.41, and 0.44 on Potsdam; and 0.77, 0.39, and 0.67 on LoveDA. Two-sided paired *t*-tests using corresponding random seeds and Holm correction yield adjusted *p*-values below 0.001 for all nine comparisons. Each module thus provides an additional mean performance contribution when used with the other modules, supplying quantitative and statistical evidence for their complementarity within the complete framework.

Table 11. Leave-one-component-out analysis of CAP-DBNet (mIoU and mACC, Mean $\pm$ SD, %).

Setting	GLCD	CAHL	CAPC	Vaihingen mIoU	Vaihingen mACC	Potsdam mIoU	Potsdam mACC	LoveDA mIoU	LoveDA mACC
Baseline + CAHL + CAPC	X	✓	✓	84.70 $\pm$ 0.16	91.62 $\pm$ 0.15	87.76 $\pm$ 0.21	93.87 $\pm$ 0.18	54.98 $\pm$ 0.30	69.16 $\pm$ 0.33

Baseline + GLCD + CAPC	✓	×	✓	84.82 ± 0.17	91.70 ± 0.14	87.91 ± 0.21	93.91 ± 0.17	55.36 ± 0.28	69.24 ± 0.31
Baseline + GLCD + CAHL	✓	✓	×	84.94 ± 0.16	91.72 ± 0.11	87.88 ± 0.22	93.91 ± 0.17	55.08 ± 0.30	69.18 ± 0.29
CAP-DBNet	✓	✓	✓	85.16 ± 0.19	92.06 ± 0.12	88.32 ± 0.26	93.99 ± 0.22	55.75 ± 0.34	69.30 ± 0.32

Note: All configurations use five random seeds and report mIoU and mACC as mean ± sample standard deviation. ✓ and × denote enabled and disabled components. Two-sided paired t-tests use the corresponding random seeds; for the nine comparisons of the full model against the three leave-one-out configurations, all Holm-adjusted *p*-values are below 0.001.

In particular, adding CAPC while retaining GLCD and CAHL increases mean mIoU by 0.22, 0.44, and 0.67 percentage points on Vaihingen, Potsdam, and LoveDA, respectively; adding CAHL while retaining GLCD and CAPC yields gains of 0.34, 0.41, and 0.39 percentage points. Together with the class-adaptive weights shared by CAHL and CAPC in the training objective, these results indicate that supervision reweighting and prototype-based feature calibration each contribute to performance under the tested joint configurations. They further support the functional complementarity of CAHL and CAPC and the rationale for their joint design in a unified segmentation framework.

To assess whether performance gains arise from the specified collaboration mechanisms rather than simple module combination, mechanism ablations are conducted on Vaihingen, Potsdam, and LoveDA (Table 12). The backbone, global and local branches, CAHL, and CAPC are retained across the configurations. Simple concatenation replaces CBI and ABW with direct feature concatenation followed by convolutional fusion. The w/o CBI configuration retains ABW, whereas w/o ABW uses fixed equal-weight fusion. The configuration without prototype-loss weight sharing retains CAHL weighting for the segmentation losses and removes only the shared class-adaptive weights from the prototype-discrimination loss.

Table 12. Collaborative mechanism ablation study of CAP-DBNet (mIoU, Mean ± SD, %).

Setting	CBI	ABW	Shared Weights	Vaihingen	Potsdam	LoveDA
Simple concatenation	×	×	✓	84.39 ± 0.18	87.65 ± 0.29	54.88 ± 0.36
w/o CBI	×	✓	✓	84.67 ± 0.21	87.91 ± 0.23	55.17 ± 0.32
w/o ABW	✓	×	✓	84.85 ± 0.23	88.03 ± 0.28	55.34 ± 0.35
w/o prototype-loss weight sharing	✓	✓	×	84.81 ± 0.18	87.97 ± 0.24	55.28 ± 0.31
CAP-DBNet	✓	✓	✓	85.16 ± 0.19	88.32 ± 0.26	55.75 ± 0.34

Note: All configurations use five random seeds and report mIoU as Mean ± SD. ✓ and × denote enabled and disabled mechanisms. Shared weights indicate whether the CAPC prototype-discrimination loss uses the class-adaptive weights generated by CAHL.

Table 12 shows that complete CAP-DBNet achieves the highest mean mIoU on all three datasets. Compared with simple dual-branch concatenation, it improves mean mIoU by 0.77, 0.67, and 0.87 percentage points on Vaihingen, Potsdam, and LoveDA, respectively. Explicit cross-branch interaction and adaptive fusion through CBI and ABW therefore provide better segmentation performance than direct aggregation of global and local features in these experiments. Removing CBI reduces mean mIoU by 0.49, 0.41, and 0.58 percentage points, while removing ABW reduces it by 0.31, 0.29, and 0.41 percentage points, supporting the contributions of bidirectional feature interaction and dynamic branch weighting under the tested configurations.

Removing the class-weight sharing between CAHL and the CAPC prototype-discrimination loss reduces mean mIoU by 0.35, 0.35, and 0.47 percentage points on Vaihingen, Potsdam, and LoveDA, respectively. Applying class-frequency-aware supervision consistently to both segmentation and prototype-discrimination losses thus helps improve overall segmentation performance. These ablations provide quantitative support for the CAP-DBNet design at the levels of feature fusion and supervision optimization.

