remote sensing

Article

A Novel Lightweight Semantic Segmentation Network in Remote Sensing

Fuxiao Liu[1], Ming Wu[2]

[1] Key Laboratory of Pattern Recognition and Intelligence System, Beijing University of Posts and Telecommunications, Beijing 100190, China ; liufuxiao@bupt.edu.cn

[2] Key Laboratory of Pattern Recognition and Intelligence System, Beijing University of Posts and Telecommunications, Beijing 100190, China ; wuming@bupt.edu.cn

Abstract: Achieving high quality of semantic segmentation on embedded systems has become a new trend in the field of computer vision. To achieve that, we propose a new lightweight semantic segmentation framework called Depthwise Separation Fully Convolutional Network (DSFCN). The new network uses pointwise group and depthwise convolutions with shortcut blocks to decrease memory and computation cost greatly. We compare DSFCN with competing methods on two datasets of remote sensing imagery: Vaihingen and Potsdam. The results indicate our new architecture achieves Fully Convolutional Network (FCN) level accuracy, while costing much less memory and computation budget.

Keywords: Semantic Segmentation; Remote Sensing; Deep Learning; Lightweight.

1. Introduction

Recently, semantic image segmentation is a very active research direction in computer vision. It combines the traditional image segmentation and object detection [1]. Its purpose is to segment the image into several groups of pixel regions with specific semantic meaning, and identify the class of each region with pixel semantic label. Because of the fact that it has a strong application in remote sensing [4], autonomous driving, medical imaging analysis, robotics and so on [2], it has been a preoccupying topic in the last few years. Before deep learning dominated computer vision, we used machine learning algorithms [6]: TextonForest [7], Boosting [8], or Support Vector Machine [11] for semantic segmentation. However, due to the increasing need of more abstract and powerful features for object detection and classification of images, and thanks to the fast development of computer computational capacities and the appearance of large-scale labeled image, Deep Convolutional Neural NetWork (DCNN) [10] can make a new breakthrough in the field of image recognition. By comparing with the traditional hand-crafted method, DCNN can automatically extract discriminative contextual features at different scales by the organization of multi-layer neurons. From then on, a series of classic DCNN models were proposed including LeNet [13], AlexNet [12], VGGNet [14], ResNet [15], the results on the ImageNet classification benchmark improved a lot.

1.1 DCNN in remote sensing imagery

Initially, the main deep learning approaches for very-high-resolution remote sensing imagery was patch classifications, proposed by Mnih [16] in 2013. However, because of the fact that overlapped patches may cause much redundant computations, the averaging process can readily lose useful edge information. To solve the problem, one of the most state-of-the-art frameworks in semantic segmentation, the Fully Convolutional Network(FCN) was proposed by Long [17], which can be trained end-to-end and produce segmentation output correspondingly-sized as the input image. From then on, several FCN-based architectures were devised and achieved better accuracy for remote sensing images. Chen [18] introduced two new models called Symmetrical Normal-Shortcut FCN(SNFCN) and Symmetrical Dense-Shortcut FCN(SDFCN) modified from standard FCN meanwhile added a creative post processing named "overlay policy". In order to preserve the resolution of output results, Sherrah presented a deep FCN-Conditional Random Field(FCN_CRF) model [19] with no transpose convolutional and downsampling layers. In addition, Audebert [20] introduced a multikernel and multiscale hybrid FCN. Thus although these modified FCN achieved state-of-art accuracy, they suffered from the same problem such as the huge size of the network and the long time consumed for computational cost.

1.2 FCN for Semantic Segmentation

One of the most important breakthroughs in semantic segmentation for dense predictions is Fully Convolutional Networks(FCN) by Long, which allows segmentation images of any size to be generated without fully connected layers. It achieved much efficiently by comparing with the patch classification approach. Additionally, the upsample layers implement deconvolutions by learning instead of simple bilinear interpolations [22]. Then to overcome the drawback of pooling layers, FCN also introduced skip layers to combine multi-scale features: FCN-8s, FCN-16s and FCN-32s. Apart from these, another two approaches are presented to address the issue.

First one is Encoder-Decoder model like U-Net [23], proposed by Ronneberger et al. Encoders decrease the spatial dimension with pooling layers, meanwhile decoders can recover object details better with the help of shortcut connections between downsample and upsample layers. The second approach we use is called dilated convolutions [24] which is the substitution for pooling layers. This technique not only increases the receptive field but keeps the location information as well.

To tackle the lack of convolutional layers that can't perform well around objects' boundaries, CRF was introduced by Chen et al. [25] as a post-processing method which can "smooth" segmentation according to the observation that similar intensity pixels tend to be labeled as the same class. Consequently, a generic framework was established: Original map to FCN to CRF to Segmentation images.

1.3 Development of Lightweight Models

However, most of the models suffer from redundant computations and size, which limit their use on normal PC and mobile apps. Additionally, due to research finding that the deeper network results in both better performance but slower computation, it's difficult to tradeoff between performance and cost.

With the increasing need for better performance in DCNN, the layers of the network got deeper and deeper. As a result, its low efficiency is still far from real-time execution in normal PC and mobile apps. To overcome the problem, the traditional approach is Model Compression including parameter pruning[27], low-rank factorization[28], compact convolutional filters[29] and knowledge distillation[30]. By compressing trained models, this technique is able to address the memory issue.

