博士論文

Automatic Mapping Through Deep Convolutional Neural Networks

(ディープニューラルネットワーク(CNN)による自動マッピング)

伍 広明

WU GUANGMING

Acknowledgments

Just like many other years, 2020 is just an ordinary year for most of the people in the world. For me, 2020 is a little bit special that it will be the end of my 22 years of study as a student and the starting point of a career in society.

Around 30 years ago, I was born in a rural village located at the corner of Guangdong Province, China. Similar to most of the other communities, both educational resources and incentives are limited. However, I was very lucky to have Mrs. Peng to serve as my math teacher of the 6th grade. After so many years, while I almost forget all of her instructions, there is still one sentence remaining in my mind –" Learn more, explore more, and be a better yourself !". With the encouragement, I entered a better high school in another city, and then an undergraduate school in another province, and finally a graduate school in another country. Problems and challenges have changed from time to time, exploring more about the knowledge as well as the world is as usual. Getting a Ph.D. degree is just the end of the school study while learning and exploring should be a life-long adventure.

During my Ph.D. training, my deepest gratitude goes first and foremost to my tutors, professor Ryosuke Shibasaki and Xiaowei Shao, for all their patience, encouragement, guidance, and understanding. Without their help, I'm not able to finish this research. I would also like to thank all members (no matter current or former) of Shibasaki laboratory for their advice and help for not only my studies but also daily life.

Then, and most importantly, I want to thank my dear familiars. I want to thank my parents for their patience, consideration as well as both financial and mental support. For my beloved wife and child, your love and understanding are the energy sources that help me through these days and nights.

Finally, I want to thank the International Society for Photogrammetry and Remote Sensing (ISPRS), the National Topographic Office of New Zealand and NTT-GEOSPACE for their kind providing of the training and testing data as well as SAKURA Internet Inc. for providing me the "koukaryoku" GPU server for the experiment. I also gratefully acknowledge the financial support from the China Scholarship Council (CSC).

Abstract

Automatic mapping from aerial or satellite imagery is an essential and challenging task because of the variety of backgrounds, building textures, and imaging conditions. To achieve automatic mapping, three challenging tasks, including polygon feature extraction, line feature extraction, and model transfer, should be addressed.

To achieve polygon feature extraction, deep-learning methods, especially fully convolutional networks (FCNs), has become a popular option. Compared with traditional solutions, these approaches have shown promising generalization capabilities and precision levels in various datasets of different scales, resolutions, and imaging conditions. To achieve superior performance, many pieces of research have focused on constructing more complex or deeper networks. However, using an ensemble of different fully convolutional models to achieve better generalization and to prevent overfitting has long been ignored. In this research, we design four stacked fully convolutional networks (SFCNs), and a feature alignment framework for multi-label land-cover segmentation. The proposed feature alignment framework introduces an alignment loss of features extracted from basic models to balance their similarity and variety. Experiments on a very high resolution(VHR) image dataset with six categories of land-covers indicates that the proposed SFCNs can gain better performance when compared to existing deep learning methods. In the 2^{nd} variant of SFCN, the optimal feature alignment gains increments of 4.2% (0.772 vs 0.741), 6.8% (0.629 vs 0.589), and 5.5% (0.727 vs 0.689) for its f1-score, jaccard index, and kappa coefficient, respectively.

For efficient line feature extraction, many indirect or direct approaches have been proposed. In recent years, due to the rapid development of deep convolutional networks, line feature extraction can also treat as a particular polygon feature extraction task that deals with the extremely biased distribution of negative and positive pixels. The existing methods are mainly focused on the network design that misalignments and rotations presented in manually created annotations are long ignored. Due to the limited positive samples, the misalignments and rotations significantly reduce the correctness of pixel-to-pixel loss that might lead to the gradient explosion. To overcome this, we propose a nearest feature selector(NFS) to dynamically realign the prediction and slightly misaligned annotations. The NFS can seamlessly integrate into existing loss functions and prevent misleading by errors or misalignment of annotations. Experiments on a large scale image dataset with centered buildings and corresponding building outlines indicate that the additional NFS brings higher performance when compared to existing naive loss functions. In the classic L1 loss, the addition of NFS gains increments of 8.8% of f1-score, 8.9% of kappa coefficient, and 9.8% of the jaccard index.