3.5. Hyperparameter Sensitivity Analysis

To examine sensitivity to key hyperparameters, one-variable-at-a-time sweeps (Table 13) are conducted on Vaihingen for the prototype temperature τ , prototype-loss weight λ_{proto} , auxiliary-loss weight λ_{aux} , and global-branch region-partition setting p . The local-attention window size is fixed at 8, and all other hyperparameters retain their default values. In addition, a separate local-attention-window sensitivity experiment is conducted on Vaihingen, Potsdam, and LoveDA to examine spatial modeling range and computational complexity (Table 14).

Table 13 reports one-variable-at-a-time sweeps for four hyperparameters on Vaihingen. The default values ($\tau = 0.1$, $\lambda_{\text{proto}} = 0.1$, $\lambda_{\text{aux}} = 0.6$, and $p = 4 \times 4$) achieve the highest mIoU in all four sweep groups. Because the four sweeps share the same default configuration, its 85.24 result is repeated for reference. The within-sweep ranges are 0.87, 0.74, 0.68, and 0.72 percentage points for τ , λ_{proto} , λ_{aux} , and p , respectively. Performance declines when prototype discrimination, supervision strength, or global-region partitioning deviates substantially from the defaults. The local-attention window size remains fixed at 8.

Table 13. Hyperparameter sensitivity on Vaihingen, measured by mIoU.

Hyperparameter	Candidate Value	mIoU	Candidate Value	mIoU
τ (prototype temperature)	0.01	84.71	0.05	85.03
	0.1	85.24	0.2	85.16
	0.5	84.37	—	—
λ_{proto} (prototype loss weight)	0.01	84.50	0.05	84.85
	0.1	85.24	0.2	85.19
	0.5	84.73	—	—
λ_{aux} (auxiliary weight)	0.2	84.56	0.4	84.88
	0.6	85.24	0.8	84.90
p (global-branch region partition)	2×2	84.52	4×4	85.24
	8×8	84.61	—	—

Note: This table reports single-random-seed hyperparameter experiments. The default mIoU of 85.24% uses a different statistical basis from the five-seed mean mIoU of 85.16% in Table 4.

The local attention window controls the range of spatial-neighborhood modeling and attention-computation cost. Window sizes of 4×4 , 8×8 , and 16×16 are tested while the global-branch region partition remains $p = (4,4)$ and the other model settings remain unchanged, as reported in Table 14.

Table 14. Sensitivity analysis of local window size.

Window Size	Vaihingen mIoU (%)	Potsdam mIoU (%)	LoveDA mIoU (%)	FLOPs (G)
4×4	84.97	87.92	55.29	41.26
8×8 (default)	85.24	88.31	55.58	44.82
16×16	85.13	88.24	55.42	53.63

Note: mIoU values are single-run results with a fixed random seed. FLOPs are profiled at 1024×1024 input resolution and batch size 1. Only the local window size varies; the global-branch region partition remains $p = (4,4)$.

With a 4×4 window, mIoU is 84.97%, 87.92%, and 55.29% on Vaihingen, Potsdam, and LoveDA, respectively, at 41.26 G FLOPs. Increasing the window to 8×8 improves mIoU by 0.27, 0.39, and 0.29 percentage points to 85.24%, 88.31%, and 55.58%, respectively, with 44.82 G FLOPs. A moderate expansion of the local attention range therefore improves segmentation performance in the tested settings.

Increasing the window further to 16×16 raises the computation to 53.63 G FLOPs, approximately 19.7% above the 8×8 configuration, while mIoU decreases by 0.11, 0.07, and 0.16 percentage points on the three datasets. The larger window therefore provides no additional accuracy benefit while increasing computation. Across the tested configurations, the 8×8 window achieves the best segmentation accuracy and a favorable accuracy–computational-complexity trade-off, and is used as the default local-attention window.

3.6. Computational Efficiency and Accuracy-Efficiency Trade-Off

CAHL introduces no additional parameters or FLOPs during inference; CAPC adds 0.03 M parameters (26.48M $\rightarrow$ 26.51M) and 0.62 G FLOPs (44.20G $\rightarrow$ 44.82G) relative to + GLCD + CAHL. Throughput is reported in frames per second (FPS).

Table 15 reports efficiency at 1024×1024 input resolution. GLCD accounts for most of the added inference cost, increasing the model from 25.19 M to 26.48 M parameters and from 37.30 G to 44.20 G FLOPs. CAHL does not modify the inference graph, and CAPC adds only 0.03 M parameters and 0.62 G FLOPs. Relative to Baseline, the complete model increases parameters by approximately 5.2% and FLOPs by 20.2%, while FPS decreases from 40.5 to 29.7 (26.7%). The resulting 29.7 FPS is reported without a real-time claim. Compared with D2LS in Table 4, CAP-DBNet uses approximately 3.55 $\times$ fewer parameters and 8.35 $\times$ fewer FLOPs. FPS is measured with batch size 1 and FP32 inference after 20 warm-up iterations and 100 timed forward passes, with torch.cuda.synchronize() before and after each pass and host-to-device transfer excluded.

Table 15. Efficiency on an NVIDIA GeForce RTX 5090 GPU (NVIDIA Corporation, Santa Clara, CA, USA) for FP32 inference with batch size 1 and 1024×1024 input.

