Reverse Difference Network for Highlighting Small Objects in Aerial Images

Huan Ni 1, Jocelyn Chanussot 2,*, Xiaonan Niu 3,*, Hong Tang 4 and Haiyan Guan 1

1 School of Remote Sensing & Geomatics Engineering, Nanjing University of Information Science & Technology, Nanjing 210044, China; nih@nuist.edu.cn or nih2015@yeah.net (H.N.); guanhy.nj@nuist.edu.cn (H.G.)
2 Jean Kuntzmann Laboratory (LJK), Institute of Engineering (Grenoble INP), CNRS, Grenoble Alpes University, INRIA, 38000 Grenoble, France; jocelyn.chanussot@grenoble-inp.fr
3 Nanjing Center, China Geological Survey, Nanjing 210016, China
4 Beijing Key Laboratory for Remote Sensing of Environment and Digital Cities, Faculty of Geographical Science, Beijing Normal University, Beijing 100875, China; tanghong@bnu.edu.cn
* Correspondence: niuxiaonan@mail.cgs.gov.cn

Abstract: The large-scale variation issue in high-resolution aerial images significantly lowers the accuracy of segmenting small objects. For a deep-learning-based semantic segmentation model, the main reason is that the deeper layers generate high-level semantics over considerably large receptive fields, thus improving the accuracy for large objects but ignoring small objects. Although the low-level features extracted by shallow layers contain small-object information, large-object information has predominant effects. When the model, using low-level features, is trained, the large objects push the small objects aside. This observation motivates us to propose a novel reverse difference mechanism (RDM). The RDM eliminates the predominant effects of large objects and highlights small objects from low-level features. Based on the RDM, a novel semantic segmentation method called the reverse difference network (RDNet) is designed. In the RDNet, a detailed stream is proposed to produce small-object semantics by enhancing the output of RDM. A contextual stream for generating high-level semantics is designed by fully accumulating contextual information to ensure the accuracy of the segmentation of large objects. Both high-level and small-object semantics are concatenated when the RDNet performs predictions. Thus, both small- and large-object information is depicted well. Two semantic segmentation benchmarks containing vital small objects are used to fully evaluate the performance of the RDNet. Compared with existing methods that exhibit good performance in segmenting small objects, the RDNet has lower computational complexity and achieves 3.9–18.9% higher accuracy in segmenting small objects.

Keywords: difference; semantic segmentation; convolutional networks; attention; deep learning

1. Introduction

With the improvement of the spatial resolution of aerial images, footpath (or cycle track)-level objects are recorded well. As shown in Figure 1, the bikes, drones, and pedestrians are of footpath-level sizes. Some applications, including urban monitoring, military reconnaissance, and national security, have urgent needs in terms of identifying small targets [1]. For example, pedestrian information is not only a data source for constructing urban human-flow patterns [2] but also useful for safe landing [3]. However, the identification of the footpath-level small targets encounters large-scale variation problems. Figure 1 shows the large-scale variations in aerial image datasets, including the UAVid [4] and Aerospaces [5]. In UAVid, pedestrians (Figure 1a,b) are considerably smaller than trees and roads. In Aerospaces, bicycles (Figure 1c) and drones (Figure 1d) are considerably smaller than roads and cars. This large-scale variation significantly lowers the accuracy of segmenting smaller objects. For example, in the published evaluations of Aerospaces,
the intersection-over-union (IoU) score of the bike category with the smallest size is only 15%, whereas that for the sky category with large objects is 94% [5].

The reason why the large-scale variation issue results in low accuracy of the segmentation of small objects has been studied by [6]. That is, most state-of-the-art methods, such as the pyramid scene parsing network (PSPNet) [7] and point-wise spatial attention network (PSANet) [8], focus on the accumulation of contextual information over significantly large receptive fields to generate high-level semantics. The high-level semantics (Figure 2c,e) extracted using deep convolutional neural network (CNN) layers mainly depict the holistic information of the large objects and ignore small objects [9,10]. Therefore, large objects achieve high accuracy; however, small objects have extremely low accuracy.

![Figure 1](image1.png)

Figure 1. The scale variations in aerial images, (a,b) are the image patches in UAVid, and (c,d) are the image patches in Aeroscapes. Pedestrians, bikes, and drones with small sizes are marked by red rectangles.

![Figure 2](image2.png)

Figure 2. The saliency maps of the low-level features and the high-level features (semantics). The low-level features (ResNet18 [11]) and the high-level features (ResNet18) are extracted by the first and fourth inner layers of ResNet18, respectively. The low-level features (BiSeNetV2) and the high-level semantics (BiSeNetV2) are extracted by the detail and semantic branches of BiSeNetV2, respectively.

Fortunately, the low-level features generated by the shallow layers contain small-object information, which is discovered by [9,10]. Consequently, several methods, such as the bilateral segmentation network (BiSeNet) [12], BiSeNet-v2 [13], and context aggregation network (CAgNet) [14] with dual branches, have been proposed and applied to remote sensing tasks. These methods set up a branch with shallow layers to extract low-level features that contain small-object information. The low-level features and high-level semantics extracted by the other branch are fused to make the final prediction. These methods improve the accuracy of the segmentation of small objects to some extent; however, their effects are limited. This is because the low-level features (Figure 2b,d) are the mixture of both the large and small objects. Specifically, the object details presented in Figure 2d are mainly for large objects, and we can hardly find the details for pedestrians. The low-level features extracted by shallow layers do not eliminate the predominant effects of large objects. Consequently, large objects push small objects aside when these models are trained.

Different to the dual-branch networks, a study by [15] uses a holistically nested edge detection (HED) [16] to produce closed contours with deep supervision. Semantic segmenta-
tion is obtained using SegNet [17] with the help of contours. This study achieves acceptable accuracy in the segmentation of relatively small objects (cars) in the ISPRS two-dimensional semantic labeling datasets (the Potsdam and Vaihingen datasets). However, cars are not small now in aerial images with the improvement of spatial resolution. For example, in the UAVid [4] and Aeroscapes [5] datasets, a range of objects (bikes, drones, and obstacles) that are considerably smaller than cars exist. Moreover, the contours extracted by HED exist for both large and small objects. Thus, the use of HED does not change the predominant relationship between large and small objects. Furthermore, the study in [15] uses a normalized digital surface model (nDSM) and DSM as the input of one of its model branches. However, nDSM and DSM are not always provided; thus, their application is limited.

To the best of our knowledge, the aforementioned small-object problem remains unsolved. We propose a reverse difference mechanism (RDM) to highlight small objects to address this issue. RDM can alter the predominant relationship between large and small objects. Thus, when the model is trained, small objects will not be pushed by large objects. RDM excludes large-object information from low-level features via the guidance of high-level semantics. The low-level features, which are a mixture of both large and small objects, can be the features produced by any shallow layer. The high-level semantics can be the features generated by any deep layer with large receptive fields. We design a novel neural architecture called a reverse difference network (RDNet) based on RDM. In RDNet, a detailed stream (DS) followed by RDM is proposed to obtain small-object semantics. Furthermore, a contextual stream (CS) is designed to generate high-level semantics to ensure sufficient accuracy in the segmentation of large objects. Both the small-object and high-level semantics are concatenated to make a prediction. The code of the RDNet will be available at https://github.com/yu-ni1989/RDNet, accessed on 21 July 2022. The contributions of this study are as follows.

- A reverse difference mechanism (RDM) is proposed to highlight small objects. RDM aligns the low-level features and high-level semantics and excludes the large-object information from the low-level features via the guidance of high-level semantics. Small objects are preferentially learned during training via RDM.
- Based on the RDM, a new semantic segmentation framework called RDNet is proposed. The RDNet significantly improves the accuracy of the segmentation of small objects. The inference speed and computational complexity of RDNet are acceptable for a resource-constrained GPU facility.
- In RDNet, the DS and CS are designed. The DS obtains more semantics for the outputs of RDM by modeling both spatial and channel correlations. The CS, which ensures the sufficient accuracy of the segmentation of large objects, produces high-level semantics by enlarging the receptive field. Consequently, the higher accuracy scores of the segmentation of both small and large objects are achieved.

2. Related Work

Semantic segmentation can be considered as a dense prediction problem, which is first solved by a fully convolutional network (FCN) [18]. Based on the FCN framework, a wide range of semantic segmentation networks has been proposed. In this study, we categorize these semantic segmentation networks into two groups: traditional and small-object-oriented methods. Additionally, semantic segmentation networks specified in remote sensing are reviewed.

2.1. Traditional Methods

Traditional semantic segmentation networks focus on overall accuracy improvement and do not consider the scale variation issue. These methods satisfy this objective primarily by solving the lack of contextual information issues in the FCN. This is caused by local receptive fields and short-range contextual information [19]. PSPNet [7], DeepLabv3 [20], and DeepLabv3+ [21] are designed with pyramid structures to pool the feature map into multiple resolutions to enlarge receptive fields. Furthermore, connected component la-
beling (CCL) [22] and a dual-graph convolutional network (DGCNet) [23] satisfy the same objectives via other ways. CCL designs a novel context that contrasts local features, and DGCNet uses two orthogonal graphs to model both spatial and channel relationships. To generate dense contextual information, an attention mechanism which facilitates the modeling of long-range dependency is commonly used. Representative methods include nonlocal networks [24], CCNet [25], dual attention network (DANet) [26], expectation-maximization attention network (EMANet) [27], and squeeze-and-attention network (SANet) [28]. We do not review them in detail because they share similar self-attention mechanisms. Recently, the vision transformers (ViTs) [29] extended the attention mechanism. Then, a range of networks such as the MobileViT [30], BANet [31], and semantic segmentation by early region proxy [32] based on the transformer mechanism are proposed. The transformer-based methods have less parameters, but they consume more GPU memory and rely on powerful GPU facilities.

Furthermore, numerous networks have focused on reducing the complexity of the model and accelerating the inference speed. To meet this objective, existing studies have replaced time-consuming backbones with lightweight backbones. For example, PSPNet [7] uses ResNet50 or ResNet101 [11] as its backbone. Both the training and inference of PSPNet are time-consuming. Recent real-time semantic segmentation studies, such as BiSeNet [12] and SFNet [33], replace ResNet101 with ResNet18 and obtain a better trade-off between inference speed and accuracy. In addition to ResNet18, recent studies have proposed a range of novel backbone networks to address real-time issues. The representative methods include MobileNet series [34,35], GhostNet [36], Xception [37], EfficientNet series [38,39], and STDC (Short-Term Dense Concatenate) [40]. Specifically, the Xception and MobileNet series use shortcut connections and depth-wise separable convolutions. The EfficientNet series use the depth-wise separable convolutions too, however, they focus on model scaling and achieve a good balance among the network depth, width, and resolution. GhostNet generates more feature maps from inexpensive operations with a series of linear transformations. STDC gradually reduces the dimensions of the feature maps and uses their aggregation for image representation. In this study, representative methods among them are tested in the experiments.