Generalization is an essential criterion of the algorithm. A well-generalized model should work appropriately on various areas and data sources. Due to the limited training data, current data-driven algorithms, including deep convolutional networks(DCNs), are susceptible to training data that can not be applied to new data directly. Different from existing methods that are trying to improve model generation capability using limited data, we propose an integrated pipeline to generated testing data that share a similar characteristic of training data. Our model transfer pipeline is consists of a super-resolution and a colorization model that can convert low-resolution panchromatic images from the satellite into high-resolution color images. Experiments on an image dataset with satellite and corresponding aerial imagery show that the pre-trained model achieves significantly higher performance on images processed by the integrated pipeline. Compared to the performance on original satellite images, even with a slight decline of precision(4.4%), the addition of SR and colorization leads to 63.6%, 29.6%, 76.6%, and 51.8% increments of recall, f1-score, jaccard index, and kappa coefficient, respectively.

Contents

List of Figures

List of Tables

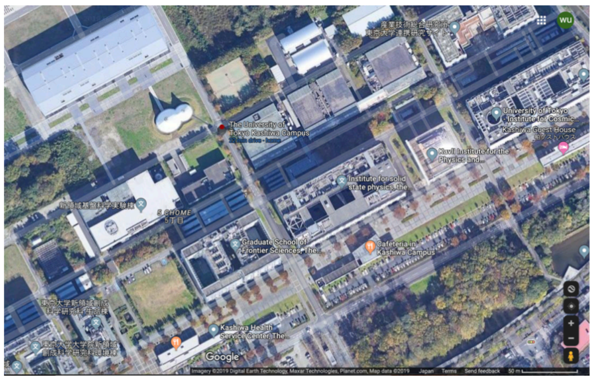
Chapter 1

Introduction

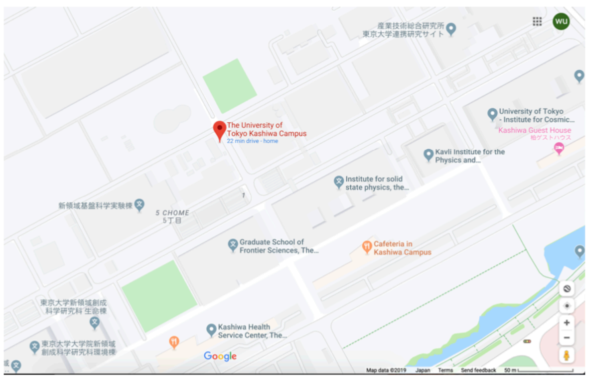
Contents

The distributions and changes of natural and artificial surfaces, such as grasslands, forests, buildings, and roads, are essential for many applications such as urban planning, navigation [1], land-used management [2], and forest monitoring [3]. Traditionally, this information was obtained by labor-intensive and time-consuming field surveys [4]. The ability to achieve precise and cost-efficient updating of land cover is a long-existing demand for remote sensing. Over the last few years, with the emerging of innovative technologies, the cost, as well as the difficulty of capturing very high resolution(VHR) aerial imagery, has significantly declined [5, 6]. Thus, robust and precise methods for automatic generation of digital maps from captured aerial or satellite imagery are the core of the whole solution.