Setting	Params (M)	FLOPs (G)	FPS
Baseline	25.19	37.30	40.5
+ GLCD	26.48	44.20	31.1
+ GLCD + CAHL	26.48	44.20	31.1
CAP-DBNet (ours)	26.51	44.82	29.7

To examine the effect of computational-budget differences on performance comparisons, parameter count and inference computation are considered separately. The same-backbone LOGCAN++ baseline has 25.19 M parameters, close to CAP-DBNet’s 26.51 M, providing a controlled comparison at similar parameter scales. The internal + GLCD configuration has 26.48 M parameters and 44.20 G FLOPs, close to the complete model’s 26.51 M and 44.82 G. At these approximately matched computational budgets, complete CAP-DBNet improves mean mIoU over + GLCD by 0.78, 0.80, and 1.07 percentage points on Vaihingen, Potsdam, and LoveDA, respectively. These gains therefore do not accompany a large increase in computation. The + GLCD configuration is an internal ablation rather than an independent external baseline; this comparison supplements the analysis of accuracy relative to computational budget.

Figure 5 shows mIoU against FLOPs for the plotted methods on Vaihingen. Among the methods compared here, UNetFormer, LOGCAN++, and CAP-DBNet form the Pareto-nondominated set, representing accuracy–computational-complexity trade-offs at different budgets. UNetFormer requires only 11.74 G FLOPs and achieves 79.45% mIoU in a single run, representing a lower-budget segmentation option; LOGCAN++ and CAP-DBNet offer higher accuracy at higher computational budgets.

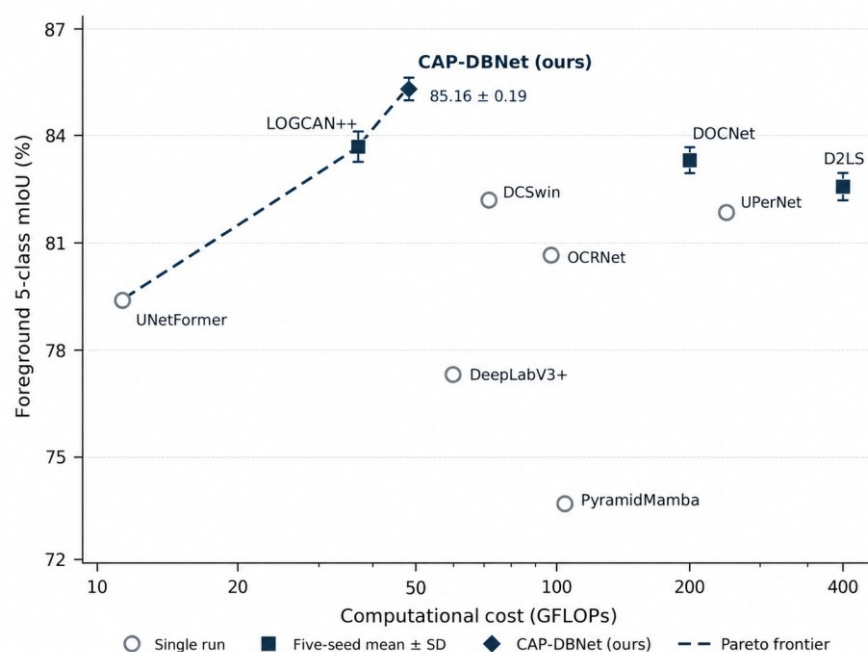


Figure 5. Pareto analysis of segmentation accuracy and computational complexity on Vaihingen.

Compared with LOGCAN++ using the same RepViT M2.3 backbone, CAP-DBNet increases FLOPs from 37.30 G to 44.82 G, approximately 20.2%, while raising five-seed mean mIoU from 83.73% to 85.16%, a gain of 1.43 percentage points. This improvement is obtained at additional computational cost rather than through simultaneous reductions in computation and gains in accuracy.

Table 4 also provides computational-complexity references for DCSwin and DOCNet. DCSwin uses a Swin-B backbone with 66.90 M parameters and 70.15 G FLOPs and achieves a single-run mIoU of 82.17% on Vaihingen. DOCNet uses HRNetV2-W32 with 39.12 M parameters and 197.21 G FLOPs and achieves a five-seed mean mIoU of 83.31%. CAP-DBNet has 26.51 M parameters and 44.82 G FLOPs and reaches a five-seed mean mIoU of 85.16%. Relative to DCSwin, its parameter count and FLOPs are lower by approximately 60.4% and 36.1%, respectively, and its reported mIoU is 2.99 percentage points higher. Relative to DOCNet, parameters and FLOPs are lower by approximately 32.2% and 77.3%, and mean mIoU is 1.85 percentage points higher. Computational complexity and segmentation accuracy are therefore not simply positively correlated across these architectures. The joint design of global–local decoding, class-adaptive supervision, and prototype-guided calibration provides a competitive combination of accuracy and computational complexity within the current comparison set.

Compared with the larger D2LS model, CAP-DBNet uses approximately 1/3.55 of the parameters and 1/8.35 of the FLOPs while achieving higher mean mIoU. Its computational-efficiency advantage thus has a defined comparison scope.

The supplementary UPerNet (RepViT M2.3) has 64.11 M parameters and 238.27 G FLOPs and achieves a single-run mIoU of 82.53% on Vaihingen. With the same backbone, CAP-DBNet has 26.51 M parameters and 44.82 G FLOPs and achieves a five-seed mean

mIoU of 85.16%. Under the reported results, it combines a smaller model and lower computation with higher segmentation accuracy, placing UPerNet outside the Pareto frontier of the current comparison set. Including this same-backbone comparison therefore leaves UNetFormer, LOGCAN++, and CAP-DBNet as the nondominated set shown in Figure 5.