2.2. Small-Object-Oriented Methods

Recently, small-object-oriented methods, such as the gated fully fusion network (GFFNet) [6], have been introduced in several studies. GFFNet improves the accuracy of segmenting small/thin objects using a gated fully fusion (GFF) module to fuse multiple-level features. Then, multiple-level features can contribute to the segmentation simultaneously. Similar to GFFNet, SFNet [33] fuses the multiple-level features using an optical flow idea. It proposes a flow alignment module (FAM) to learn semantic flow between feature maps of adjacent levels. However, GFFNet and SFNet do not change the predominant effect of the large objects. Therefore, the improvements in the segmentation of small objects are limited.

Except for GFFNet and SFNet, most methods, such as BiSeNet [12], BiSeNetV2 [13], CAgNet [14], attentive bilateral contextual networks (ABCNet) [41], and the classification with an edge [15], improve the accuracy of the segmentation of small objects by integrating object details into the convolutions. BiSeNet, BiSeNet-v2, CAgNet, and ABCNet use dual branches to extract both object details and high-level semantics. Specifically, ABCNet integrates the self-attention mechanism into the dual-branch framework, thus it is a self-attention-based method at the same time. Classification with an edge [15] employs HED [16] to extract close contours of dominant objects and make predictions with the help of nDSM and DSM. Because the method in [15] requires diverse data inputs, it is difficult to compare it with other methods.

2.3. Semantic Segmentation Networks in Remote Sensing

In the remote sensing field, the studies by [42,43] lay the foundation for semantic segmentation. Subsequently, several excellent methods are proposed [44,45]. Among these,
the study by [46] proposes a set of distance maps to improve the performance of deep CNNs. The local attention-embedding network (LANet) [47], relation-augmented FCN (RA-FCN) [48], and cross-fusion net [49] further develop the attention mechanism. The study by [50] argues that existing methods lack foreground modeling and proposes a foreground-aware relation mechanism.

In terms of high-resolution aerial image datasets, Aeroscapes [5] and UAVid [4], based on unmanned aerial vehicle (UAV) images that provide complex scenes with oblique views, include a large range of small objects. Aeroscapes contain a large range of scenes that include rural landscapes, zoos, human residences, and the sea; however, scenes near urban streets are not included. The UAVid [4] dataset is published to address this issue. Coupled with the UAVid dataset in the study of [4], a multiscale-dilation net is proposed. Experiments demonstrate its superiority over existing algorithms. However, the accuracy of segmenting UAV images is relatively lower compared with that of segmenting natural images. This issue remains unsolved.

3. RDNet

The RDNet is constructed as shown in Figure 3. Figure 3 shows the neural architecture; only the RDM, DS, CS, and the loss functions are necessary to be presented in detail. For the backbone, although the experimental tests (Section 5.3) demonstrate the superior performance of ResNet18 in RDNet, we do not specify the backbone network. Any general-purpose network, such as Xception [37], MobileNetV3 [35], GhostNet [36], EfficientNetV2 [39], and STDC [40], can be integrated into RDNet, and we cannot ensure that ResNet18 achieves superior results compared with future work. The aforementioned general-purpose networks can be easily divided into four main layers (LY1, LY2, LY3, and LY4). If the input image patch has $H \times W$ pixels, LY1, LY2, LY3, and LY4 extract features with reduced resolutions $\frac{1}{4}H \times \frac{1}{4}W$, $\frac{1}{8}H \times \frac{1}{8}W$, $\frac{1}{16}H \times \frac{1}{16}W$, and $\frac{1}{32}H \times \frac{1}{32}W$, respectively, as shown in Figure 3. Notably, each LY1, LY2, LY3, and LY4 contains several sublayers. For example, ResNet18 has four main layers, which is well known. If we combine the convolution, batch norm, ReLU, and max-pooling operators at the beginning of ResNet18 with the first ample, ResNet18 has four main layers, which is well known.

3.3 Constructing the RDNet

Subsequently, $f_4^h$ is processed by the CS to generate high-level semantics $f_3^h \in \mathcal{R}^{(2C_1^h+2C_5^h)\times \frac{1}{8}H \times \frac{1}{8}W}$. $f_3^h$ and down-sampled $f_4^h$ are placed in an RDM (denoted by RDM1). Additionally, $f_4^h$ and $f_3^h$ are placed in another RDM (denoted by RDM2). Consequently, the difference features $f_4^d \in \mathcal{R}^{(2C_1^d+2C_5^d)\times \frac{1}{8}H \times \frac{1}{8}W}$ and $f_3^d \in \mathcal{R}^{2C_1^d\times \frac{1}{8}H \times \frac{1}{8}W}$ are produced. Then, $f_4^d$ and $f_3^d$ are concatenated and performed by the DS to generate the small-object semantics $f_2^s \in \mathcal{R}^{(2C_1^s+2C_5^s)\times \frac{1}{8}H \times \frac{1}{8}W}$. Thus, $f_2^s$ and $f_3^d$ mainly depict small and large objects, respectively. Finally, $f_2^s$ and $f_3^h$ are concatenated to obtain the final prediction $P^f$. 

Given an input image patch $I \in \mathcal{R}^{3\times H \times W}$, the backbone network is performed, and a set of features are generated. We select the features $f_1^h \in \mathcal{R}^{C_1^h\times \frac{1}{32}H \times \frac{1}{32}W}$ and $f_2^h \in \mathcal{R}^{C_1^h\times \frac{1}{2}H \times \frac{1}{2}W}$ generated by LY1, and LY2 as the low-level features. We present the saliency maps of the features extracted by LY1, LY2, LY3, and LY4 in ResNet18 in Figure 4 to demonstrate the rationale behind the selection for low-level features. From Figure 4b,c, $f_1^h$ and $f_2^h$ produced by ResNet18 are the mixture of both the large and small objects. Meanwhile, $f_3^h \in \mathcal{R}^{C_1^h\times \frac{1}{4}H \times \frac{1}{4}W}$ and $f_4^h \in \mathcal{R}^{C_1^h\times \frac{1}{8}H \times \frac{1}{8}W}$ generated by deep layers (LY3 and LY4 in ResNet18) mainly depict large objects (as shown in Figure 4d,e). We can hardly find the small-object information in $f_3^h$ and $f_4^h$. 

$P^f$ is demonstrated in the study by [46] to improve the performance of deep CNNs. The local attention-embedding network (LANet) [47], relation-augmented FCN (RA-FCN) [48], and cross-fusion net [49] further develop the attention mechanism. The study by [50] argues that existing methods lack foreground modeling and proposes a foreground-aware relation mechanism.
Figure 3. The architecture of reverse difference network. Down and Up are the interpolation operators for down-sampling and up-sampling.

Figure 4. The saliency maps of the features extracted by the inner layers in ResNet18. (a) is the input image patch, (b-e) are the saliency maps of the $f^b_1, f^b_2, f^b_3$, and $f^b_4$ extracted by $LY_1, LY_2, LY_3$, and $LY_4$ in ResNet18, respectively.

3.1. RDM

The low-level features $f^l \in \mathcal{R}^{C_l \times H_l \times W_l}$ (such as $f^b_2$ in Figure 4) extracted by shallow layers are the mixture of both the large and small objects. The high-level semantics $f^h \in \mathcal{R}^{C_h \times H_h \times W_h}$ generated by the deep layers primarily depict large objects (see the saliency map of $f^h$ in Figure 4). RDM attempts to exclude large objects from $f^l$ via the guidance of $f^h$. 
thus, the predominant effects of large objects are eliminated. The principle of RDM differs from the dual-branch framework and skip connection, which have some positive effects on the segmentation of small objects. However, they do not eliminate the predominant effects of large objects. Consequently, when these models are trained, large objects push small objects aside. RDM eliminates the predominant effects of large objects using the idea of alignment and difference, as shown in Figure 5. The key innovations are two-fold: (1) the reverse difference concept and (2) the semantic alignment between $f^l$ and $f^h$.

Figure 5. The proposed reverse difference mechanism. The $S(\cdot)$ is the sigmoid function.

The reverse difference concept lays the foundation for RDM. By considering the cosine alignment branch (see Figure 5) as an instance, after $f^l$ and $f^h$ are aligned, $f^h$ is transformed into $f^h_{\cos}$. Subsequently, a difference map $S(f^l) - S(f^h_{\cos})$ is produced by a difference operator. Here, $S(\cdot)$ is the Sigmoid function which transforms the intensity values in $f^l$ and $f^h_{\cos}$ into intervals from 0 to 1. Notably, the difference must subtract $f^h_{\cos}$ from $f^l$. Only in this manner can numerous intensity values on the positions of large objects in $S(f^l) - S(f^h_{\cos})$ be negative. Subsequently, we use the ReLU function to set all the negative values to zero. Consequently, the large-object information is washed out, and the small-object information is highlighted.

For semantic alignment between $f^l$ and $f^h$, $f^h$ typically has more channels and lower resolutions than $f^l$. The up-sampling and down-sampling can change the resolution but fail to change the number of channels. Even if they have the same number of channels, we cannot ensure semantic alignments between $f^l$ and $f^h$. For example, the $i$th channel in $f^h$ is more likely to contain a specific category of large objects. Does the $i$th channel in $f^l$ contain similar information of the same categories as the $i$th channel in $f^h$? If not, how do we ensure that the reverse difference mechanism is in effect? RDM provides two alignment modules to align the semantics from different perspectives by fully modeling the relationship between $f^h$ and $f^l$ as shown in Figure 5. The aligned high-level semantics produced by the cosine and neural alignments are $f^h_{\cos}$ and $f^h_{neu}$, respectively. The difference features $f^d \in \mathcal{R}^{2C_l \times H^l \times W^l}$ produced by RDM are computed as follows:

$$f^d_{\cos} = S(f^l) - S(f^h_{\cos}), \quad (1)$$
$$f^d_{neu} = S(f^l) - S(f^h_{neu}), \quad (2)$$
$$f^d = \text{ReLU}(\text{Cat}(f^d_{\cos}, f^d_{neu})), \quad (3)$$

where $\text{Cat}(\cdot)$ denotes the concatenation. In the following sections, the details of cosine and neural alignments are presented.