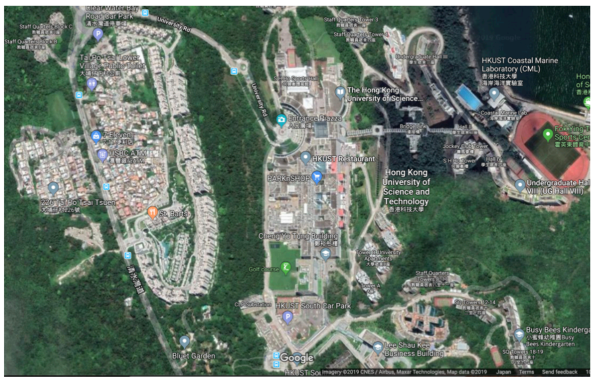
As shown in Figure 1.1, compared to satellite images, the digital maps simplified and summarized the RGB values satellite images into two main categories: polygon features(*e.g.*, buildings, green lands, and lakes) as well as line features(*e.g.*, roads, and building outline). Besides, even with a significantly different appearance of satellite images from different locations, the digital maps should maintain a similar style. Thus, to achieve automatic mapping, there are mainly three tasks: (1) Polygon Feature Extraction, (2) Line Feature Extraction, and (3) Model Transfer. The (1) and (2) make sure that the methods should be able to extract both polygon and line features efficiently. With (3), the algorithm built for one location should also work for another location or the same location with a different appearance.


The University of Tokyo Kashiwa Campus
Kavli Institute for the Physics and...
Institute for solid state physics, the...
Graduate School of Frontier Sciences, The...
Cafeteria in Kashiwa Campus
Kashiwa Health Service Center, The...
5 CHOME
Google

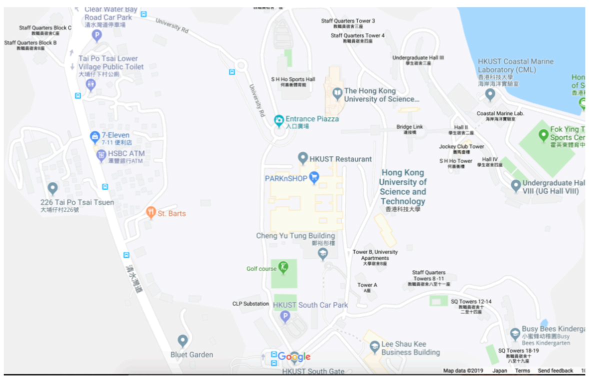
(a) Satellite Imagery of UTokyo


The University of Tokyo Kashiwa Campus
University of Tokyo - Institute for Cosmic...
Kashiwa Guest House
Kavli Institute for the Physics and...
Institute for solid state physics, the...
5 CHOME
Graduate School of Frontier Sciences, The...
Cafeteria in Kashiwa Campus
Kashiwa Health Service Center, The...
Google

(b) Digital map of UTokyo


HKUST Coastal Marine Laboratory (CML)
The Hong Kong University of Science...
Entrance Piazza
HKUST Restaurant
PARKnSHOP
Hong Kong University of Science and Technology
Undergraduate Hall VIII (UG Hall VIII)
HKUST South Car Park
Busy Bees Kindergarten
Bluet Garden
Google

(c) Satellite Imagery of HKUST


Clear Water Bay Road Car Park
Tai Po Tsai Lower Village Public Toilet
The Hong Kong University of Science...
Entrance Piazza
HKUST Coastal Marine Laboratory (CML)
7-Eleven
HSBC ATM
HKUST Restaurant
PARKnSHOP
Hong Kong University of Science and Technology
Fok Ying Tung Sports Center
Undergraduate Hall VIII (UG Hall VIII)
226 Tai Po Tsai Tsuen
St. Barts
Cheng Yu Tung Building
Golf course
HKUST South Car Park
Busy Bees Kindergarten
Lee Shau Kee Business Building
Bluet Garden
HKUST South Gate
Google

(d) Digital map of HKUST

Figure 1.1: Example of Satellite images and corresponding digital images from Google Maps. (**a**) Satellite image of kashiwa campus, The University of Tokyo, (**b**) Digital map of kashiwa campus, The University of Tokyo, (**c**) Satellite image of The Hong Kong University of Science and Technology, and (**d**) Digital map of The Hong Kong University of Science and Technology.

1.1 Polygon Feature Extraction

Polygon feature extraction can be viewed as a semantic segmentation of different polygon features from input aerial or satellite images. Over the past decades, a significant amount of polygon feature extraction algorithms have been proposed. According to the conditions of image datasets such as scale, color space, and resolution, various automatic segmentation methods have been introduced. Depending on whether it is necessary to have ground truth, these methods can be mainly divided into two categories : (1) unsupervised methods and (2) supervised methods.