3.7. Qualitative Results and Error Analysis

For each dataset, a consistent class-color mapping is used across all compared methods. For LoveDA, Building is red, Road is yellow, Water is blue, Barren is purple (159, 129, 183), Forest is green, Agricultural is orange (255, 195, 128), and Background is white. For Potsdam and Vaihingen, ImSurf is white, Building is blue, LowVeg is cyan, Tree is green, Car is yellow, and Clutter is red. These dataset-specific mappings ensure visual comparability within each benchmark.

The qualitative figures show predictions from complete CAP-DBNet and the comparison methods, including regional continuity, class transitions, boundary details, small-target responses, and visible misclassifications.

The qualitative comparison selects two representative samples from each dataset. In every panel, the first column is the input image, Mask denotes the ground truth, the remaining columns follow the displayed method labels, and the second row enlarges the red-boxed region. All methods are evaluated on the same sample, so the following discussion focuses on visible differences in regional continuity, class boundaries, and small-target responses and is interpreted together with Tables 4–8.

Figure 6 highlights narrow roads and building/bare-land boundaries in LoveDA. LOGCAN++ and CAP-DBNet both retain the main road corridor. Compared with some of the other methods, CAP-DBNet produces a comparatively continuous road response and cleaner adjacent building regions in the enlarged junction, although some local boundary leakage remains.

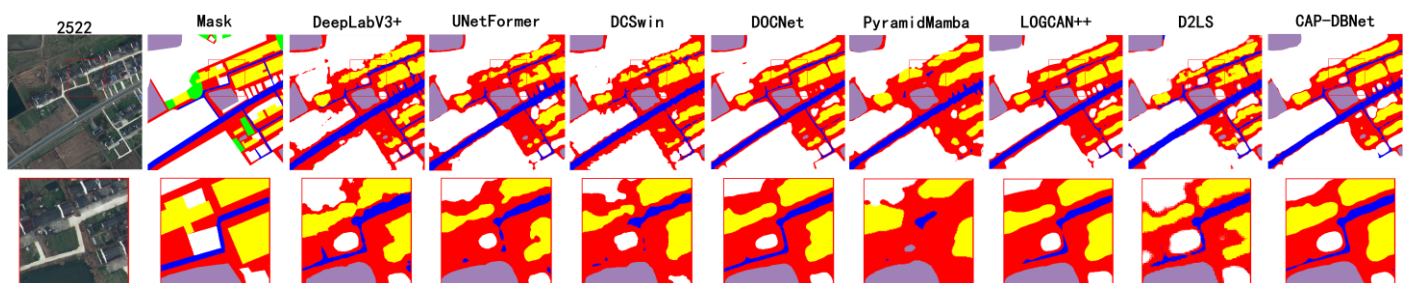


Figure 6. Semantic segmentation comparison of different methods on a representative LoveDA image with road and building boundaries. The top row shows the input image, ground-truth mask, and model predictions; the bottom row shows enlarged views of the regions outlined by the red boxes. Colors denote the land-cover classes: Building, red; Road, yellow; Water, blue; Barren, purple; Forest, green; Agricultural, orange; and Background, white.

Figure 7 examines a large water region intersecting with buildings and roads. CAP-DBNet preserves a continuous water band and a compact transition to the surrounding building layout; some comparison methods show irregular water boundaries or fragmented neighboring structures. A small number of transition pixels are also misclassified in the enlarged region.

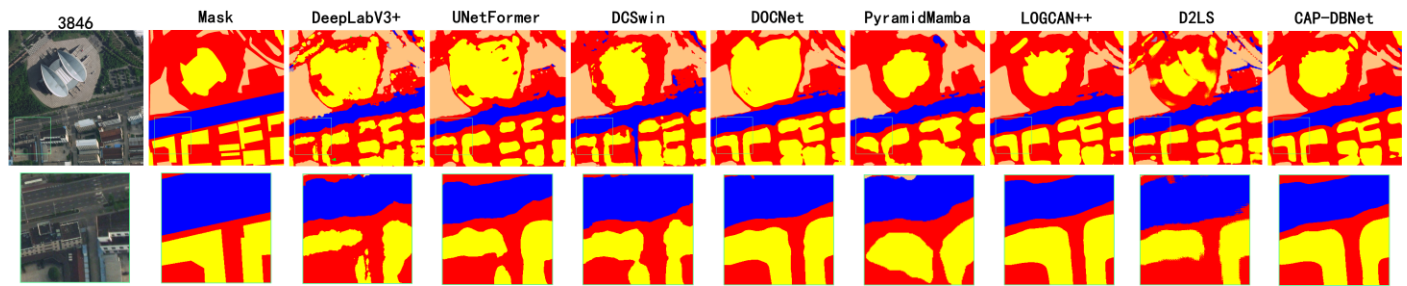


Figure 7. Semantic segmentation comparison of different methods on a LoveDA image with large-area land cover and building intersections. The top row shows the input image, ground-truth mask, and model predictions; the bottom row shows enlarged views of the regions outlined by the red boxes. Colors denote the land-cover classes: Building, red; Road, yellow; Water, blue; Barren, purple; Forest, green; Agricultural, orange; and Background, white.

Figure 8 focuses on building shadows, vehicles, and tree boundaries in Potsdam. In the enlarged region, CAP-DBNet recovers the main vehicle response that is missed or fragmented by several comparison methods, while LOGCAN++ and D2LS also retain most of the target. CAP-DBNet maintains a compact building boundary, although a small number of vehicle pixels remain missing.