3.1.1. Cosine Alignment

The cosine alignment presented in Algorithm 1 determines the relationship between $f^h$ and $f^l$ without learning from the data. It has a mathematical explanation based on cosine similarity. First, we down-sample $f^l$ as $f^l_{down} \in \mathcal{R}^{C_l \times H^l \times W^l}$, which has the same resolution as $f^h$. Subsequently, $f^h \in \mathcal{R}^{C_h \times H^h \times W^h}$ and $f^l_{down}$ are converted into vector forms...
along each channel such that \( f^h \in \mathbb{R}^{C^h \times N^h} \) and \( f^l_{\text{down}} \in \mathbb{R}^{C^l \times N^h} \), where \( N^h = H^h \times W^h \).

The cosine similarity between each pair of channels in \( f^h \) and \( f^l_{\text{down}} \) is then computed as

\[
s_{ij}^{\cos} = \frac{f^l_{\text{down}}[i] \cdot f^h[j]}{|f^l_{\text{down}}[i]| \times |f^h[j]|},
\]

where \( f^l_{\text{down}}[i] \) and \( f^h[j] \) are vectors that belong to the \( i \)-th and \( j \)-th channels in \( f^l_{\text{down}} \) and \( f^h \), respectively. “\( \cdot \)” is the dot product between the vectors, and “\( \times \)” is the product between the scalars. After all pairs of vectors in \( f^h \) and \( f^l_{\text{down}} \) are performed using Equation (4), a similarity matrix \( M^{\text{sim}} \in \mathbb{R}^{C^l \times C^h} \) is constructed. We do not compute the cosine similarity per element to facilitate the implementation. The matrix multiplication “\( \ast \)” can be used to obtain \( M^{\text{sim}} \) as

\[
M^{\text{sim}} = \text{Norm}(f^l_{\text{down}}) \ast \text{Norm}(f^h)^T,
\]

where \( \text{Norm}(\cdot) \) is the \( l_2 \) normalization for each channel in \( f^l_{\text{down}} \) and \( f^h \). \( l_2 \) normalization is essential because it ensures that the matrix multiplication is equal to the cosine similarity. \( f^h)^T \) denotes the transpose of \( f^h \).

**Algorithm 1** Cosine alignment.

**Require:** The low-level features \( f^l \in \mathbb{R}^{C^l \times H^l \times W^l} \) and the high-level semantics \( f^h \in \mathbb{R}^{C^h \times H^h \times W^h} \)

**Ensure:** The aligned \( f^h \) (denoted by \( f^h_{\text{cos}} \in \mathbb{R}^{C^h \times H^h \times W^h} \))

1. Down-sample \( f^l \) as \( f^l_{\text{down}} \in \mathbb{R}^{C^l \times H^l \times W^l} \).
2. Transform \( f^l_{\text{down}} \) and \( f^h \) into vector forms along channels such that \( f^h \in \mathbb{R}^{C^h \times N^h} \) and \( f^l_{\text{down}} \in \mathbb{R}^{C^l \times N^h} \).
3. Produce the similarity matrix \( M^{\text{sim}} \in \mathbb{R}^{C^l \times C^h} \) using \( f^l_{\text{down}} \) and \( f^h \) as Equation (5).
4. Obtain \( f^h_{\text{cos}} \in \mathbb{R}^{C^h \times N^h} \) using \( M^{\text{sim}} \) and \( f^h \) as Equation (6).
5. Transform and up-sample \( f^h_{\text{cos}} \in \mathbb{R}^{C^h \times N^h} \) such that \( f^h_{\text{cos}} \in \mathbb{R}^{C^l \times H^l \times W^l} \).
6. return \( f^h_{\text{cos}} \)

A Softmax function is performed along the \( C^h \) dimension to render \( M^{\text{sim}} \) more representable. This transforms the elements in \( M^{\text{sim}} \) into a set of normalized weights for \( f^h \) channels. Finally, \( M^{\text{sim}} \) and \( f^h \) are multiplied as follows.

\[
f^h_{\text{cos}} = M^{\text{sim}} \ast f^h,
\]

where \( f^h_{\text{cos}} \) is the aligned \( f^h \). The multiplication “\( \ast \)” reduces the number of channels in \( f^h \) such that \( f^h_{\text{cos}} \) has the same number of channels as \( f^l \). Furthermore, using Equation (6), we can obtain an alignment effect. As shown in Figure 6, when we take the value \( f^h_{\text{cos}}(i, k) \) on the \( i \)-th row and \( k \)-th column in \( f^h_{\text{cos}} \) as an example, \( f^h_{\text{cos}}(i, k) \) is computed as:

\[
f^h_{\text{cos}}(i, k) = \sum_{j=1}^{C^h} M^{\text{sim}}(i, j) \times f^h(j, k),
\]

where \( f^h(j, k) \) is the value on the \( j \)-th row and \( k \)-th column in \( f^h \), and \( M^{\text{sim}}(i, j) \) is the value on the \( i \)-th row and \( j \)-th column in \( M^{\text{sim}} \). Consequently, \( f^h_{\text{cos}}(i, k) \) is a weighted average of \( f^h(j, k), j = 1, \ldots, C^h \). If the \( j \)-th channel in \( f^h \) is more similar to the \( i \)-th channel in \( f^h_{\text{down}} \), \( M^{\text{sim}}(i, j) \) will be larger. Therefore, the equation gives larger weights to the channels in \( f^h \) that are more similar to the \( i \)-th channel in \( f^h_{\text{down}} \). As a result, the cosine alignment tries its best to meet the objective that the \( i \)-th channel in \( f^l_{\text{down}} \) and the \( i \)-th channel in \( f^h_{\text{cos}} \) containing similar large object information.
Finally, $f_{\text{cos}}^h$ is transformed back into the image plane and up-sampled such that $f_{\text{cos}}^h \in \mathcal{R}^{C_l \times H^l \times W^l}$. $f_{\text{cos}}^h$ is the aligned high-level semantics produced by the cosine alignment.

### 3.1.2. Neural Alignment

The neural alignment, which is shown in Algorithm 2, aligns the semantics between $f^h$ and $f^l$ using convolutions. Firstly, a convolution layer with $1 \times 1$ kernels followed by an up-sampling operator is performed to compress $f^h$:

$$f_{\text{red}}^h = U_p(W_{\text{neu}}[C_l, 1 \times 1] \odot f^h),$$

(8)

where $W_{\text{neu}}[C_l, 1 \times 1]$ is the weight matrix in the convolution layer, $\odot$ is the convolution, $U_p(\cdot)$ is the up-sampling, and $f_{\text{red}}^h \in \mathcal{R}^{C_l \times H^l \times W^l}$ is the compressed $f^h$. Via Equation (8), $f_{\text{red}}^h$ has the same number of channels as $f^l$. Actually, $W_{\text{neu}}[C_l, 1 \times 1]$ is the learnable projection bases which projects $f^h$ into the space we prefer. This procedure has an effect of semantic alignment by iteratively optimizing the projection bases during the training.

**Algorithm 2 Neural alignment.**

**Require:** The low-level features $f^l \in \mathcal{R}^{C_l \times H^l \times W^l}$ and the high-level semantics $f^h \in \mathcal{R}^{C_l \times H^l \times W^l}$

**Ensure:** The aligned $f^h$ (denoted by $f_{\text{neu}}^h \in \mathcal{R}^{C_l \times H^l \times W^l}$)

1. Project and up-sample $f^h$ as Equation (8) to generate $f_{\text{red}}^h \in \mathcal{R}^{C_l \times H^l \times W^l}$.
2. Generate the channel attention vector $v^l \in \mathcal{R}^{C_l \times 1}$ via average pooling using both $f^l$ and $f_{\text{red}}^h$.
3. Transform $v^l$ into $\mathcal{R}^{1 \times H^l \times W^l}$ and model the correlation between the channels of $f^l$ and $f_{\text{red}}^h$ based on $v^l$.
4. Obtain $f_{\text{neu}}^h$ using $v^l$ as Equation (9).
5. return $f_{\text{neu}}^h$

Then, a mutual channel attention is proposed to enhance the alignment. The traditional channel attention has been used to select informative features via introducing global weights for channels of the input features [49,51]. However, the traditional channel attention is a self-attention mechanism which does not have the alignment effect between the features generated by different layers. Here, to consider both the information in $f^l \in \mathcal{R}^{C_l \times H^l \times W^l}$ and $f_{\text{red}}^h \in \mathcal{R}^{C_l \times H^l \times W^l}$, we concatenate $f^l$ and $f_{\text{red}}^h$ together. Based on this, the average pooling layer is performed to generate a channel attention vector $v^a \in \mathcal{R}^{2C_l \times 1}$. Subsequently, a convolution layer with $1 \times 1$ kernels, a batch normalization, and a sigmoid activation are performed on $v^a$ to model the correlations of the channels in $f^l$ and $f_{\text{red}}^h$. As a result, $v^a$ is transformed as $v^a \in \mathcal{R}^{C_l \times 1}$, which can be used directly to select the information of $f_{\text{red}}^h$ by an element-by-element multiplication along the $C_l$ channels. Concretely, using $v^a$, the aligned high-level semantics $f_{\text{neu}}^h \in \mathcal{R}^{C_l \times H^l \times W^l}$ are formulated as:

$$f_{\text{neu}}^h = v^a \odot f_{\text{red}}^h,$$

(9)

where $\odot$ is the element-by-element product along the $C_l$ channels. Finally, $f_{\text{neu}}^h$ is the output of the neural alignment.
3.2. DS

The $f^d$ generated by RDM is the input to DS. In RDNet, two RDM components produce difference features $f^d_1 \in \mathbb{R}^{2C^b_1 \times \frac{1}{8}H \times \frac{1}{8}W}$ and $f^d_2 \in \mathbb{R}^{2C^b_2 \times \frac{1}{8}H \times \frac{1}{8}W}$. $f^d_1$ and $f^d_2$ are concatenated as the final output $f^d \in \mathbb{R}^{C^d \times \frac{1}{8}H \times \frac{1}{8}W}$ of the RDM components, where $C^d = 2C^b_1 + 2C^b_2$, $H^d = \frac{1}{8}H$, and $W^d = \frac{1}{8}W$. $f^d$ includes only small-object information with fewer semantics. The DS is proposed to obtain more semantics to facilitate prediction without adding excessive parameters.