Different from the idea of changing the trained models, the unique design of lightweight framework is to establish a more efficient "Computing Network" mainly for convolutional layers. The first remarkable lightweight model: SqueezeNet [31], which achieved AlexNet-level accuracy with 50 times fewer parameters. Its main novel contributions are replacing 3x3 filters with 1x1 ones and reducing the number of channels to 3x3 filters. In 2017, a new efficient network for mobile vision applications: MobileNet [32], which was introduced by Google. This technique utilizes depthwise separable convolutions to design lightweight streamlined models. However, lacking information flow is a crucial drawback of depthwise convolutions which means that it blocks information flow between channel groups and weakens representation.

In order to overcome this issue, an extremely efficient framework: ShuffleNet [33], established by Face++, which can improve the information flow much efficiently. According to its creative ideas: pointwise group convolution and "channel shuffle", ShuffleNet works especially well on images with more channels. Moreover, the superior performance on ImageNet classification and MSCOCO object detection demonstrated its state-of-art property. As a result, the lightweight model is an excellent approach to achieve efficiency.

1.4 the main aim of our work

In this paper, we develop a new cost-efficient architecture called DSFCN for remote sensing imagery, which is a variant of FCN, to address the expensive cost in remote sensing imagery. Moreover, we designed a series of comparative experiments with SegNet [21], Mobile_FCN and ShuffleNet_FCN and trained them on ISPRS semantic segmentation dataset without using Digital Surface Model DSM. In conclusion, the novel contributions of our work are:

- We introduce DSFCN framework by using pointwise group and depthwise convolutions with shortcut blocks in Encoder-Decoder style, which tradeoff between performance and cost.
- Our framework equipped with residual blocks produces better performance than other lightweight models
- Our DSFCN has the quality of generalization which achieves state-of-the-art results on both Vaihingen and Potsdam dataset.

2. Methods

In this section, we first introduce our design strategies for CNN framework which have low cost while maintaining competitive accuracy. Then the basic blocks of our architecture are presented. Finally, we describe the Encoder-Decoder and streamlined structure.

2.1Design Strategies

2.1.1 Replace Standard Convolutions with New Depthwise Separable Convolution Blocks

Given the expensive budget of a large number of standard convolutions, we finally choose pointwise group and depthwise convolutions after comparison with other structure units, which will be shown in Section 4 in detail. A convolution block can be an operation which converts the input with three dimensions into downsampled 2-D representation. As we can see in Fig. 1(a), the standard convolution has the effect of filtering and then combining inputs to produce a new feature map.

We will calculate the computations and sizes of different structures. Let's take a feature map with M × Win × Hin as input and results in a N × Wout × Hout feature map, where Win and Hin are spatial width and height of the input, M is the input depth, Wout and Hout are spatial width and height of the output, N is the output depth. The convolution kernel is of size K × K × Hin×Hout where K is supposed as the spatial dimension of the kernel. So, the computational cost of standard convolution is K2MNWinHin *FLOPS*, which is floating-point operations per second. The memory cost is (K2M+1)N *Bytes*.The multiplying Relationship between these parameters produces huge budgets. To address such problems, the depthwise separable convolutions which factorize the standard one into a depthwise convolution and a 1 ×1 convolution named pointwise convolution was proposed as the basic block of MobileNet. We can see it in Fig. 1(b). This technique applies a single filter for every input channel as a depthwise convolution operation. In spite that this operation is extremely more efficient, it also breaks the interaction between different channels. So, the pointwise convolutions combine them to generate new 2-D features. Depth Wise Separable convolutions have the computational cost of: (K2MWinHin+MNWinHin) *FLOPS*. The memory cost: (K2+1)M+(M+1)N *Bytes*. If we assume that K=3 and the number of channels is 100(Most of the channels in our framework are over 100), the depthwise separable convolutions cost 9 times less computations and parameters than standard convolutions while the overall accuracy is just at a small reduction.

In order to overcome the expensive pointwise convolution, a straightforward solution is pointwise group convolution, which executes on its corresponding input channel group only. In addition, as we can see in Fig. 1(c), "channel shuffle" can divide the channels of input into many subgroups from each group, and then "shuffle" the groups to make them with different subgroups in next layers. Another state-of-art approach to reduce computation cost is to employ the channel concatenation instead of element-wise addition when the channel dimension is increased. This technique aligns with the idea of residual block.

If we assume g as the number of groups and set the bottleneck channels to 1/4 of output channels, the computation cost is: (M2/g+K2M/2)WinHin/2 *FLOPS*. The memory cost: (M/g+3+K2/2)M/2 *Bytes*. By comparison with Fig. 1(b), it can achieve over 4 times less computation and size when g=1. So given the less budget, this new block can use wider feature maps to improve accuracy.

So in consideration of state-of-art performance of the new depthwise separable convolution, we utilize pointwise group convolutions and depthwise convolutions as main components of our new basic block, shown in Fig. 2.

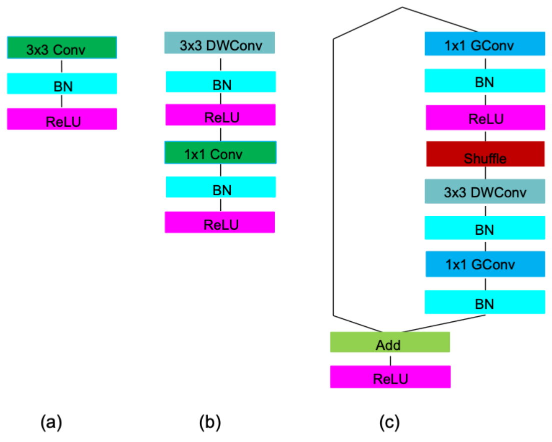

Figure 1. (a) Standard unit. **(b)** MobileNet unit. **(c)** ShuffleNet unit.