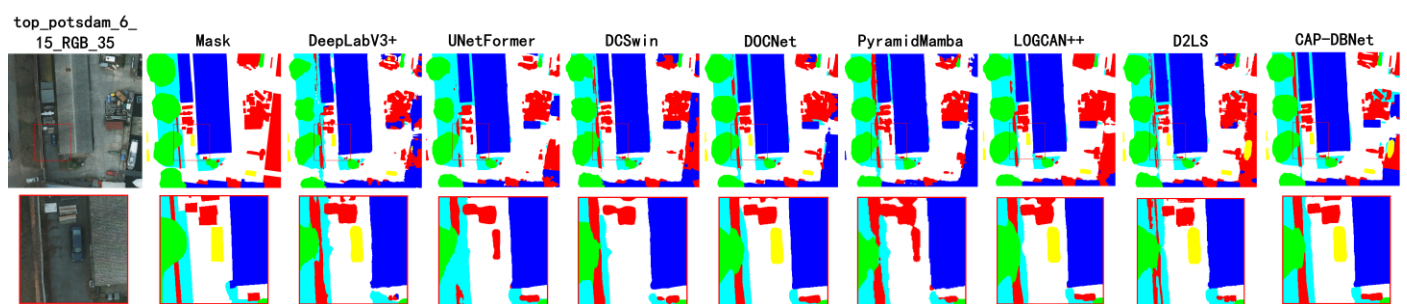


Figure 8. Semantic segmentation comparison of different methods on a Potsdam image with vehicles and building shadows. The top row shows the input image, ground-truth mask, and model predictions; the bottom row shows enlarged views of the regions outlined by the red boxes. Colors denote the land-cover classes: ImSurf, white; Building, blue; LowVeg, cyan; Tree, green; Car, yellow; and Clutter, red.

Figure 9 contains a dense urban block with interleaved buildings, impervious surfaces, low vegetation, and vehicles. CAP-DBNet, LOGCAN++, and D2LS preserve the principal vehicle responses more completely than methods that suppress or merge these small regions. CAP-DBNet also maintains a comparatively clean low-vegetation/impervious-surface transition, with minor residual boundary confusion.

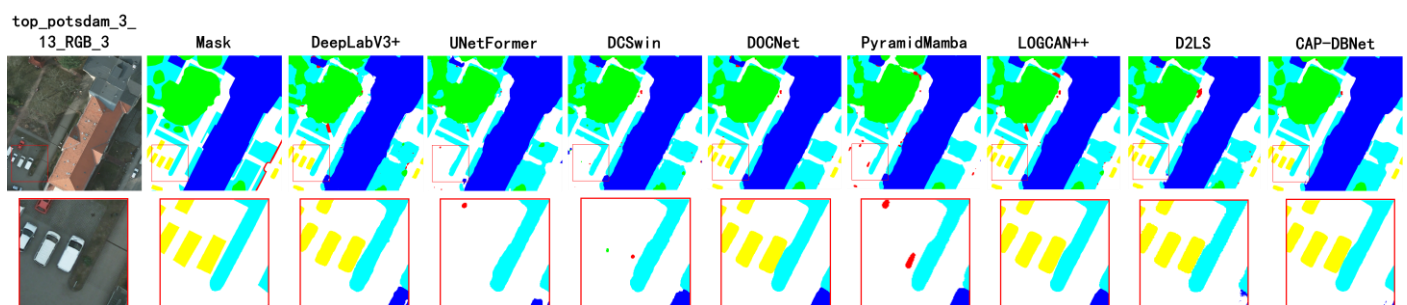


Figure 9. Semantic segmentation comparison of different methods on a Potsdam image with complex urban-block boundaries. The top row shows the input image, ground-truth mask, and model predictions; the bottom row shows enlarged views of the regions outlined by the red boxes. Colors

denote the land-cover classes: ImSurf, white; Building, blue; LowVeg, cyan; Tree, green; Car, yellow; and Clutter, red.

In Figure 10, buildings, impervious surfaces, low vegetation, and trees form a narrow transition zone. CAP-DBNet follows the thin LowVeg strip and the adjacent building and tree boundaries with fewer irregular intrusions than the most fragmented comparison outputs. LOGCAN++ and D2LS recover the same main structure, while CAP-DBNet produces a smoother transition; some LowVeg/Tree confusion remains.

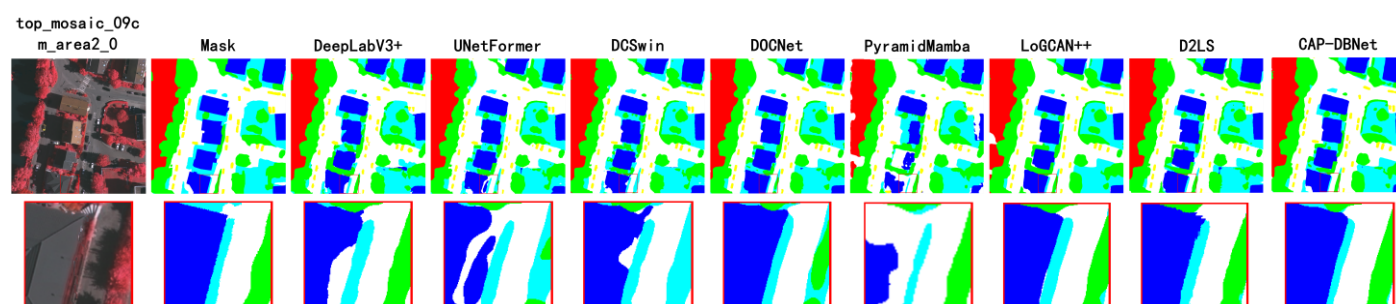


Figure 10. Semantic segmentation comparison of different methods on a Vaihingen image with vegetation-building transitions. The top row shows the input image, ground-truth mask, and model predictions; the bottom row shows enlarged views of the regions outlined by the red boxes. Colors denote the land-cover classes: ImSurf, white; Building, blue; LowVeg, cyan; Tree, green; Car, yellow; and Clutter, red.