The commonly used method adds several convolutional layers with strides, such as ResNet18 to generate semantics. This method significantly increases the number of parameters and computational complexity. Here, we use depth-wise convolution to address these issues. Furthermore, when the stride is set to two or larger, the convolution enlarges the receptive field; however, some extremely small objects, such as pedestrians in UAVid, are ignored. Therefore, we do not use any strides. Additionally, we do not know which kernel size is most suitable. A larger kernel size suppresses small objects, whereas a smaller kernel size generates less semantic information. Here, the DS designs two convolutional branches using $1 \times 1$ and $3 \times 3$ kernels to alleviate the negative effects produced by a fixed kernel size, as shown in Figure 7.

**Figure 7.** The detailed stream (DS). The input of DS is the difference features $f^d$ extracted by RDM, and it generates small-object semantics $f^s$. Conv, convolution layer; DW Conv, depthwise convolution layer; APool, adaptive average pooling; Sigmoid and ReLU, activation functions; Expand, expanding based on duplication.

The first branch contains only a single convolutional layer with $1 \times 1$ kernels and is simple and lightweight. It convolves $f^d$ using a small kernel size and creates a correlation between all the channels. We use $f^d_{1 \times 1} \in \mathbb{R}^{C^d \times \frac{1}{8}H \times \frac{1}{8}W}$ to denote the output of the first branch. The second branch convolves $f^d$ using a larger kernel size. It makes spatial correlations using a depth-wise convolution with $3 \times 3$ kernels and makes a cross-channel correlation using shared kernels. Specifically, $f^d$ is first processed using a depth-wise convolutional layer to obtain a spatially correlated feature map $f^{sp} \in \mathbb{R}^{C^d \times \frac{1}{8}H \times \frac{1}{8}W}$. Subsequently, a cross-channel correlation is performed. This adaptively pools $f^{sp}$ into feature vector $v^{cc} \in \mathbb{R}^{C^d \times 1}$. A two-dimensional convolutional layer with padding and $3 \times 3$ kernels, followed by sigmoid activation, is then used to make correlations along the channels:

$$v^{cc} = S(W_c[2C^d, 2C^d, 3 \times 3] \odot P(v^{sp})), \quad (10)$$

where $v^{cc} \in \mathbb{R}^{C^d \times 1}$ is the derived cross-channel correlated feature vector, $P(\cdot)$ is padding, $S(\cdot)$ is sigmoid activation, and $W_c[2C^d, 2C^d, 3 \times 3]$ is the weight matrix of the convolution. Subsequently, the features $f^d_{3 \times 3} \in \mathbb{R}^{C^d \times \frac{1}{8}H \times \frac{1}{8}W}$ extracted by the second branch are:

$$f^d_{3 \times 3} = f^{sp} \bullet \text{Expand}(v^{cc}), \quad (11)$$
where “•” is the element-by-element product of the image plane. The activated $v^{cc}$ is expanded to the same dimension as $f^{sp}$ by an expanding operator $\text{Expand}(\cdot)$ to perform multiplication.

After both branches are performed, the small-object semantics $f^s \in \mathcal{R}^{C_d \times H_l \times W_l}$ that fully depict small objects are generated as follows.

$$f^s = \text{ReLU}(f^d_{1 \times 1} + f^d_{3 \times 3}),$$  \hspace{1cm} (12)

where “+” is element-by-element addition.

3.3. CS

The CS is designed to process the output $f^b \in \mathcal{R}^{C_b \times H \times W}$ of $LY_4$ in the backbone network. Here, we generate high-level semantics $f^h$ by extending the receptive field. This compels $f^h$ to further focus on large objects. The average pooling pyramid (APP) \[7\] is utilized to enlarge the receptive field. Unlike the strategy that directly enlarges the kernel size, APP does not increase the computational complexity \[7\]. As shown in Figure 8, the APP of the CS pools $f^b$ into three layers with different down-sampled resolutions: $11 \times 11 \approx 164 \times 164, 8 \times 8, \text{and } 5 \times 5$.

Figure 8. The pyramid structure of the contextual stream (CS). The input of CS is $f^b$ extracted by the $LY_4$ in the backbone network, and the high-level semantics $f^h$ are generated. APool, adaptive average pooling; Conv, convolution layer; DW Conv, depth-wise convolution layer; Up, up-sampling; Concat, concatenation; PAM, position attention module \[26\]; CAM, channel attention module \[26\].

Although the receptive field is enlarged, it is difficult to generate continuous contextual information that washes out heterogeneity in large-object features. Dual attention \[26\] is used to fully model long-range dependencies to address this issue. Dual attention, which includes a position attention module (PAM) and channel attention module (CAM), considers both spatial and channel information. The PAM computes the self-attention map for each position on the feature-image plane, and the CAM computes the self-attention map along the channels. They share similar procedures, which are detailed in \[26\], for producing these self-attention maps using matrix multiplication. Then, the input features are enhanced by these self-attention maps. Thus, the enhanced features produced by PAM and CAM contain continuous spatial dependency and channel dependency information, respectively.

In the $i$-th layer of Figure 8, $f^b \in \mathcal{R}^{C \times H/8 \times W/8}$ is averagely pooled as the down-sampled feature $f^{cs}_i \in \mathcal{R}^{C \times H_i \times W_i}$. Two reduced feature maps $f^{cs}_i \in \mathcal{R}^{C' \times H_i \times W_i}$ and $f^{cs}_i \in \mathcal{R}^{C' \times H_i \times W_i}$ are first generated by two convolutions with $1 \times 1$ kernels based on $f^{cs}_i$. Here, $C'$ is set to $C_i / 12$. $f^{cs}_i$ is processed by the PAM to obtain continuous spatial dependency information. Subsequently, a depth-wise convolutional layer with $3 \times 3$ kernels is used to generate the semantics $f^{sem}_i \in \mathcal{R}^{C' \times H_i \times W_i}$. $f^{cs}_i$ is processed by CAM to obtain dependence information...
along the channels. Again, a depth-wise convolutional layer with 3 × 3 kernels is used to generate the semantics $f_{i}^{sem} \in \mathcal{R}^{C_i \times H_i \times W_i}$.

After all of the layers are performed, a set of $f_{i}^{sem}$ and $f_{i}^{sem}$, $i = 1 \ldots 3$ are obtained. All $f_{i}^{sem}$ and $f_{i}^{sem}$, $i = 1 \ldots 3$ are up-sampled to the same resolution as $f_{i}^{s}$ and concatenated with $f_{i}^{s}$. Subsequently, the high-level semantics $f^{th} \in \mathcal{R}^{C^h \times \frac{H}{4} \times \frac{W}{4}}$ are obtained by a convolution with 1 × 1 kernels.

### 3.4. The Loss Function

As shown in Figure 3, the loss function $L$ is composed of two terms:

$$L = L^{main}(T, P^{f}) + L^{aux}(T, P^{a}),$$

where $L^{main}(T, P^{f})$ is the selected main loss which penalizes the errors in the final prediction, $P^{f}$. $L^{aux}(T, P^{a})$ is the auxiliary loss which penalizes the errors in the auxiliary prediction $P^{a}$ produced by the high-level semantics $f^{h}$. $P^{f}$ and $P^{a}$ are produced using similar prediction layers. First, the prediction layer convolves the input features using $3 \times 3$ kernels. Subsequently, batch normalization and ReLU activation are performed. Finally, the prediction is produced using a convolutional layer with $1 \times 1$ kernels. The differences between the prediction layers for $P^{f}$ and $P^{a}$ are only the parameters for the two inner convolutions. For $P^{f}$, we set the input and output numbers of the channels of the first convolution as $C^{f} + 2C^{b} + 2C^{s}$ and 128, where $C^{f}, C^{b}$ and $C^{s}$ are the number of channels for the small-object semantics $f_{1}^{s}$ and $f_{2}^{s}$ and the high-level semantics $f^{h}$, respectively. The input and output numbers of the channels of the remaining convolution are set to 128 and $K$, where $K$ is the number of classes. For $P^{a}$, the input and output numbers of the channels for the first convolution are set to $C^{b}$ and 64, respectively, and the input and output numbers of the channels for the remaining convolution are set to 64 and $K$.

We integrate a threshold $T_{r}^{L}$ with the standard cross-entropy loss to further highlight small objects without adding prior information about the data using $L^{main}(T, P^{f})$. For more detail, we compute the individual main loss $L^{main}(T, P^{f})[i]$ of each pixel in $P^{f}$ as follows:

$$L^{main}(T, P^{f})[i] = -P^{f}(i)[T(i)] + \log \left( \sum_{j=1}^{K} e^r_{P^{f}(i)[j]} \right),$$

where $T(i)$ is the ground-truth label of the $i$-th pixel and $K$ is the number of categories. $P^{f}(i)$ is the $i$-th vector in $P^{f}$ containing probabilistic values belonging to all categories of the $i$-th pixel. $P^{f}(i)[j]$ is the probability that the $i$-th pixel belongs to the $j$-th category. Then, an indicator function $\mathbb{I}$ is defined as

$$\mathbb{I}[i] = \begin{cases} 1, & L^{main}(T, P^{f})[i] \geq T_{r}^{L}, \\ 0, & L^{main}(T, P^{f})[i] < T_{r}^{L}. \end{cases}$$

where $\mathbb{I}[i]$ is the indicator of the $i$-th pixel in the input image $I$, and $T_{r}^{L} = 0.7$. $L^{main}(T, P^{f})$ is computed based on $\mathbb{I}$ and $L^{main}(T, P^{f})[i]$ as follows:

$$L^{main}(T, P^{f}) = \frac{\sum_{i=1}^{N} \mathbb{I}[i] \times L^{main}(T, P^{f})[i]}{\sum_{i=1}^{N} \mathbb{I}[i]},$$

where $N$ denotes the number of pixels in $I$. CNN-based models are advantageous for the recognition of large objects and elimination of heterogeneity in large objects. After several training iterations, the losses in the prediction of large objects will be smaller than those of small objects. Larger losses are more likely to occur on pixels in small objects. Therefore, apart from the RDM, $L^{main}(T, P^{f})$ further highlights the small objects during the backward procedure to some extent.
$L_{\text{aux}}(T, P^a)$ uses the standard cross-entropy loss function without changing the optimization preference for large (dominant) objects. Thus, $L_{\text{aux}}(T, P^a)$ favors the preference of $f^b$ for depicting large objects, which ensures the accuracy of large-object recognition. $L_{\text{aux}}(T, P^a)$ is formulated as

$$L_{\text{aux}}(T, P^a) = \frac{\sum_{i=1}^{N} -P^a(i)\log[T(i)] + \log(\sum_{j=1}^{K} e^{P^a(i)j})}{N}. \hspace{1cm} (17)$$

4. Datasets

In this paper, the UAVid and Aeroscapes datasets are employed to evaluate the performance in the segmentation of small objects. Both the two datasets contain objects having large-scale variations. It is notable that we do not employ the ISPRS Potsdam and Vaihingen datasets. As discussed in Section 1, only cars contained in the Potsdam and Vaihingen datasets are smaller. However, now, cars in UAV images are not small, and a large range of objects are much smaller than cars.