2.1.2 Decrease Scale Factor and Group Number

As our new network is customized from ShuffleNet, introduced by Zhang, we learnt the relationship between scale factor and group number from his paper[33]. It's obvious from Table 2 in [33] that for given a complexity budget, the network with larger g produces better overall accuracy. After further analysis, we concluded that given a constant Stage depth, the framework with the smaller g achieves much more remarkable accuracy.

So, in Section 4, we adopt this strategy: decrease Stage depth and g, to customize our DSFCN which achieves SegNet level accuracy while less complexity than other lightweight models.

2.1.3 Construct Symmetrical FCN-based Style

Like most FCN-based models, our DSFCN is an Encoder-Decoder style. Different from FCN-8s and SegNet whose encoders are transplanted from per-trained VGG16 network, our model adept modified ShuffleNet as the encoder part because the pre-trained models in a huge size are difficult to train. Apart from this, we replace the common convolution blocks with our new blocks: Light

Unit(Fig. 2) which is able to accelerate the training process and reduces the model size greatly.

As for the decoder part, different from FCN-8s, SegNet employs the upsampling layer with the principle of pooling-indices shared from downsampling stages symmetrically. However, this technique isn't learning during the training process. To tackle this issue, the decoders in our model are stacked by four transpose-convolutional layers. Moreover, we adopt three additional identity mapping shortcut connections between downsampling and upsampling layers symmetrically. As a result, the output of decoders is influenced by itself and the features extracted from the corresponding encoder.

In our experiments (shown in Section 3), a set of designed comparative experiments will demonstrate the state-of-art performance of our network.

2.2 Basic Shortcut Convolutional Block

The basic block in our DSFCN is a residual block called Light Unit, which is modified from ShuffleNet unit. After some experiments, we found when set g as 1 and channels of Stage2 as 100, our proposed framework performs best, which is relatively high accuracy and small size, on ISPRS datasets of remote sensing imagery. That is because the class number of our data is just six, so we can use a shallow network structure. Consequently, our proposed shortcut unit is shown in Fig. 2.

In Fig. 2(a), the Light Unit has two branches. The main branch of the bottleneck block is started by the 1 × 1 pointwise group convolutions. Then we apply the depthwise convolution with a 3 × 3 window size in the bottleneck feature map. In order to recover the bottleneck channels which is 1/4 of the output ones, we add the second pointwise group convolution. Additionally, for the purpose to solve the covariate shift problem, we employ batch normalization after each convolution layer.

$$BN(x) = \max\left(\frac{x - E(x)}{\sqrt{Var(x) + \alpha}} \bullet \beta + \gamma, 0\right) \tag{1}$$

Where Var(x) and E(x) are variance and mean of the input x, β is the scale parameter of the batch data, α is a positive small value. As a result, this technique can accelerate the training process a lot.

Apart from this, we also add ReLU activation layer [37] after the first pointwise group convolution. So, for Fig. 2(a), the block can be written as follows:

$$F_{light}(x) = Dropout(\mathrm{Re}LU(F_{main}(x) + x)) \tag{2}$$

As for Fig. 2(b), which is the case that input layer downsamples with stride 2 instead of 1, then we add average pooling on the fine branch.

$$F_{\mathrm{light}}(x) = Dropout(\mathrm{Re}LU(F_{main}(x) + F_{fine}(x))) \tag{3}$$

where, we apply channel concatenation instead of addition, and

$$F_{fine}(x) = Pool(x) \tag{4}$$

Finally, we add a dropout, proposed by Hinton [35], which can ignore half of the feature detectors in the training process. This technique can overcome the overfitting problem by decreasing interaction between detectors.

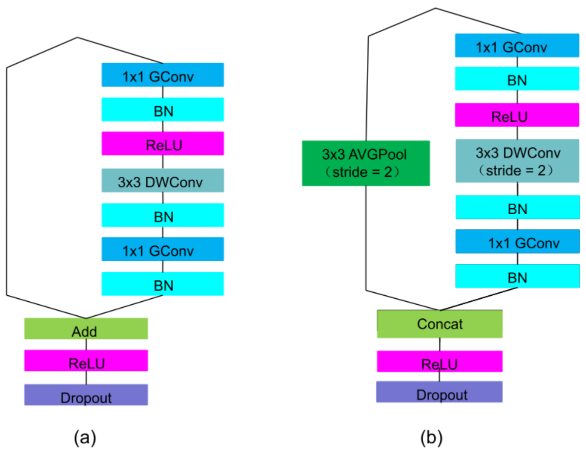

Figure 2. (a) Light Unit with stride =1. **(b)**Light Unit with stride =2.

2.3 Network Architecture

We propose the overall symmetrical framework in Table 1 based on the Light Unit. The network presented include one standard convolution and three stages which consist of a set of Light Units in the encoder part. Symmetrically, we apply four shortcut deconvolution modules in the decoder part. The shortcut deconvolution module can be written as follows:

$$F_{output}(x_n, x_n') = F_{main}(x_n) + F_{fine}(x_n) + F'(x_n') \quad (5)$$

where xn′ is the corresponding decoder layer to xn which is the last layer of each stage in encoder part, Fmain(xn) and Ffine(xn) are (3) and (5), and

$$F'(x_n') = Dropout(BN(\mathrm{Re}LU(Conv(x_n', W)), \beta = 0.5) \quad (6)$$

where β represents the percent of feature detectors ignored in the training process.
Finally, the softmax layer transforms the 2-D abstract features into a classification map with the size of 6 × width × height, where six means number of classes in the ground truth.