Figure 11 combines roads, building shadows, vehicles, and vegetation patches. CAP-DBNet retains several small vehicle responses and the principal spatial layout of roads and vegetation. LOGCAN++ and D2LS also recover the main land-cover structures, while CAP-DBNet still produces some prediction errors around small targets and in locally occluded regions.

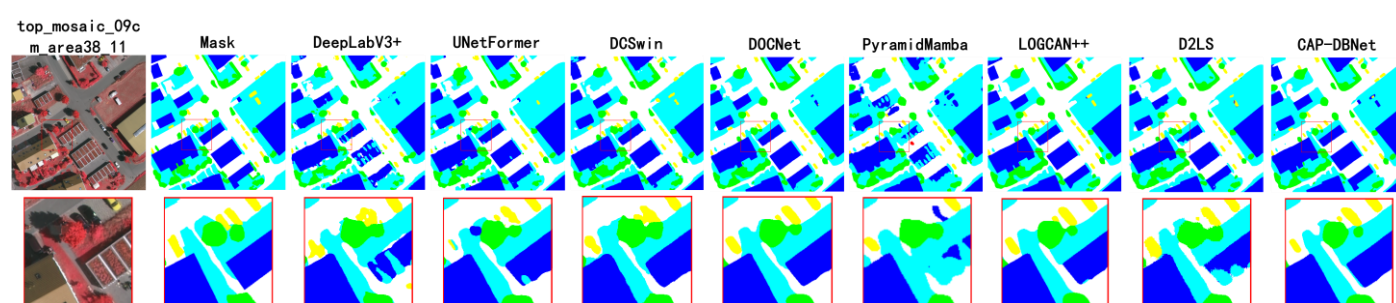


Figure 11. Semantic segmentation comparison of different methods on a Vaihingen image with small targets and occluded regions. The top row shows the input image, ground-truth mask, and model predictions; the bottom row shows enlarged views of the regions outlined by the red boxes. Colors denote the land-cover classes: ImSurf, white; Building, blue; LowVeg, cyan; Tree, green; Car, yellow; and Clutter, red.

Overall, the qualitative comparisons corroborate the quantitative results in Tables 4–6. CAP-DBNet demonstrates favorable segmentation performance in regional continuity, complex land-cover class transitions, and small-target retention, reflecting the application of joint global–local feature fusion, class-adaptive supervision, and prototype calibration. Together with the five-seed stagewise and leave-one-component-out results in Tables 9 and 11, these observations provide combined qualitative and quantitative support for the effectiveness of the overall framework.

4. Discussion

4.1. Performance Interpretation and Design Trade-Off

The performance improvement of CAP-DBNet benefits from the coordinated design of global–local representation learning, class-adaptive supervision, and prototype-guided feature calibration. GLCD enhances the interaction between global semantic context and local spatial structures through bidirectional feature exchange and adaptive branch fusion. CAHL dynamically adjusts supervision according to mini-batch class distributions to improve optimization under class-imbalanced conditions, while CAPC further refines pixel-level features through prototype-guided residual calibration.

The ablation experiments demonstrate that these components provide complementary contributions to the overall framework. The stagewise and leave-one-component-out evaluations verify the effectiveness of GLCD, CAHL, and CAPC across the Vaihingen, Potsdam, and LoveDA datasets. Furthermore, class-level analysis indicates that CAP-DBNet improves the recognition of both small-scale objects and large-area land-cover categories, suggesting its capability to model diverse spatial structures and complex semantic relationships in remote sensing scenes.

The accuracy improvement is achieved with additional computational cost introduced by feature interaction and prototype calibration. Compared with the same-backbone LOGCAN++ baseline, CAP-DBNet achieves higher segmentation accuracy with a moderate increase in parameters and FLOPs. Compared with heavier segmentation frameworks, CAP-DBNet maintains a favorable balance between accuracy and computational complexity, demonstrating its effectiveness as a practical framework for high-resolution remote sensing image semantic segmentation.

4.2. Limitations and Future Work

Although CAP-DBNet achieves competitive performance on three public remote sensing benchmarks, further studies can investigate its application in more diverse geographic regions and sensing conditions to evaluate broader generalization capability. In addition, further optimization of inference efficiency for specific deployment environments remains valuable. The current CAPC adopts a learnable prototype bank, and future work may explore more adaptive prototype updating strategies to improve robustness in complex remote sensing scenarios.