4.1. The UAVid Benchmark

The UAVid benchmark [4] is a newly published dataset of aerial imagery. Its detailed information is given on its official website (https://uavid.nl/ (accessed on 1 May 2020)). UAVid contains 8 classes, defined as “Clutter”, “Building”, “Road”, “Tree”, “Vegetation”, “Moving car”, “Static car”, and “Human”. UAVid focuses on the scenes near streets but consists of diverse landscapes including downtown areas (Figure 9a), villa areas (Figure 9b), and outskirts (Figure 9c). For the scale variations, to accurately measure the size, we calculate the average number of pixels for a single object in each category in Table 1. To simplify the notation, we use “Build.”, “Veg.”, “Mov.c.”, and “Stat.c.” to denote the Building, Vegetation, Moving car, and Static car classes. Definitely, the instances in the Human class have the smallest size (555.0 pixels) on average. More importantly, our objective is for the recognition of footpath-level objects. It is notable that here it is “footpath”, not “road” or “street”. Footpaths are much narrower than roads or streets. In UAVid, only the Human class contains the footpath-level objects. Therefore, we only consider that the Human class has such a small size. In addition, although cars are much larger than pedestrians, they are not as large as the other objects. Therefore, we consider that the Stat.c and Mov.c classes have medium sizes. In this way, the existing categories are divided into three groups, as shown in Table 1.

![Figure 9. The diverse landscapes in UAVid, (a) downtown area, (b) villa area, and (c) outskirts.](image)

<table>
<thead>
<tr>
<th>Clutter</th>
<th>Build.</th>
<th>Road</th>
<th>Tree</th>
<th>Veg.</th>
<th>Mov.c.</th>
<th>Stat.c.</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (pixels)</td>
<td>10,521.5</td>
<td>365,338.1</td>
<td>50,399.9</td>
<td>90,888.4</td>
<td>35,338.9</td>
<td>2586.5</td>
<td>4012.4</td>
</tr>
<tr>
<td>Group</td>
<td>large</td>
<td>large</td>
<td>large</td>
<td>large</td>
<td>medium</td>
<td>medium</td>
<td>small</td>
</tr>
</tbody>
</table>
4.2. The Aeroscapes Benchmark

The Aeroscapes [5] benchmark is another aerial dataset annotated for dense semantic segmentations. The details of Aeroscapes are shown on its official website (https://github.com/ishann/aeroscapes (accessed on 29 May 2020)). As shown in Figure 10, Aeroscapes contains a large range of scenes, including countryside, playgrounds, farmlands, roads, animal zoos, human settlements, and seascapes, among others. There are 12 classes that are defined as: “Background”, “Person”, “Bike”, “Car”, “Drone”, “Boat”, “Animal”, “Obstacle”, “Construction”, “Vegetation”, “Road”, and “Sky”. For brevity, we use “b.g.”, “Constr.”, and “Veg.” as the simplified notations for the “Background”, “Construction”, and “Vegetation” classes, respectively. Among these categories, the Person, Bike, Drone, Obstacle, and Animal classes contain footpath-level objects from a human perspective. As we all know, the UAV platform is oblique and the flight altitude is variable. For some specific scenes, the spatial resolutions of the images obtained by the UAV platform are really high. In this case, the footpath-level objects may not be small. To eliminate the larger ones caused by the UAV platform among the footpath-level objects, we calculate the average number of pixels for a single object in each category in Table 2. In Table 2, the oblique view and flight altitude do not enlarge the sizes of instances in the Person, Bike, Drone, and Obstacle classes. Therefore, we consider that the Person, Bike, Drone, and Obstacle classes have small sizes. However, the Animal class is influenced. The animal zoo scene is relatively smaller than the other scenes. When the UAV platform collects data, it must be nearer to the targets. This leads to really high spatial resolutions, and thus the animals in the image are not small (see Figure 10f). Therefore, we consider that the Animal class has medium sizes. In detail, the existing categories in Aeroscapes are divided into three groups, as shown Table 2.

![Figure 10. The diverse landscapes in Aeroscapes, (a) countryside, (b) playground, (c) farmland, (d) downtown area, (e) road, (f) animal zoo, (g) human settlement, and (h) seascape.](image)

<table>
<thead>
<tr>
<th></th>
<th>b.g.</th>
<th>Person</th>
<th>Bike</th>
<th>Car</th>
<th>Drone</th>
<th>Boat</th>
<th>Animal</th>
<th>Obstacle</th>
<th>Constr.</th>
<th>Veg.</th>
<th>Road</th>
<th>Sky</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (pixels)</td>
<td>84,770.38</td>
<td>2076.04</td>
<td>206.53</td>
<td>10,048.31</td>
<td>6154.82</td>
<td>18,984.38</td>
<td>2726.77</td>
<td>46,935.03</td>
<td>232,037.61</td>
<td>151,683.97</td>
<td>141,605.99</td>
<td></td>
</tr>
<tr>
<td>Group</td>
<td>large</td>
<td>small</td>
<td>small</td>
<td>medium</td>
<td>small</td>
<td>medium</td>
<td>small</td>
<td>large</td>
<td>large</td>
<td>large</td>
<td>large</td>
<td></td>
</tr>
</tbody>
</table>

5. Experiments

Firstly, we introduce the evaluation metrics used in this study and the implementation details of the training procedure. Subsequently, the experimental settings are discussed.
Then, the results of RDNet and the comparisons with existing methods for both the UAVid and Aeroscapes datasets are presented. Finally, we discuss some issues of interest.

5.1. Evaluation Metrics

To keep consistency with existing public evaluations, such as the studies by [4,5,14], we mainly use the IoU scores as the evaluation metrics. For the dataset having a range of classes with small/medium/large objects that pose a challenge to analyze the performance across scales, we divide the existing classes into small, medium, and large groups based on the scale information presented in Tables 1 and 2. Accordingly, the mean IoU score of the classes in each group is computed. Thus, three mean IoU scores $mIoU_s$, $mIoU_m$, and $mIoU_l$ are defined for the small, medium, and large groups, respectively. As these metrics are well-known and easily understood, we do not present them in any detail here.

5.2. Implementation Details

The proposed RDNet is implemented based on Pytorch. To train the network, we use stochastic gradient descent as the optimizer. The commonly used learning rate policy is employed, where the initial learning rate is multiplied by $(1 - \frac{iter}{max\_iter})^{power}$ for each iteration. Here, $power$ and the initial learning rate are set to 0.9 and $10^{-4}$, respectively. We train the RDNet on the training set for 10 epochs. In each epoch, we randomly selected 30,000 image patches with a batch size of three. Each patch is $704 \times 704$ in size. In more detail, to randomly select each image patch, we randomly generate two values $h^r$ and $w^r$ to indicate the top-left corner. Then, the image patch is copied from the original image based on the coordinates $(h^r, w^r)$ and $(h^r + 704, w^r + 704)$. Both the RDNet models for UAVid and Aeroscapes are trained using these settings. As is known, the training procedure usually trains the model iteratively. For each iteration, we actually obtain a trained model. As we focus on the small objects, we use the trained model that gives the highest accuracy scores for the small objects to obtain the final segmentation results. In terms of the comparison methods, we obtain the trained model for testing in the same way.

5.3. Experimental Settings Based on Validation

5.3.1. The Backbone Selection

Currently, there are a large range of general-purpose networks which can be used as the backbone. Our objective is to select a backbone which can make a good trade-off between computational complexity and accuracy. We list the integrations of RDNet with the representative general-purpose networks which have the real-time property in Table 3. MobileNetV3, GhostNet, STDC, EfficientNetV2, Xception, and ResNet18, which are commonly used CNN-based networks, can be integrated into RDNet in the way shown in Figure 3. MobileViT is a new general-purpose network that combines the strengths of CNNs and ViTs [29]. Although the architecture of MobileViT is different with the purely CNN-based networks, it can easily integrate with the RDNet framework. At the beginning of the MobileViT, a MobileNetV2 [34] block sequence is used to extract a set of low-level features. Subsequently, these low-level features are put into the ViT sequence to generate high-level semantics. We select the low-level feature maps with $\frac{1}{4}H \times \frac{1}{4}W$ and $\frac{1}{8}H \times \frac{1}{8}W$ pixels as the $f_1^l$ and $f_2^l$ and select the high-level semantics produced by the ViT sequence as the $f^h$. Then, the RDNet based on MobileViT can be constructed using the way shown in Figure 3.

In Table 3, the number of parameters (Pars.), computation complexity, GPU memory consumption (Mem.), and mIoU scores are presented. The computation complexity is measured by the float-point operations (FLOPs) and frames per second (FPS), which are computed using an image patch with $3 \times 1536 \times 1536$ pixels. The accuracy scores are obtained in the validation set of UAVid. Notably, the validation set is not used for training the model. RDNet(ResNet18), which is the RDNet version using ResNet18 as backbone, obtains the highest accuracy. RDNet(Xception) and RDNet(EfficientNetV2) obtain really close accuracy to RDNet(ResNet18), however, they cannot surpass RDNet(ResNet18). Xception
and EfficientNetV2 are constructed based on the depth-wise separable convolutions, which reduces many weights for cross-channel modeling. In contrast, ResNet18 fully models the cross-channel relationship using full convolutional operators. This suggests that the cross-channel modeling in the backbone is beneficial to RDNet. The RDNet(MobileViT) has the potential for accuracy improvements because it contains only 2.08M parameters. Unfortunately, the GPU memory consumption of RDNet(MobileViT) is really huge (28.5G), which limits its application. The RDNet(MobileNetV3) and RDNet(GhostNet) are lightweight and fast. However, they achieve much lower accuracy than the RDNet(ResNet18). From the comparison, the RDNet(ResNet18) obtains better trade-off among parameters, complexity, speed, GPU memory consumption, and accuracy. Therefore, we select the ResNet18 as the default backbone.