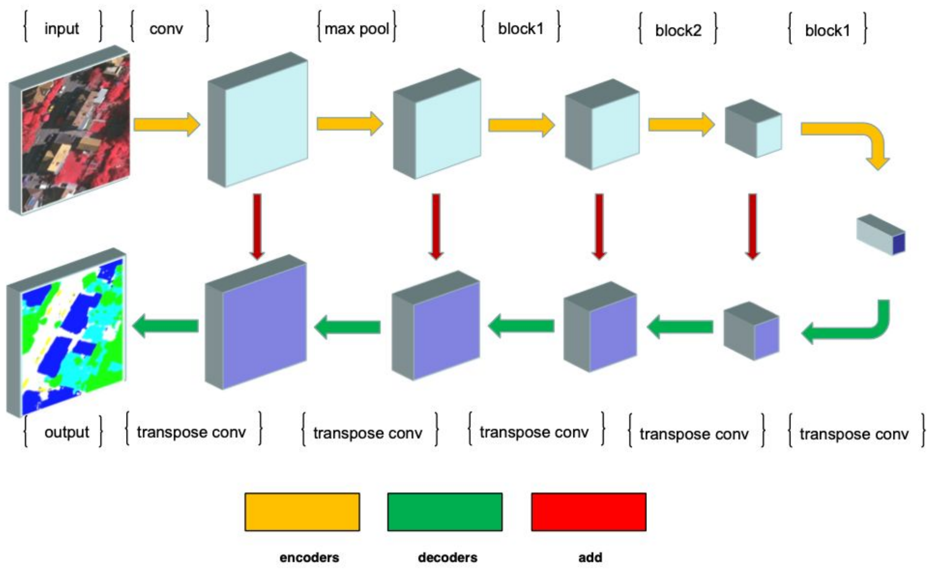

Figure 3. Architecture of DSFCN.

Table 1. DSFCN architecture

Layer name	Output Size	KSize	Stride	Repeat	Parameters
Original Image	128x128x3				
Conv1	64x64x24	3x3	2	1	672
MaxPool	32x32x24	3x3	2		
Stage2	16x16x100		2	1	378
	16x16x100		1	3	16,125
Stage3	8x8x200		2	1	5,375
	8x8x200		1	7	145,250
Stage4	4x4x400		2	1	20,750
	4x4x400		1	3	244,500
Transpose Conv1	8x8x200	3x3	2	1	720,200
Transpose Conv2	16x16x100	3x3	2	1	180,100
Transpose Conv3	32x32x24	3x3	2	1	21,624
Transpose Conv4	64x64x24	3x3	2	1	5,208
Output Image	128x128x6	3x3	2	1	1,302
Total of parameters complexity					**1,361,484**
Total of computation complexity (MAdds)					**66M**

3. Experiment and Discussion

Our experiments were conducted on the TensorFlow deep learning framework. It took 4 hours on the two ISPRS VHR(Very High Resolution) image datasets: Vaihingen[Fig. 4(a)-(c)] and Potsdam[Fig. 4(d)-(f), on GPU GeForce GTX 1080. The aim of the experiments is to classify all the pixels in the digital orthophoto maps into six categories[see Fig. 4]. The details of our experiments are divided into two parts. In the first part, we describe our processes to find the best mini batch size, group numbers and stage channel numbers for our DSFCN on Vaihingen dataset. Then contrast our model with some other models. The second part demonstrates the state-of-the-art performance of our network on the Potsdam dataset.

In the implementation, each model will keep training until no better result appears within 20 epochs. To address the vanishing gradient problem in the training process[Fig.5], we use Adadelta method [38] because of its unique updating quality(decrease from 0.5 to 0). In addition, we set weight decay to 4e-5. Other hyper-parameters follow [39]. The assessment metrics utilized in our experiments are overall accuracy(OA), kappa, model size, computation complexity(MAdds).