5. Conclusions

This paper presents CAP-DBNet, integrating the global–local collaborative decoder (GLCD), class-adaptive hybrid loss (CAHL), and class-adaptive prototype-guided calibration (CAPC) in a compact RepViT-based framework. Across five repeated experiments with different random seeds, CAP-DBNet achieves mean mIoUs of 85.16%, 88.32%, and 55.75% on Vaihingen, Potsdam, and LoveDA, respectively, improving over the same-backbone LOGCAN++ baseline by 1.43, 1.24, and 1.44 percentage points. Paired significance tests further support the observed performance differences among the main comparison methods. CAP-DBNet also uses fewer parameters and FLOPs than D2LS, demonstrating a favorable accuracy–computational-complexity trade-off. Further mechanism ablations show positive contributions from CBI bidirectional feature interaction, ABW adaptive branch weighting, and the class supervision weights shared by CAHL and CAPC. Batch-size and local-attention-window sensitivity experiments show relatively small performance variation over the tested batch sizes and a favorable balance between segmentation accuracy and computational complexity with the 8×8 local attention window. These results provide further experimental support for the collaborative design of

CAP-DBNet and the selection of key parameters. Future work will expand repeated experiments and explore scene-adaptive prototype calibration and boundary refinement.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/rs18203552/s1>, File S1: Tile-level training and validation splits for the ISPRS Vaihingen and Potsdam datasets; File S2: Single-run comparison of foreground five-class and all six-class mIoU on the Vaihingen and Potsdam datasets (Tables S2a and S2b).

Author Contributions: Conceptualization, S.S. and M.Y.; methodology, S.S.; software, S.S.; validation, S.S. and R.L.; formal analysis, S.S.; investigation, S.S.; resources, M.Y.; data curation, S.S.; writing—original draft preparation, S.S.; writing—review and editing, R.L. and M.Y.; visualization, S.S.; supervision, M.Y.; project administration, M.Y.; funding acquisition, M.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Inner Mongolia Autonomous Region’s Unveiling and Leadership Project, grant number 2024JBG50014.

Data Availability Statement: The datasets analyzed in this study are publicly available. LoveDA is available from its official public repository, and the Potsdam and Vaihingen datasets are available through the ISPRS 2D Semantic Labeling benchmark. No new dataset was created in this study. The source code and implementation files are not publicly available.

Acknowledgments: During the preparation of this manuscript, the authors used ChatGPT (OpenAI, GPT-5.5) for language polishing and sentence-level refinement of text originally written by the authors. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Abbreviation	Definition
ABW	Adaptive Branch Weighting
BN	Batch Normalization
CAHL	Class-Adaptive Hybrid Loss
CAPC	Class-Adaptive Prototype-Guided Calibration
CBI	Cross-Branch Interaction
FLOPs	Floating-Point Operations
FPS	Frames per Second
GAP	Global Average Pooling
GLCD	Global–Local Collaborative Decoder
GSD	Ground Sampling Distance
IoU	Intersection over Union
ISPRS	International Society for Photogrammetry and Remote Sensing
mACC	Mean Accuracy
mF1	Mean Class F1
mIoU	Mean Intersection over Union
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit

References

1. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3431–3440. <https://doi.org/10.1109/CVPR.2015.7298965>.

2. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), Munich, Germany, 5–9 October 2015; pp. 234–241. https://doi.org/10.1007/978-3-319-24574-4_28.
3. Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 833–851. https://doi.org/10.1007/978-3-030-01234-2_49.
4. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2881–2890. <https://doi.org/10.1109/CVPR.2017.660>.
5. Wang, J.; Sun, K.; Cheng, T.; Jiang, B.; Deng, C.; Zhao, Y.; Liu, D.; Mu, Y.; Tan, M.; Wang, X.; et al. Deep high-resolution representation learning for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 3349–3364. <https://doi.org/10.1109/TPAMI.2020.2983686>.
6. Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J.M.; Luo, P. SegFormer: Simple and efficient design for semantic segmentation with Transformers. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12077–12090.
7. Wang, L.; Li, R.; Zhang, C.; Fang, S.; Duan, C.; Meng, X.; Atkinson, P.M. UNetFormer: A UNet-like Transformer for efficient semantic segmentation of remote sensing urban scene imagery. *ISPRS J. Photogramm. Remote Sens.* **2022**, *190*, 196–214. <https://doi.org/10.1016/j.isprsjprs.2022.06.008>.
8. Wang, L.; Li, R.; Duan, C.; Zhang, C.; Meng, X.; Fang, S. A novel Transformer based semantic segmentation scheme for fine-resolution remote sensing images. *IEEE Geosci. Remote Sens. Lett.* **2022**, *19*, 6506105. <https://doi.org/10.1109/LGRS.2022.3143368>.
9. Wang, L.; Li, D.; Dong, S.; Meng, X.; Zhang, X.; Hong, D. PyramidMamba: Rethinking pyramid feature fusion with selective space state model for semantic segmentation of remote sensing imagery. *Int. J. Appl. Earth Obs. Geoinf.* **2025**, *144*, 104884. <https://doi.org/10.1016/j.jag.2025.104884>.
10. Wang, A.; Chen, H.; Lin, Z.; Han, J.; Ding, G. RepViT: Revisiting mobile CNN from ViT perspective. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024; pp. 15909–15920.
11. Ma, X.; Lian, R.; Wu, Z.; Guo, H.; Yang, F.; Ma, M.; Wu, S.; Du, Z.; Zhang, W.; Song, S. LOGCAN++: Adaptive Local-Global Class-Aware Network for Semantic Segmentation of Remote Sensing Images. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 4404216. <https://doi.org/10.1109/TGRS.2025.3541871>.
12. Ma, X.; Che, R.; Wang, X.; Ma, M.; Wu, S.; Feng, T.; Zhang, W. DOCNet: Dual-domain optimized class-aware network for remote sensing image segmentation. *IEEE Geosci. Remote Sens. Lett.* **2024**, *21*, 2500905. <https://doi.org/10.1109/LGRS.2024.3350211>.
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
14. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical vision Transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Virtually, 11–17 October 2021; pp. 10012–10022.
15. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999–3007.
16. Milletari, F.; Navab, N.; Ahmadi, S.-A. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In Proceedings of the International Conference on 3D Vision (3DV), Stanford, CA, USA, 25–28 October 2016; pp. 565–571. <https://doi.org/10.1109/3DV.2016.79>.
17. Salehi, S.S.M.; Erdogmus, D.; Gholipour, A. Tversky loss function for image segmentation using 3D fully convolutional deep networks. In Proceedings of the International Workshop on Machine Learning in Medical Imaging (MLMI), Quebec City, QC, Canada, 10 September 2017; pp. 379–387. https://doi.org/10.1007/978-3-319-67389-9_44.
18. Cui, Y.; Jia, M.; Lin, T.-Y.; Song, Y.; Belongie, S. Class-balanced loss based on effective number of samples. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 9268–9277.
19. Zou, X.; Li, Y.; Zhang, S.; Li, K.; Wang, S.; Tao, P.; Xing, J.; Lang, C. Dynamic dictionary learning for remote sensing image segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Honolulu, HI, USA, 19–23 October 2025; pp. 22457–22466. <https://doi.org/10.1109/ICCV51701.2025.02085>.