### Table 3. The selection of experimental settings based on validations.

<table>
<thead>
<tr>
<th>Methods (Backbone)</th>
<th>Characteristics</th>
<th>Pan.</th>
<th>FLOPs</th>
<th>FPS</th>
<th>mIoU</th>
<th>mIoUs</th>
<th>mIoUt</th>
<th>mIoUl</th>
<th>mIoUll</th>
<th>mIoUlll</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDNet(MobileNetV3)</td>
<td>RDM, DS, CS</td>
<td>1.4M</td>
<td>14.6G</td>
<td>54.1</td>
<td>2.4G</td>
<td>65.6</td>
<td>35.6</td>
<td>64.9</td>
<td>71.5</td>
<td></td>
</tr>
<tr>
<td>RDNet(GhostNet)</td>
<td>RDM, DS, CS</td>
<td>3.6M</td>
<td>13.4G</td>
<td>56.7</td>
<td>1.7G</td>
<td>62.3</td>
<td>25.8</td>
<td>60.8</td>
<td>70.2</td>
<td></td>
</tr>
<tr>
<td>RDNet(STDC)</td>
<td>RDM, DS, CS</td>
<td>10.8M</td>
<td>126.4G</td>
<td>32.9</td>
<td>2.5G</td>
<td>67.4</td>
<td>42.5</td>
<td>67.5</td>
<td>72.4</td>
<td></td>
</tr>
<tr>
<td>RDNet(EfficientNetV2)</td>
<td>RDM, DS, CS</td>
<td>21.4M</td>
<td>169.0C</td>
<td>14.9</td>
<td>6.1G</td>
<td>70.3</td>
<td>45.6</td>
<td>68.0</td>
<td>76.2</td>
<td></td>
</tr>
<tr>
<td>RDNet(Xception)</td>
<td>RDM, DS, CS</td>
<td>29.4M</td>
<td>298.6G</td>
<td>11.7</td>
<td>6.3G</td>
<td>71.7</td>
<td>50.4</td>
<td>71.2</td>
<td>76.2</td>
<td></td>
</tr>
<tr>
<td>RDNet(MobileViT)</td>
<td>RDM, DS, CS</td>
<td>2.08M</td>
<td>48.8G</td>
<td>6.7</td>
<td>28.5G</td>
<td>69.9</td>
<td>43.5</td>
<td>71.2</td>
<td>74.8</td>
<td></td>
</tr>
<tr>
<td>RDNet(ResNet18)</td>
<td>RDM, DS, CS</td>
<td>13.5M</td>
<td>130.6G</td>
<td>32.8</td>
<td>1.9G</td>
<td>72.4</td>
<td>50.4</td>
<td>72.5</td>
<td>76.5</td>
<td></td>
</tr>
<tr>
<td>BiSeNet(ResNet18)</td>
<td>Bilateral network</td>
<td>13.5M</td>
<td>147.9G</td>
<td>28.9</td>
<td>2.2G</td>
<td>63.0</td>
<td>27.0</td>
<td>57.7</td>
<td>72.2</td>
<td></td>
</tr>
<tr>
<td>ABCNet(ResNet18)</td>
<td>Self-attention, bilateral network</td>
<td>13.9M</td>
<td>144.1G</td>
<td>29.6</td>
<td>2.1G</td>
<td>67.8</td>
<td>37.8</td>
<td>66.9</td>
<td>74.7</td>
<td></td>
</tr>
<tr>
<td>SFNet(ResNet18)</td>
<td>Multi-level feature fusion</td>
<td>13.7M</td>
<td>206.4G</td>
<td>22.9</td>
<td>2.3G</td>
<td>67.1</td>
<td>41.3</td>
<td>63.6</td>
<td>73.9</td>
<td></td>
</tr>
<tr>
<td>BiSeNetV2(Nano)</td>
<td>Bilateral network</td>
<td>19.6M</td>
<td>194.4G</td>
<td>22.4</td>
<td>2.8G</td>
<td>69.3</td>
<td>41.9</td>
<td>69.1</td>
<td>74.9</td>
<td></td>
</tr>
<tr>
<td>GFFNet(ResNet18)</td>
<td>Multi-level feature fusion</td>
<td>20.4M</td>
<td>446.1G</td>
<td>10.2</td>
<td>3.0G</td>
<td>67.3</td>
<td>38.9</td>
<td>66.4</td>
<td>73.3</td>
<td></td>
</tr>
</tbody>
</table>

### 5.3.2. The Selection of Comparison Methods

As of now, there are hundreds of networks for semantic segmentation tasks, and we cannot compare all these methods with our RDNet. As our RDNet focuses on small objects, we test the existing small-object-oriented methods for which official codes have been released. They are BiSeNet, ABCNet, SFNet, BiSeNetV2, and GFFNet, which were introduced in Section 2.2. Notably, there exists a large range of backbone networks; we cannot test all the backbones for each method. Here, we use the backbone provided by each code of the official codes. Table 3 shows the properties and accuracies of these methods. We can find that most of the small-object-oriented methods have real-time property and prefer ResNet18. In addition, we select three representative networks from traditional methods (see Section 2.1). The DANet [26] is one of the self-attention-based methods, and the DeepLabV3 [21] is one of the methods using atrous convolutions and pyramids. MobileViT [30] is the representative method using vision transformer [29]. For the semantic segmentation task, MobileViT [30] uses the DeepLabV3 architecture and replaces the ResNet101.

In Table 3, DANet and DeepLabV3, can only achieve higher mIoU scores than BiSeNet when they use ResNet101. In most cases, they receive lower accuracy scores than the small-object-oriented methods, though they use much more complex backbones (ResNet101) as default. More importantly, based on the mIoU scores, the performance for segmenting small objects is more inferior. Although MobileViT achieves a relatively higher mIoU score of segmenting small objects than DANet and DeepLabV3, it still cannot surpass the existing small-object-oriented methods. Furthermore, the GPU memory consumption of MobileViT is huge (27.8G). Therefore, we pay attention mainly to the comparison between RDNet and small-object-oriented methods in more detail in the following. Meanwhile, to fully compare RDNet to the state of the art, we show the published results for each dataset.
5.4. Results on UAVid

This section is divided into two parts: the quantitative and visual results. The details of the UAVid dataset are presented in Section 4.1. Specifically, the human class has small sizes.

5.4.1. Quantitative Results

Table 4 shows the quantitative results for the UAVid test set. In Table 4, the comparisons are divided into two groups. The first group shows state-of-the-art results obtained by MSD [4], CAgNet [14], ABCNet [41], and BANet [31]. MSD, CAgNet, ABCNet, and BANet officially publish their results on UAVid. We directly quote their accuracy scores here. The second group shows the accuracy scores obtained by the small-object-oriented methods that are not included in state-of-the-art results. It shows the results obtained by BiSeNet [12], GFFNet [6], SFNet [33], and BiSeNetV2 [13]. We trained BiSeNet, GFFNet, SFNet, and BiSeNetV2 ourselves and used the official server (https://uavid.nl/, accessed on 1 May 2020) of UAVid to obtain the accuracy scores.

Table 4. Quantitative comparisons with the state of the art and the small-object-oriented methods on the test set for UAVid.

<table>
<thead>
<tr>
<th>Methods</th>
<th>mIoU(%)</th>
<th>mIoU*(%) (Small)</th>
<th>mIoU*(%) (Medium)</th>
<th>mIoU*(%) (Large)</th>
<th>Clutter (Large)</th>
<th>Build. (Large)</th>
<th>Road (Large)</th>
<th>Tree (Large)</th>
<th>Veg. (Large)</th>
<th>Mov.c. (Medium)</th>
<th>Stat.c. (Medium)</th>
<th>Human (Small)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSD(-) [4]</td>
<td>57.0</td>
<td>19.7</td>
<td>47.5</td>
<td>68.24</td>
<td>57.0</td>
<td>79.8</td>
<td>74.0</td>
<td>74.5</td>
<td>75.9</td>
<td>62.9</td>
<td>32.1</td>
<td>19.7</td>
</tr>
<tr>
<td>CAgNet(MobileNetV3) [14]</td>
<td>63.5</td>
<td>19.9</td>
<td>58.1</td>
<td>74.4</td>
<td>66.0</td>
<td>86.6</td>
<td>62.1</td>
<td>79.3</td>
<td>78.1</td>
<td>47.8</td>
<td>68.3</td>
<td>19.9</td>
</tr>
<tr>
<td>ABCNet(ResNet18) [41]</td>
<td>63.8</td>
<td>13.9</td>
<td>59.1</td>
<td>75.6</td>
<td>67.4</td>
<td>86.4</td>
<td>81.2</td>
<td>79.9</td>
<td>63.1</td>
<td>69.8</td>
<td>48.4</td>
<td>13.9</td>
</tr>
<tr>
<td>BANet(ResNet18) [31]</td>
<td>64.6</td>
<td>21.0</td>
<td>61.1</td>
<td>74.7</td>
<td>66.6</td>
<td>85.4</td>
<td>80.7</td>
<td>78.9</td>
<td>62.1</td>
<td>69.3</td>
<td>52.8</td>
<td>21.0</td>
</tr>
<tr>
<td>BiSeNet(ResNet18) [12]</td>
<td>61.5</td>
<td>17.5</td>
<td>56.0</td>
<td>73.4</td>
<td>64.7</td>
<td>85.7</td>
<td>61.1</td>
<td>78.3</td>
<td>77.3</td>
<td>48.6</td>
<td>63.4</td>
<td>17.5</td>
</tr>
<tr>
<td>GFFNet(ResNet18) [6]</td>
<td>61.8</td>
<td>24.4</td>
<td>53.7</td>
<td>72.6</td>
<td>62.1</td>
<td>83.4</td>
<td>78.6</td>
<td>78.0</td>
<td>60.7</td>
<td>70.7</td>
<td>36.6</td>
<td>24.4</td>
</tr>
<tr>
<td>SFNet(ResNet18) [33]</td>
<td>65.3</td>
<td>27.0</td>
<td>60.6</td>
<td>74.8</td>
<td>65.7</td>
<td>86.2</td>
<td>80.9</td>
<td>79.4</td>
<td>62.0</td>
<td>71.0</td>
<td>50.2</td>
<td>27.0</td>
</tr>
<tr>
<td>BiSeNetV2(-) [13]</td>
<td>65.9</td>
<td>28.3</td>
<td>62.1</td>
<td>75.1</td>
<td>66.6</td>
<td>86.3</td>
<td>80.8</td>
<td>79.5</td>
<td>62.3</td>
<td>72.5</td>
<td>51.6</td>
<td>28.3</td>
</tr>
<tr>
<td>RDNet(ResNet18) (ours)</td>
<td>68.2</td>
<td>32.8</td>
<td>66.0</td>
<td>76.2</td>
<td>68.5</td>
<td>87.6</td>
<td>81.2</td>
<td>80.1</td>
<td>63.7</td>
<td>73.0</td>
<td>58.9</td>
<td>32.8</td>
</tr>
</tbody>
</table>

In Table 4, RDNet obtains the highest mIoU score across these methods. More importantly, for each of the small, medium, and large groups, our RDNet obtains the highest accuracy (which is denoted by the mIoU*, mIoU* and mIoU scores). This means that the performance of segmenting large and medium objects does not become worse when our RDNet improves the accuracy of segmenting small objects. On the contrary, the accuracy scores (mIoU and mIoU) of segmenting the medium and large objects are further improved. For the segmentation of small objects, our RDNet achieves 4.5–18.9% higher mIoU scores than the existing methods. Notably, BiSeNetV2, GFFNet, and SFNet, which are powerful small-object-oriented methods, already have great performance in segmenting small objects, as noted in the introduction in Section 2.2. Therefore, our RDNet, which obtains higher mIoU than BiSeNetV2, GFFNet, and SFNet, has a stronger ability for segmenting small objects. Moreover, compared with state-of-the-art results, our RDNet achieves 11.8–18.9% higher mIoU scores, which is a significant improvement. Therefore, based on the results in Table 4, the superiority of our RDNet, especially for the segmentation of small objects, is validated.