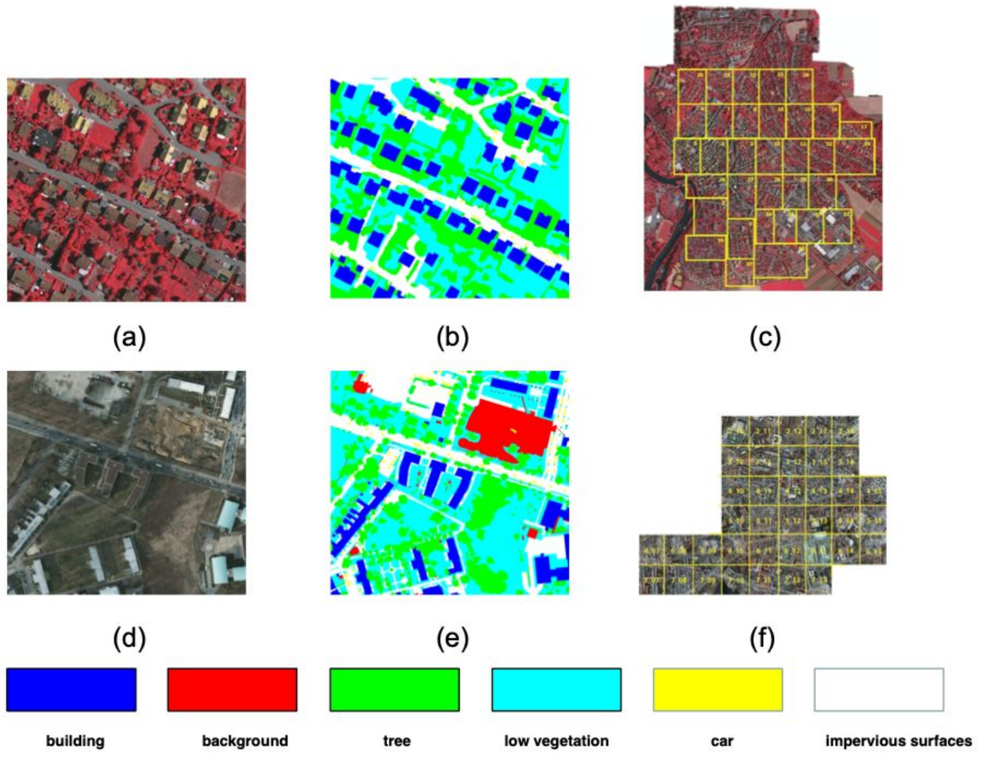

Figure 4. (a) Tile 33 in Vaihingen dataset. **(b)**Ground truth. **(c)** Vaihingen dataset. **(d)**Tile 3-12 in Potsdam. **(f)**Ground truth. **(e)**Potsdam dataset.

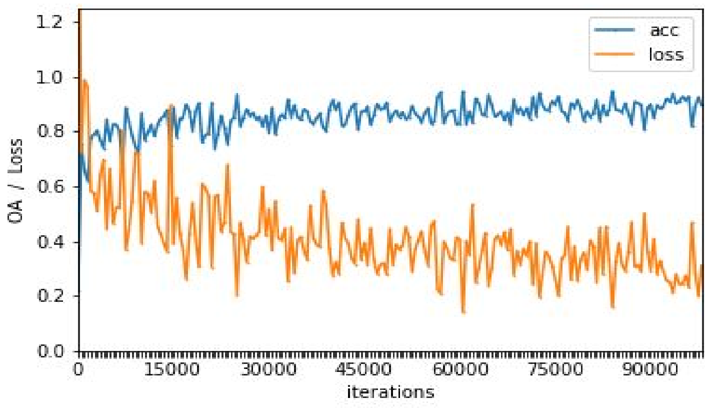

Figure 5. Monitoring training of DSFCN.

3.1 Experiments on Vaihingen dataset

3.1.1 Data details

The Vaihingen dataset contains three spectral bands: red (R), green (G), and near infrared (IR). In our training process, we only utilize digital orthophoto maps(DOMs) without digital surface models(DSMs). Among the DOMs, we use ID 1, 5, 7, 13, 15, 17, 21, 23, 28, 32,34, 37 as training data and ID 3, 11, 26, 30 as validation. Then we sliced these data into the images with a shape of 128 × 128 × 3 and each two adjacent small training images have 64% overlap. In addition, we also flip our original data horizontally and vertically to increase the training data. As a result, 23474 patches exist as training dataset and 1023 patches are used as validation dataset.

3.1.2 Minibatch size

In this part, to find the relationship between batch size and performance, our DSFCN is evaluated with different batch size while the group number is 1 and stage channels are 100, 200, 400. The Validation OA, Validation Kappa and Training OA are listed in Table 2. The result indicates that under the same settings, our framework with batch size 5 have a better performance on validation datasets, although it may not perform best on training datasets

Table 2. Accuracy assessment of different batch size on Vaihingen dataset. Group number =1, Stage depth = 100,200,400.

Minibatch Size	Validation OA	ValidationKappa	Training OA
3	0.84762	0.79321	0.88315
4	0.84631	0.79117	0.87212
5	0.85141	0.80209	0.90681
6	0.84266	0.78535	0.87978
10	0.84465	0.78908	0.89839
15	0.83845	0.78472	0.92140
20	0.84103	0.78853	0.92631
25	0.83944	0.78591	0.87832
30	0.83391	0.77865	0.88941
35	0.83830	0.78439	0.89267

3.1.3 Group number

As ShuffleNet [33] suggests, when the group number is larger, this model with channel shuffle performs better than the counterparts. In this part, to get the best result, we evaluate DSFCN with different group numbers(1,2,4,8) on constant condition that batch size is 5 and stage channels are 100,200,400. The complete results are exhibited in Table 3. On the contrary of [33], when we choose the group number as 1, the performance outperforms others both in accuracy and model size. That is because models with other group numbers block the process of information and representation flowing between channel groups.

Table 3. Accuracy assessment of different group numbers on Vaihingen dataset. Batch size = 5, Stage depth = 100,200,400.

Group Number	Validation OA	Validation Kappa	Training OA	Model Size
1	**0.85141**	**0.80209**	**0.90681**	**20.2MB**
2	0.84583	0.79474	0.87913	22.3MB
4	0.84656	0.79565	0.87608	26MB
8	0.83838	0.78457	0.87408	32.4MB

3.1.4 Stage depth

As Section 3 mentions, the stage depth plays a key role in the memory and computation complexity. In order to find the best depth which can achieve relatively small model size and maintain accuracy, we adapt our experiment with three different depths while the batch size is 5 and group number is 1. The result in Table 4 indicates that as the stage depth goes deeper, its relative size will be larger. Moreover, the Validation OA, Kappa and Training OA of Stage depth with 100, 200, 400 perform best accuracy while maintaining a relatively small model size.