20. Ni, Y.; Liu, J.; Chi, W.; Wang, X.; Li, D. CGGLNet: Semantic segmentation network for remote sensing images based on category-guided global-local feature interaction. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5615617. <https://doi.org/10.1109/TGRS.2024.3379398>.
21. Liu, R.; Chen, B.; Yu, L.; Zhang, S. CRECA-Net: Class representation-enhanced class-aware network for semantic segmentation of high-resolution remote sensing images. *Remote Sens.* **2026**, *18*, 950. <https://doi.org/10.3390/rs18060950>.
22. Li, T.; Guo, L.; Xin, J.; Yu, H.; Li, B. GC2F-Net: A global category-center prior-guided spatial-frequency collaborative network for remote sensing semantic segmentation. *Remote Sens.* **2026**, *18*, 1600. <https://doi.org/10.3390/rs18101600>.
23. Volpi, M.; Tuia, D. Dense semantic labeling of sub-decimeter resolution images with convolutional neural networks. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 881–893. <https://doi.org/10.1109/TGRS.2016.2616585>.
24. Wang, J.; Zheng, Z.; Ma, A.; Lu, X.; Zhong, Y. LoveDA: A remote sensing land-cover dataset for domain adaptive semantic segmentation. *arXiv* **2021**, arXiv:2110.08733.
25. ISPRS WG II/4. 2D Semantic Labeling Contest. Available online: <https://www.isprs.org/resources/datasets/benchmarks/UrbanSemLab/semantic-labeling.aspx> (accessed on 11 June 2026).
26. Maggiori, E.; Tarabalka, Y.; Charpiat, G.; Alliez, P. High-resolution semantic labeling with convolutional neural networks. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 7092–7103. <https://doi.org/10.1109/TGRS.2017.2740362>.
27. Lin, T.-Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
28. Li, R.; Zheng, S.; Zhang, C.; Duan, C.; Wang, L. A2-FPN for semantic segmentation of fine-resolution remotely sensed images. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5607713. <https://doi.org/10.1109/TGRS.2021.3093977>.
29. Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; Sang, N. BiSeNet: Bilateral segmentation network for real-time semantic segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 334–349. https://doi.org/10.1007/978-3-030-01261-8_20.
30. Hong, Y.; Pan, H.; Sun, W.; Jia, Y. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *arXiv* **2021**, arXiv:2101.06085.
31. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
32. Zhang, H.; Xie, G.; Li, L.; Xie, X.; Ren, J. Frequency-domain guided Swin Transformer and global-local feature integration for remote sensing images semantic segmentation. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 5612611. <https://doi.org/10.1109/TGRS.2025.3535724>.
33. Jing, W.; Zhang, W.; Di, D.; Li, C.; Emam, M.; Mian, A. Hypergraph BiFormer for semantic segmentation of high-resolution remote sensing images. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 4406915. <https://doi.org/10.1109/TGRS.2025.3543556>.
34. Wang, L.; Wang, M.; Luo, X.; Wang, L.; He, M.; Long, C.; Ding, M. MSTNet: Multi-scale contextual analysis network for semantic segmentation of remote sensing images. *IET Image Process.* **2026**, *20*, e70326. <https://doi.org/10.1049/ipr2.70326>.
35. Snell, J.; Swersky, K.; Zemel, R. Prototypical networks for few-shot learning. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4077–4087.
36. Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; Krishnan, D. Supervised contrastive learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 18661–18673.
37. Zhou, T.; Wang, W.; Konukoglu, E.; Van Gool, L. Rethinking semantic segmentation: A prototype view. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–24 June 2022; pp. 2582–2593.
38. Liu, Y.; Liu, N.; Yao, X.; Han, J. Intermediate prototype mining Transformer for few-shot semantic segmentation. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 38020–38031. <https://doi.org/10.52202/068431-2755>.
39. Yuan, Y.; Chen, X.; Wang, J. Object-contextual representations for semantic segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Virtually, 23–28 August 2020. https://doi.org/10.1007/978-3-030-58539-6_11.
40. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; pp. 248–255. <https://doi.org/10.1109/CVPR.2009.5206848>.
41. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.

42. Xiao, T.; Liu, Y.; Zhou, B.; Jiang, Y.; Sun, J. Unified perceptual parsing for scene understanding. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 432–448. https://doi.org/10.1007/978-3-030-01228-1_26.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.