1.1.1 Unsupervised Methods

1.1.1.1 Threshold-based Methods

Image thresholding is a simple and commonly used segmentation method. Pixels with different values are allocated to different parts according to manually or automatically selected thresholds [7]. Usually, image thresholding is not capable of differentiating among various regions with similar grayscale values.

1.1.1.2 Edge-based Methods

Edge-based methods adopt edge-detection filters, such as Laplacian of Gaussian [8], Sobel [9] and Prewitt [10], to detect the abrupt changes among neighboring pixels and generate boundaries.

1.1.1.3 Region-based Methods

Region-based methods segment different parts of an image through clustering [11–14], region-growing [15] or shape analysis [16,17]. Due to the variety of illuminance and texture conditions of an image, edge-based or region-based methods cannot provide stable and generalized results.

1.1.2 Supervised Methods

Because of manually adjustable parameters and the lack of need for ground truth, unsupervised methods are more comfortable to implement and are widely adopted for small scale datasets. However, for larger datasets, as the variety and complexity increase, the performance of unsupervised segmentation methods usually lacks generalization capability [18]. In direct contrast, supervised methods utilize the ground truth to learn segmentation patterns and then apply it to new data. For supervised methods, the segmentation problem is converted into a pixel-to-pixel image classification where pixels of different parts are classified into their corresponding categories [19]. Because the segmentation is made by classifying each pixel, these methods generally produce more precise segmentation.

1.1.2.1 Hand-crafted Features

Conventional supervised methods usually undergo a two-step procedure of feature extraction and classification. The spatial and textural features of an image are extracted through mathematical feature descriptors, such as haar-like [20], scale-invariant feature transform [21], local binary pattern [22], and histogram of oriented gradients (HOG) [23]. After that, the prediction for every pixel is made on the basis of the extracted features through classifiers such as support vector machines [24], adaptive boosting (AdaBoost) [25], random forests [26] and conditional random fields (CRF) [27]. However, because of the complexity of building structures and also because of strong similarities with other classes (*e.g.*, pieces of roads), the prediction results rely heavily on manual feature design and adaptation, which easily leads to bias and poor generalization.

1.1.2.2 Patch-based Convolutional Neural Network(CNN)

With the development of algorithms, computational capability, and the availability of big data, convolutional neural networks (CNNs) [28] have attracted more and more attention in this field. Unlike two-step methods requiring artificial feature extraction, CNNs can automatically extract features and make classifications through sequential convolutional and fully connected layers. The CNN method can be considered as a one-step method that combines feature extraction and classification within a single model. Since the feature extraction is learned from the data itself, CNN usually possesses better generalization capability.

In the early stages, patch-based CNN approaches label a pixel by classifying the patch that centers around that pixel [29, 30]. Even for a small patch of 32×32 pixels, to cover the whole area, the memory cost of these patch-based methods increases by 32×32 times. For larger areas or patch sizes, these approaches encounter dramatically increased memory cost and significantly reduced processing efficiency [31].

1.1.2.3 Fully Convolutional Networks (FCNs)

To avoid patches, the fully convolutional networks (FCNs) utilizes a fully convolutional network architecture [32]. The architecture enables direct pixel-to-pixel translation of the input images to the ground truth. In this manner, the FCN method significantly improves computational efficiency and model performance [31].

As shown in Figure 1.2, in classic FCNs (FCN32s, FCN16s, and FCN-8s) [32], the methods adopt multiple-scale bilinear upsampling operations to generate segmentation output with the same height and width of input. These operations lead to information loss that affects the precision of prediction.

Recently, more advanced and accurate FCN-based methods have been developed [33]. These methods improve model performance through different strategies.

- U-Net. The U-Net architecture was proposed by Ronneberger *et al.* [34] for medical image segmentation. This method adopts bottom-up/top-down architecture with skip connections that combine both the lower and higher layers to generate the final output, resulting in better performance.