Table 4. Accuracy assessment of different stage depth on Vaihingen dataset. Batch size = 5, Group number = 1.

CNN depth	Validation OA	Validation kappa	Training OA	Model Size
72_144_288	0.84545	0.79412	0.88167	**17.2MB**
100_200_400	**0.85141**	**0.80209**	**0.90681**	**20.2MB**
256_384_512	0.85131	0.80192	0.90169	30MB
384_768_1536	**0.85231**	**0.80301**	**0.90935**	151MB

3.1.5 Comparison experiments

To emphasize our superiority, we compare our model with another lightweight net: MobileNet. It's the same as our model except that the down-sampling layers are replaced. Meanwhile, we train SegNet as a comparison. The results are listed in Table 5 and Fig. 6. It is important to notice that our DSFCN with Depthwise Separable Convolution Blocks achieve slightly better accuracy than SegNet with much less model size and MAdds, which means the pointwise group and Depthwise convolutions help to improve networks' performance. In addition, as compared to Mobile_FCN, it shows that bottlenecks in our model make great contributions to reduce complexity and shortcut blocks produce better accuracy.

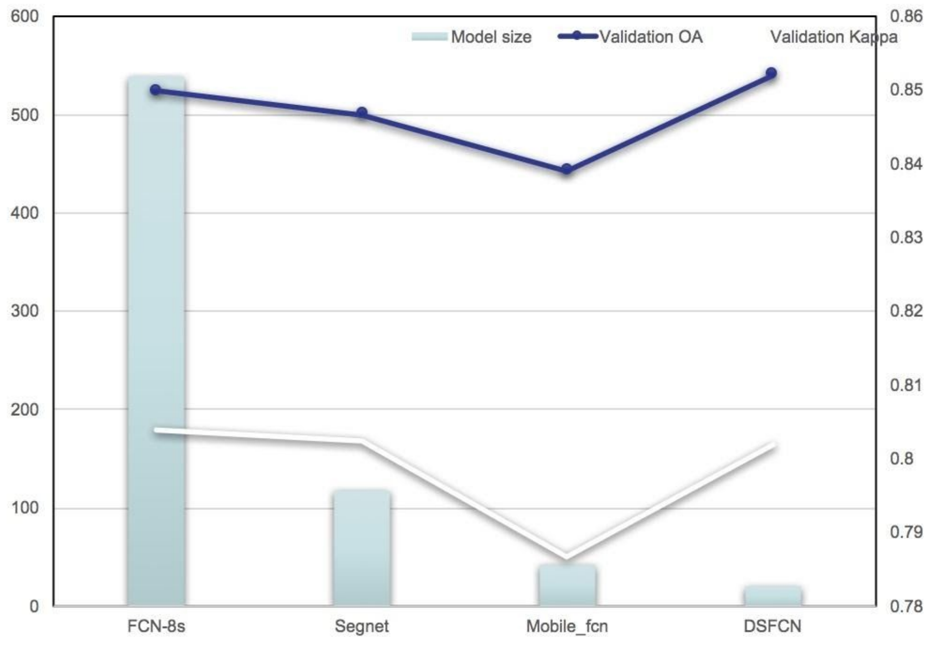

Figure6. Accuracy and model size assessment of different models

Table5. Metrics of comparison experiments with different models on Vaihingen dataset.

Model	Validation OA	Validation Kappa	Training OA	Model Size	MAdds
SegNet	0.8467	**0.8023**	0.8590	117.8MB	370M
Mobile_FCN	0.8394	0.7867	**0.9158**	42.3MB	184MB
DSFCN	**0.85141**	0.8020	0.9068	**20.2MB**	**66MB**

3.1.6 Output visualization

For the purpose to access the output label more obviously and directly, we list four examples of output images in Fig. 7. In addition, our detailed semantic segmentation results on Vaihingen dataset are exhibited in Table 6. Among the integrated result, building, low vegetation and impervious surfaces possess better accuracy. However, we found some regions labeled with trees are misclassified to cars (in the first row of the figure). Meanwhile, cars are also confused with trees. (in the second row of the figure). Moreover, the "salt and pepper" problems also exist.