5.4.2. Visual Results

To facilitate the visual comparison, we use the prediction maps of the validation set which has the ground-truth. Notably, the validation set is not used for training. Figure 11 shows the visual results for all classes in UAVid. Notably, we do not present the scenes that do not satisfy the scale-variation issue. Furthermore, SFNet and BiSeNetV2 achieve higher mIoU and mIoU scores than the other existing methods, as shown in Table 4; therefore, we only visually compare the results of RDNet with those of SFNet and BiSeNetV2 owing to the page limits. Based on the ground-truth (Figure 11, second column), RDNet provides more reliable results compared with other methods. The small objects extracted by RDNet are more complete, and the segmented large and medium objects are better.
In terms of small objects, we highlight the subareas that challenge the segmentation of small objects in the first and second images (Figure 11, first and second rows) using red rectangles. The sanitation worker in the first image and the man near a car in the second image are combined with a particularly complex background. The pedestrians in the second image have similar spectral features to the shadow of the buildings. Consequently, both SFNet and BiSeNet-v2 miss these small targets in the results, whereas RDNet identifies all of them (Figure 11, last column). In terms of large objects, we highlight the subareas that challenge the segmentation of large objects in the second and third images (Figure 11, second and last rows) using black rectangles. In the second image, the subarea contains trees, vegetation, and roads. These are different types of large objects shading each other, which results in difficulty of recognition. Neither SFNet nor BiSeNet-v2 segments this subarea well, whereas RDNet succeeds. In the last image, the subarea contains a building surrounded by vegetation. BiSeNet-v2 and RDNet misclassify the building, whereas RDNet produces a perfect result.

Figure 11. The visual comparisons for the UAVid dataset.

The visual results show that RDNet obtains superior results for both the small and large objects in UAVid. The accuracy of segmenting large objects is not reduced when RDNet highlights the small objects. On the contrary, the accuracy of segmenting large objects is further improved.

5.5. Results on Aeroscapes

The structure of the results on the Aeroscapes dataset is similar to Section 5.4. The details of the Aeroscapes dataset are presented in Section 4.2. Specifically, the person, bike, drone, and obstacle classes have small sizes.
5.5.1. Quantitative Results

We compare the results of RDNet with the state-of-the-art results specified in Aeroscapes and those obtained using BiSeNet, ABCNet, GFFNet, SFNet, and BiSeNetV2. The accuracy values are presented in Table 5. In terms of the state-of-the-art results, that is, EKT-Ensemble [5], specific issues must be recognized. In addition to the precise mIoU scores, the evaluation for IoU scores is only provided in a bar chart with an accuracy axis (see details in [5]). We cite this evaluation by carefully determining the values along the accuracy axis. Therefore, these values are considered approximate (≈) in Table 5. [5] assembled different datasets, including PASCAL [52], CityScapes [53], ADE20k [54], and ImageNet [55] to improve accuracy, owing to the large range of scenes contained in Aeroscapes. The results using this assembly, denoted as EKT-Ensemble, achieve a 57.1% mIoU score.

Table 5. Quantitative comparisons with the state of the art and the small-object-oriented methods for Aeroscapes.

<table>
<thead>
<tr>
<th>Methods</th>
<th>mIoU (%)</th>
<th>mIoU_s (%) (Small)</th>
<th>mIoU_m (%) (Medium)</th>
<th>mIoU_l (%) (Large)</th>
<th>b.g. (Large)</th>
<th>Person (Small)</th>
<th>Bike (Small)</th>
<th>Car (Medium)</th>
<th>Drone (Small)</th>
<th>Boat (Medium)</th>
<th>Animal (Medium)</th>
<th>Obstacle (Small)</th>
<th>Conste (Large)</th>
<th>Veg. (Large)</th>
<th>Read (Large)</th>
<th>Sky (Large)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKT-Ensemble(-) [5]</td>
<td>57.1</td>
<td>≈30.5</td>
<td>≈52.7</td>
<td>≈83.8</td>
<td>≈56.0</td>
<td>≈47.0</td>
<td>≈15.0</td>
<td>≈69.0</td>
<td>≈46.0</td>
<td>≈51.0</td>
<td>≈38.0</td>
<td>≈14.0</td>
<td>≈70.0</td>
<td>≈93.0</td>
<td>≈86.0</td>
<td>≈94.0</td>
</tr>
<tr>
<td>BiSeNet(ResNet18) [12]</td>
<td>60.1</td>
<td>35.4</td>
<td>58.4</td>
<td>80.9</td>
<td>74.4</td>
<td>46.1</td>
<td>24.8</td>
<td>86.3</td>
<td>58.9</td>
<td>49.3</td>
<td>39.4</td>
<td>11.7</td>
<td>57.2</td>
<td>93.5</td>
<td>84.6</td>
<td>95.1</td>
</tr>
<tr>
<td>ABCNet(ResNet18) [41]</td>
<td>62.2</td>
<td>33.3</td>
<td>63.3</td>
<td>83.5</td>
<td>77.8</td>
<td>44.3</td>
<td>23.3</td>
<td>82.8</td>
<td>54.1</td>
<td>72.2</td>
<td>41.1</td>
<td>11.3</td>
<td>65.2</td>
<td>93.7</td>
<td>89.3</td>
<td>91.3</td>
</tr>
<tr>
<td>GFFNet(ResNet18) [5]</td>
<td>62.3</td>
<td>36.7</td>
<td>63.2</td>
<td>82.2</td>
<td>75.6</td>
<td>45.8</td>
<td>27.4</td>
<td>84.6</td>
<td>59.2</td>
<td>63.5</td>
<td>41.5</td>
<td>14.4</td>
<td>59.6</td>
<td>93.4</td>
<td>87.6</td>
<td>94.8</td>
</tr>
<tr>
<td>SFNet(ResNet18) [13]</td>
<td>63.7</td>
<td>37.7</td>
<td>65.0</td>
<td>83.7</td>
<td>76.8</td>
<td>49.4</td>
<td>25.8</td>
<td>84.4</td>
<td>57.3</td>
<td>66.0</td>
<td>44.4</td>
<td>18.2</td>
<td>65.4</td>
<td>93.7</td>
<td>88.9</td>
<td>93.9</td>
</tr>
<tr>
<td>RDNet(ResNet18)(ours)</td>
<td>66.7</td>
<td>42.2</td>
<td>70.6</td>
<td>83.9</td>
<td>79.8</td>
<td>54.4</td>
<td>38.7</td>
<td>85.3</td>
<td>60.0</td>
<td>78.8</td>
<td>47.7</td>
<td>15.8</td>
<td>62.9</td>
<td>94.1</td>
<td>88.8</td>
<td>93.6</td>
</tr>
</tbody>
</table>

As shown in Table 5, RDNet obtains the highest mIoU score among all the methods. More importantly, the mIoU_s score for segmenting small objects, mIoU_m score for segmenting medium objects, and mIoU_l score for segmenting large objects obtained by RDNet are all higher than those obtained by the other methods. This validates that RDNet further improves accuracy for the segmentation of large objects when it improves the accuracy of segmenting small objects. Specifically, the mIoU_s score is 3.9% higher than that of BiSeNetV2, which ranks the second. For the bike class with the smallest objects, our RDNet obtains at least a 7.8% higher IoU score than the other methods. For the person class with small objects, RDNet obtains at least 5.0% higher accuracy than the other methods. SFNet obtains a higher IoU score for segmenting the obstacle class with small objects. This validates that SFNet has great power for segmenting small objects. However, compared with our RDNet, SFNet obtains lower IoU scores for segmenting the other classes with small objects, including the person, bike, and drone classes. This validates the superiority of RDNet.

5.5.2. Visual Results

Aeroscapes contains numerous scenes with large-scale variations between the classes. Here, we select four images that cover the classes with small objects and the objects challenging the segmentation in Figure 12. From these visual results, we observe some mistakes in each of the predictions, which demonstrates the challenges of the Aeroscapes dataset. RDNet obtains reasonable segmentation results, in which all objects are recognized. Compared with SFNet and BiSeNet-v2, RDNet obtains superior results with less misclassification in each image. Specifically, the first row of Figure 12 presents the smallest object, a bike. The result obtained by RDNet are much better than those obtained by SFNet and BiSeNetV2. In the second row of Figure 12, RDNet obtains superior results for the segmentation of the person and drone classes. In Table 5, SFNet obtains a higher IoU score for segmenting obstacles than RDNet. However, in the visual results, RDNet can achieve better segmentation results in a range of scenes, such as the third row in Figure 12. In the last row of Figure 12, we show the segmentation results of the animal class. RDNet obtains the best results for segmenting animals in the image.
6. Discussion

6.1. Ablation Study

In RDNet, we propose three modules, including RDM, DS, and CS. RDM has two alignment branches: cosine alignment \( \text{RDM}^C \) and neural alignment \( \text{RDM}^N \). Here, we analyze the influence of each of these factors. We compare the results obtained by RDNet without each component with those of the complete RDNet to directly demonstrate the effects. We use RDNet-RDM as the simplified form of the RDNet without the RDM. The other variants of RDNet are denoted in similar forms.