- DeconvNet. The DeconvNet architecture was published at IEEE International Conference on Computer Vision(ICCV) 2015 [35]. Instead of upsampling layers, the model introduced a learnable deconvolution(*i.e.*, convolution transpose) operations to increase the height and width of the intermediate features gradually.

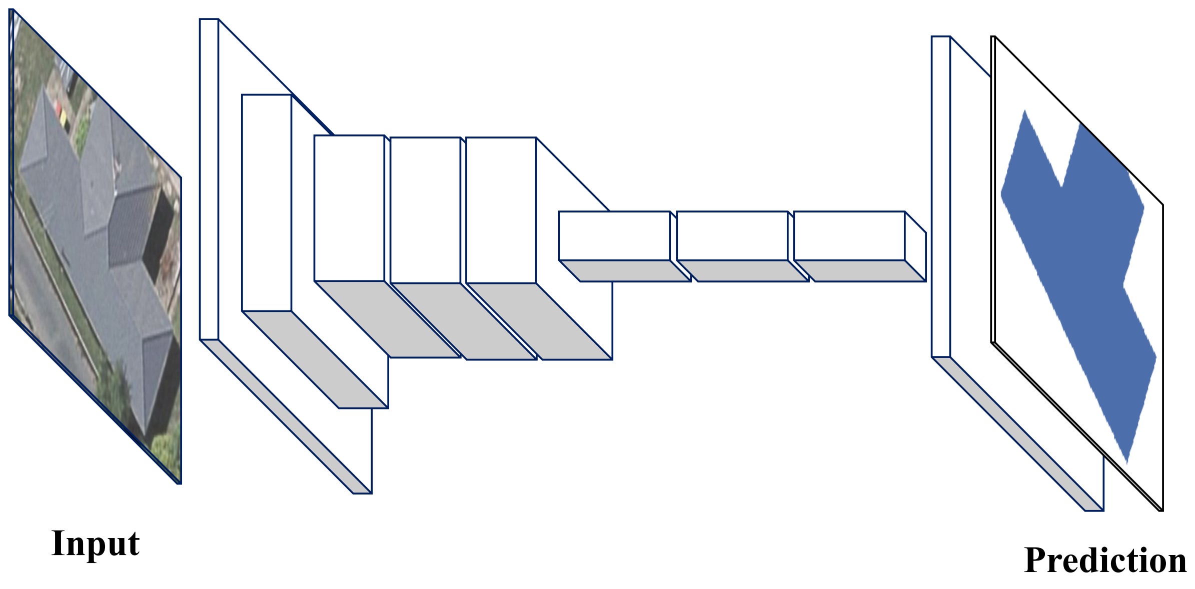


Figure 1.2: Network architecture of the FCN.

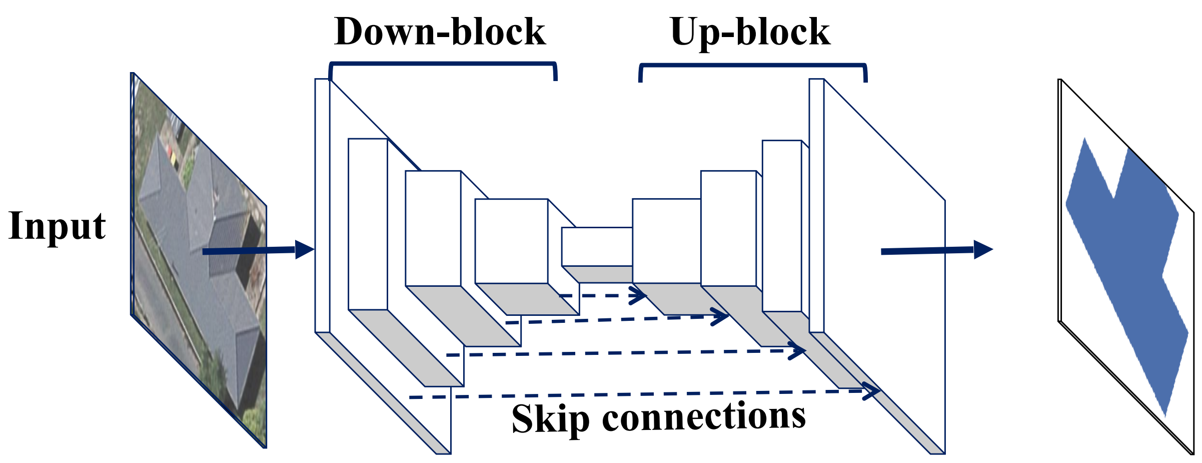


Figure 1.3: Network architecture of the U-Net.