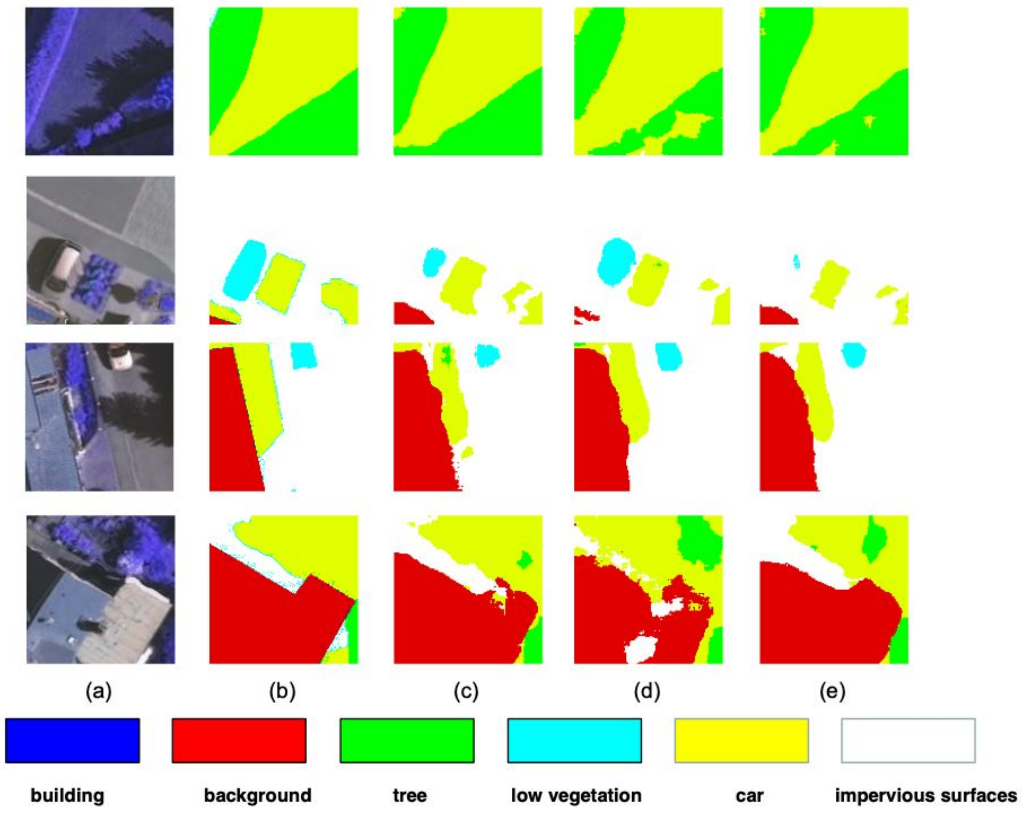

Figure 7. Example output of four images on Vaihingen data. **(a)** Original image. **(b)** Ground Truth. **(c)** SegNet. **(d)** Mobile_FCN. **(f)** DSFCN.

Table 6. Our result of DSFCN on Vaihingen dataset.

Reference/Predicted	Building	Background	Tree	Low Vegetation	Car	Surfaces
Building	**0.9227**	0.0021	0.0045	0.0206	0.0072	0.0426
Background	0.1271	**0.0012**	0.0253	0.0004	0.0321	0.8389
Tree	0.0051	0.0004	**0.8388**	0.1343	0.0051	0.0159
Low vegetation	0.0228	0.0013	0.1394	**0.7805**	0.0072	0.0486
Car	0.0255	0.0003	0.0017	0.0169	**0.7483**	0.2070
Surfaces	0.0555	0.0017	0.0162	0.0516	0.0342	**0.8404**

3.2 Experiments on Potsdam dataset

In order to demonstrate the generalization of our architecture, we compare our DSFCN with FCN-8s, SegNet and Mobile_FCN on Potsdam dataset.

3.2.1 Data details

The Potsdam dataset contains three spectral bands: red (R), blue (B), green (G) and near infrared (IR). Among the DOMs, we use 18 image tiles as training data and 6 tiles as validation (the same setting as [18]). Then, we sliced these data into the images with a shape of 128 × 128 × 3. In total, 38088 patches exist as training dataset and 12696 patches are used as validation dataset.

3.2.2 Comparison experiments

The training result is listed in Table 7 and Table 8. We find that DSFCN can get better accuracy than SegNet and Mobile_FCN. Meanwhile, it maintains the least memory and computation cost.

The results again demonstrate that our DSFCN with depthwise separable convolutions possesses the best performance. The result in Table 8 indicates that different from the result in the Vaihingen dataset, there is no confusion between cars and trees. However, trees and vegetation are still misclassified probably.

Table 7. Metrics of comparison experiments with different models on Potsdam dataset.

Model	Validation OA	Validation Kappa	Training OA	Model Size	MAdds
SegNet	0.8069	**0.7615**	0.8069	117.8MB	370M
Mobile_FCN	0.8200	0.7487	0.8203	42.3MB	184MB
DSFCN	**0.8256**	0.7596	**0.8559**	**20.2MB**	**66MB**

Table 8. Our result of DSFCN on Vaihingen dataset.

Reference/Predicted	Building	Background	Tree	Low Vegetation	Car	Surfaces
Building	**0.8834**	0.0171	0.0053	0.0094	0.0089	0.0756
Background	0.4336	**0.2002**	0.0106	0.0341	0.0424	0.2788
Tree	0.0119	0.0083	**0.6757**	0.2325	0.0046	0.0667
Low vegetation	0.0574	0.0163	0.0745	**0.7443**	0.0065	0.1007
Car	0.0102	0.0628	0.0249	0.0056	**0.8100**	0.0862
Surfaces	0.0549	0.0191	0.0231	0.0241	0.0162	**0.8623**

4. CONCLUSION

In this paper, we established a lightweight semantic segmentation architecture: DSFCN. Our new model integrates depthwise separable convolutions and Encoder-Decoder style. According to the experiments on ISPRS Vaihingen and Potsdam datasets, we prove that DSFCN is more efficient than the classical and SegNet. What's more, in the comparison experiments with Mobile_FCN, the result demonstrates that our residual block makes a great contribution to improve the accuracy, reduce the memory and computation cost. In the future, we will work on to implement the following trials:

1. We will add CRF blocks to our DSFCN to address the "salt and pepper" phenomenon in Figure 7.
2. We will run our network in embedded devices to access its performance and practicability.