Table 6 shows the ablation study using the Aeroscapes dataset. RDNet-RDM shows the lowest accuracy. Compared with RDNet, the mIoU score decreases by more than 6%. Moreover, the mIoU score of RDNet-RDM is only 31.9% which is considerably lower than those of the RDNet and other variants. This demonstrates that RDM has a significant influence on improving accuracy scores of segmenting small objects. DS has positive roles for segmenting small and medium objects based on the changes in the mIoU and mIoU scores. CS generates high-level semantics which primarily depict large objects. From Table 6, CS satisfies the objective. When the CS is removed, the high-level semantics become \( f_4 \) produced by the deepest layer \( (LY_4) \) in ResNet18, and the mIoU score for the segmentation of large objects decreases. \( \text{RDM}^C \) and \( \text{RDM}^N \) are the inner parts of RDM. Based on the mIoU scores, each of these has a positive influence on the segmentation of small objects. After they are combined as RDM, the accuracy of the segmentation of small objects increases significantly.

Figure 12. Visual comparisons for the Aeroscapes dataset.
Table 6. The ablation study for each module in RDNet.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pars.</th>
<th>FLOPs</th>
<th>mIoU (%)</th>
<th>mIoU* (%)</th>
<th>mIoU** (%)</th>
<th>mIoU*** (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDNet - CS - DS (ResNet18 + RDM)</td>
<td>12.6M</td>
<td>124.4G</td>
<td>63.3</td>
<td>40.9</td>
<td>68.2</td>
<td>78.2</td>
</tr>
<tr>
<td>RDNet - CS</td>
<td>12.8M</td>
<td>129.9G</td>
<td>64.0</td>
<td>42.0</td>
<td>69.7</td>
<td>78.2</td>
</tr>
<tr>
<td>RDNet - DS</td>
<td>13.6M</td>
<td>125.1G</td>
<td>64.6</td>
<td>40.6</td>
<td>68.2</td>
<td>81.6</td>
</tr>
<tr>
<td>RDNet - RDM</td>
<td>13.8M</td>
<td>130.5G</td>
<td>62.7</td>
<td>37.5</td>
<td>66.0</td>
<td>81.0</td>
</tr>
<tr>
<td>RDNet - RDMν</td>
<td>13.7M</td>
<td>130.4G</td>
<td>63.8</td>
<td>38.6</td>
<td>68.2</td>
<td>81.3</td>
</tr>
<tr>
<td>RDNet</td>
<td>13.8M</td>
<td>130.6G</td>
<td>66.7</td>
<td>42.2</td>
<td>70.6</td>
<td>83.9</td>
</tr>
</tbody>
</table>

When CS and DS are both removed from RDNet, the network only contains ResNet18 and RDM, which is shown in the second row in Table 6. The high-level semantics $f^h$ in ResNet18+RDM are the output of $L_Y^4$ in ResNet18 directly. ResNet18+RDM is really simple, its FLOPs score is 70G less than that of BiSeNetV2. The mIoU* score of segmenting small objects obtained by ResNet18+RDM can meet 40.7%, which is 2.4% higher than that obtained by BiSeNetV2. As shown in Table 5, BiSeNetV2 obtains a mIoU* score of segmenting small objects 3.9% lower than our RDNet, however, it obtains a mIoU* score higher than the other existing methods. In other words, just using RDM and the simple backbone ResNet18, a higher accuracy of segmenting small objects than other existing methods can be obtained. This validates the great power of RDM for segmenting small objects again.

6.2. The Output of Each Module in RDNet

As shown in Figure 2, the high-level semantics extracted by deep layers primarily depict large objects. The low-level features are a mixture of the large and small objects. Although small-object information is recorded in the low-level features, large-object information plays a dominant role. This is why the existing deep-learning-based methods obtain high accuracy for segmenting large objects, while achieving low accuracy for segmenting small objects (see Table 4).

To fully analyze the positive influences of each module in RDNet, we present the saliency map of the features produced by each module in Figure 13. The features $f^1_2$ and $f^2_3$ generated by the $L_Y^1$ and $L_Y^2$ in ResNet18 have the same properties as the low-level features in Figure 2. The high-level semantics $f^h$ extracted by CS and $f^h_3$ generated by the deep layer $L_Y^4$ in ResNet18 have the same properties as the high-level semantics presented in Figure 2. However, from the comparison between $f^h_1$ and $f^h_3$ in Figure 13, the contextual information in $f^h$ is more homogeneous, which facilitates the representation of large objects. This validates the positive effect of the CS from a visual perspective.

In Figure 13, the difference features $f^d_1$ and $f^d_2$ produced by the proposed RDM fully highlight small objects and exclude the predominant effect of large objects. This proves that RDM meets the objective of changing the predominant relationship between large and small objects. However, $f^d_1$ and $f^d_2$ only locate the small objects; they cannot depict the small objects well. A DS is proposed to generate more semantics to address this issue. After the DS is performed using $f^d_1$ and $f^d_2$ as its input, the generated small-object semantics $f^s$ contain the necessary semantics for depicting more properties of a small object. Coupled with the quantitative analysis in Table 6, we can conclude that RDM, DS, and CS are essential for RDNet, specifically RDM, which plays a vital role in improving the representation ability of small objects.

6.3. The Usage of Complex Backbones

In Section 5.3.1, we select ResNet18 to make a good trade-off between computational complexity and accuracy. Here, we give a discussion for the usage of more complex backbones via the segmentation in Aeroscopes. In Table 7, ResNet50 and ResNet101 are integrated. For comparison, the small-object-oriented methods that use backbone networks are tested. In terms of the small-object-oriented methods, including BiSeNet, ABCNet, GFFNet, and SFNet, the complex backbones cannot improve the mIoU* score.
for segmenting small objects. However, the \( mIoU^m \) and \( mIoU^l \) scores improve compared with those in Table 5 to some extent. The main reason is that they do not change the predominant effects of large objects. For our RDNet, the complex backbones improve not only the \( mIoU^m \) and \( mIoU^l \) scores for segmenting medium and large objects but also the \( mIoU^s \) score for segmenting small objects.

In Table 7, RDNet obtains the highest \( mIoU^s \), \( mIoU^m \), and \( mIoU^l \) scores. Specifically, when ResNet50 is used, RDNet achieves a 7.4% higher \( mIoU^s \) score for segmenting small objects than SFNet, which ranks second. For the usage of ResNet101, RDNet obtains a 7.9% higher \( mIoU^s \) score than SFNet. To conclude, the usage of complex backbones cannot alter the superiority of RDNet.

Table 7. The comparison using complex backbones.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Backbones</th>
<th>( mIoU )</th>
<th>( mIoU^s )</th>
<th>( mIoU^m )</th>
<th>( mIoU^l )</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiSeNet [12]</td>
<td>ResNet50</td>
<td>61.8</td>
<td>35.3</td>
<td>60.5</td>
<td>83.7</td>
</tr>
<tr>
<td>ABCNet [41]</td>
<td>ResNet50</td>
<td>62.8</td>
<td>30.3</td>
<td>66.4</td>
<td>86.6</td>
</tr>
<tr>
<td>GFFNet [6]</td>
<td>ResNet50</td>
<td>63.1</td>
<td>35.5</td>
<td>64.2</td>
<td>84.5</td>
</tr>
<tr>
<td>SFNet [33]</td>
<td>ResNet50</td>
<td>64.5</td>
<td>35.9</td>
<td>66.6</td>
<td>86.0</td>
</tr>
<tr>
<td>RDNet</td>
<td>ResNet50</td>
<td>67.4</td>
<td>43.3</td>
<td>67.6</td>
<td>86.7</td>
</tr>
<tr>
<td>BiSeNet [12]</td>
<td>ResNet101</td>
<td>62.4</td>
<td>35.3</td>
<td>61.1</td>
<td>84.8</td>
</tr>
<tr>
<td>ABCNet [41]</td>
<td>ResNet101</td>
<td>63.2</td>
<td>30.1</td>
<td>66.8</td>
<td>87.4</td>
</tr>
<tr>
<td>GFFNet [6]</td>
<td>ResNet101</td>
<td>63.4</td>
<td>35.4</td>
<td>66.2</td>
<td>86.5</td>
</tr>
<tr>
<td>SFNet [33]</td>
<td>ResNet101</td>
<td>65.1</td>
<td>36.1</td>
<td>66.5</td>
<td>86.8</td>
</tr>
<tr>
<td>RDNet</td>
<td>ResNet101</td>
<td>68.5</td>
<td>44.0</td>
<td>69.7</td>
<td>87.4</td>
</tr>
</tbody>
</table>

7. Conclusions

In this study, a novel semantic segmentation network called RDNet is proposed for aerial imagery. In RDNet, RDM is first proposed to highlight small objects. RDM develops a reverse difference concept and aligns the semantics for both high-level and low-level features. Consequently, RDM eliminates the predominant effect of large objects in the low-level features. Then, the DS, which models the spatial and cross-channel correlations, is proposed to generate small-object semantics using the output of RDM. Additionally, the CS is designed using an average pooling pyramid and dual attention to generate high-level semantics. Finally, the small-object semantics bearing the small objects and high-level semantics that focus on the large objects are combined to make a prediction.

Two aerial datasets are used to fully analyze the performance of RDNet: UAVid and Aeroscapes. Based on the experimental results, RDNet obtains superior results compared with the existing methods for both datasets. The accuracy scores of segmenting small, medium, and large objects are all improved. More importantly, the accuracy im-
provement for segmenting small objects is prominent. According to Table 3, RDNet has less computational complexity and uses less GPU memory compared with the existing small-object-oriented methods. This shows that RDNet achieves superior results using less computing resources. The ablation study demonstrates that the proposed RDM plays a vital role in the accuracy improvement of the segmentation of small objects. The DS further enhances the output of RDM, and the CS ensures good performance in segmenting large objects. Meanwhile, based on the visualization of the output, the positive effect of each module in RDNet is vividly shown.

In the future, the resolution of remote sensing images will be further improved, though the current resolution is so high that footpath-level objects are recorded well. Small object recognition will become more important. RDNet architecture, which highlights small objects for deep-learning-based models, will be used for more applications in the field of remote sensing.

Author Contributions: Methodology, software, validation, writing—original draft preparation, Huan Ni; supervision, resources, investigation, writing—review and editing, Jocelyn Chanussot; methodology, investigation, resources, funding acquisition, Xianan Niu; investigation, writing—review and editing, Hong Tang and Haiyan Guan. All authors have read and agreed to the published version of the manuscript.

Funding: National Natural Science Foundation of China (Grant No. 41901310, 41971414, and 41801384).

Data Availability Statement: Not applicable.

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

References