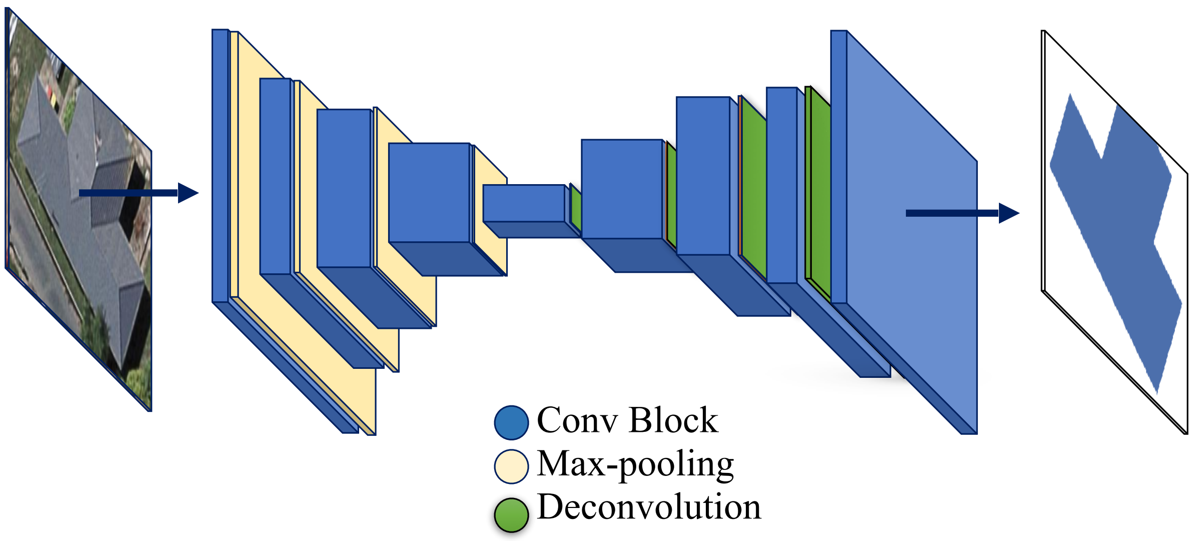


Figure 1.4: Network architecture of the DeconvNet.

- SegNet. The SegNet architecture was proposed by Badrinarayanan *et al.* [36] in 2017. This method proposed a deep convolutional encoder-decoder architecture with unpooling layers to consequentially upsampled height and width of intermediate featured maps. With unpooling operation, the SegNet model can avoid information loss during max-pooling and thus brings a better result.

- MC-FCN. In our previous study, we proposed a multi-constraint fully convolutional network (MC–FCN) [37]. The MC–FCN model adopts the basic structure of a U–Net and adds multi-scale constraints for variant layers. Here, an optimization target between the prediction and the corresponding ground truth for a specific layer is defined as a constraint. During every iteration, parameters are updated through the multi-constraints, which prevents the parameters from biasing to a single constraint. Also, the constraints are applied to different layers, which helps to optimize the hidden representation of variant layers better.

These advanced methods further develop the potential of fully convolutional networks. However, with more complex architectures and stronger representation capabilities, overfitting becomes inevitable [38].