Reference:

1. Mottaghi, Roozbeh, et al. "The role of context for object detection and semantic segmentation in the wild." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014
2. G.J.HayandG.Castilla, “Geographicobject-basedimageanalysis(GEOBIA):Anewnameforanewdiscipline,”inObject-BasedImageAnalysis. Berlin, Germany: Springer, 2008, pp. 75–89
3. Y. Qian, W. Zhou, J. Yan, W. Li, and L. Han, “Comparing machine learning classifiers for object-based land cover classification using very high resolution imagery,” Remote Sens., vol. 7, no. 1, pp. 153–168, Dec. 2014
4. R.C.Estoque,Y.Murayama,andC.M.Akiyama,“Pixel-basedandobjectbasedclassificationsusinghigh-andmedium-spatial-resolutionimageries in the urban and suburban landscapes,” Geocarto Int., vol. 30, no. 10, pp. 1113–1129, Nov. 2015.
5. X. Zhang, G. Chen, W. Wang, Q. Wang, and F. Dai, “Object-based landcoversupervisedclassificationforvery-high-resolutionUAVimagesusing stackeddenoisingautoencoders,”IEEEJ.Sel.TopicsAppl.EarthObserv. Remote Sens., vol. 10, no. 7, pp. 3373–3385, Jul. 201
6. S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149,2015. 2
7. V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. arXiv preprint arXiv:1412.6553,2014. 1,2,8
8. N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun. Fast convolutional nets with fbfft: A gpu performance evaluation. arXiv preprint arXiv:1412.7580,2014. 2
9. A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with lowprecision weights. arXiv preprint arXiv:1702.03044, 2017. 2
10. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature521.7553 (2015): 436-444.
11. B. Fulkerson, A. Vedaldi, and S. Soatto, “Class segmentation and object localization with superpixel neighborhoods,” in ICCV, 2009.
12. Krizhevsky, Alex, Ilya Sutskever, and GeoffreyE.Hinton."Imagenetclassificationwithdeepconvolutionalneural networks." Advances in neural information processing systems. 2012.
13. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.
14. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556(2014).
15. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556(2014).
16. V.Mnih,“Machinelearningforaerialimagelabeling,”Ph.D.dissertation, Univ. Toronto, Toronto, ON, Canada, 2013.
17. Z. Zhong, J. Li, W. Cui, and H. Jiang, “Fully convolutional networks for building and road extraction: Preliminary results,” in Proc. IEEE Geosci. Remote Sens. Symp., 2016, pp. 1591–1594.
18. Guanzhou Chen, Xiaodong Zhang, Qing Wang, Fan Dai, Yuanfu Gong, and Kun Zh, “Symmetrical Dense-Shortcut Deep Fully Convolutional Networks for Semantic Segmentation of Very-High-Resolution Remote Sensing Images ” in IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, 2018.
19. J. Sherrah, “Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery,” arXiv:1606.02585, 2016.
20. N. Audebert, B. L. Saux, and S. Lefvre, “Semantic segmentation of earth observation data using multimodal and multi-scale deep networks,” in Asian Conf. Comput. Vision. Springer, 2016, pp. 180–196
21. Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." arXiv preprint arXiv:1511.00561 (2015).
22. Jiang, Hao, and Cecilia Moloney. "A new direction adaptive scheme for image interpolation." Image Processing. 2002. Proceedings. 2002 International Conference on. Vol.3. IEEE,2002.
23. Dosovitskiy, A., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with convolutional neural networks. In: NIPS (2014)
24. F. Yu and V. Koltun,“Multi-scale context aggregation by dilated convolutions,” arXivpreprintarXiv:1511.07122, 2015.
25. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrousconvolution,andfullyconnectedCRFs,”IEEETrans.PatternAnal. Mach. Intell., vol. 40, no. 4, pp. 834–848, Apr. 2018
26. S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, “Conditional random fields as recurrent neural networks,” in Proceedings of the IEEE International Conference on Computer Vision,2015,pp.1529–1537
27. S.Han,J.Pool,J.Tran,andW.Dally. Learningbothweights andconnectionsforefficientneuralnetwork. InAdvancesin Neural Information Processing Systems, pages 1135–1143, 2015. 2.
28. J.Jin,A.Dundar,andE.Culurciello.Flattenedconvolutional neuralnetworksforfeedforwardacceleration. arXivpreprint arXiv:1412.5474,2014. 2.
29. H. Bagherinezhad, M. Rastegari, and A. Farhadi. Lcnn: Lookup-based convolutional neural network. arXiv preprint

arXiv:1611.06473,2016. 2
30. G.Hinton,O.Vinyals,andJ.Dean. Distillingtheknowledge inaneuralnetwork. arXivpreprintarXiv:1503.02531,2015. 2.
31. F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. arXiv preprintarXiv:1602.07360,2016. 1,7,8
32. Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXi preprint arXiv:1704.04861(2017).
33. XiangyuZhang, XinyuZhou, MengxiaoLin, JianSun, MegviiInc(Face++), "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices" arXiv:1707.01083v2,7 (2017)
34. S.IoffeandC.Szeged,"Batchnormalization:Acceleratingdeepnetwork trainingbyreducinginternalcovariateshift,"inInt.Conf.MachineLearn., 2015, pp. 448–456.
35. G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "**Dropout: A Simple Way to Prevent Neural Networks from Overfitting** ", JMLR(Journal of Machine Learning Research), 2014.
36. G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "**Improving neural networks by preventing co-adaptation of feature detectors** ", arXiv preprint arXiv: 1207.0580, 2012.
37. K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 1026–1034.
38. M. D. Zeiler, "ADADELTA: An adaptive learning rate method," arXiv preprint arXiv:1212.5701, 2012.
39. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," J. Mach. Learn. Res., vol. 11, pp. 3371–3408, 2010.