Overfitting is a long-existing problem in deep learning. This problem is more critical for smaller datasets. Several approaches, including early stopping, data augmentation, regularization, and ensemble learning, are proposed for this problem. The early stopping approach stops the training model before convergence to prevent overfitting [39,40]. For the data augmentation approach, the original images are rotated, resized, random cropped, or re-colorized to generate more training samples and increase the variety of data [41]. As for regularization, extra penalty (*e.g.*, L1/L2) [42, 43] or dropout [44] is implemented to reduce and regulate the representation capability of the model. By contrast, ensemble learning combines several models to generate a final prediction [45]. Owing to its capability of utilizing a variety of different models, biased predictions from one model can be compensated for by other models, and better

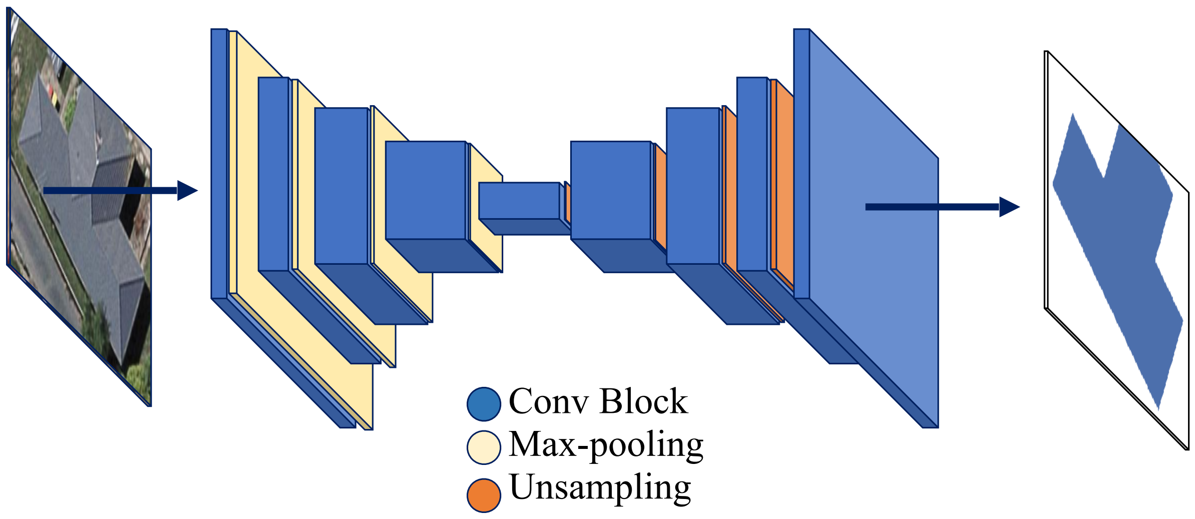


Figure 1.5: Network architecture of the SegNet.

results can be produced. Currently, ensemble learning is mainly applied to patch-based CNN for pixel-level classification [46]. Ensemble learning has not received any attention in FCN-based architectures. Besides, research on ensemble learning is mainly focused on various numbers or combinations of basic models. The studies on the combination approaches of different basic models are not sufficient.

To explore the capability of ensemble learning using fully convolutional networks, we design four stacked fully convolutional networks(SFCNs) using FCN-8s, U-Net, and FPN. Furthermore, we propose a feature alignment framework for efficient ensemble learning, which enhances the relations between basic models. Compared with traditional ensemble learning approaches, the proposed method implements basic segmentation loss between prediction and corresponding ground truth as well as extra alignment loss between features that are extracted separately from different basic models. The value of the alignment loss is determined by the consistency of features extracted by different models. If these features are similar, the alignment loss is zero. During iterations, the optimizer is required to update parameters to reduce the value of the weighted sum of segmentation loss and alignment loss. Thus, the optimized network is capable of generating predictions using features extracted from basic models that contain a balance of similarity and variety.

A VHR image dataset demonstrates the effectiveness of the proposed feature alignment framework (refer to Figure 3.1). In comparative experiments, the performances of achieved by the proposed method (SFCNv3, +FA) are 0.785($\pm$0.004) of F1-score, 0.646($\pm$0.005) of jaccard index [47], and 0.742($\pm$0.005) of kappa coefficient [48], respectively. Furthermore, sensitivity analysis indicates that the proposed feature alignment can control the balance between similarity and variety of features extracted from different basic models. By optimizing the feature alignment level, ensemble fully convolutional networks gain better model performance.

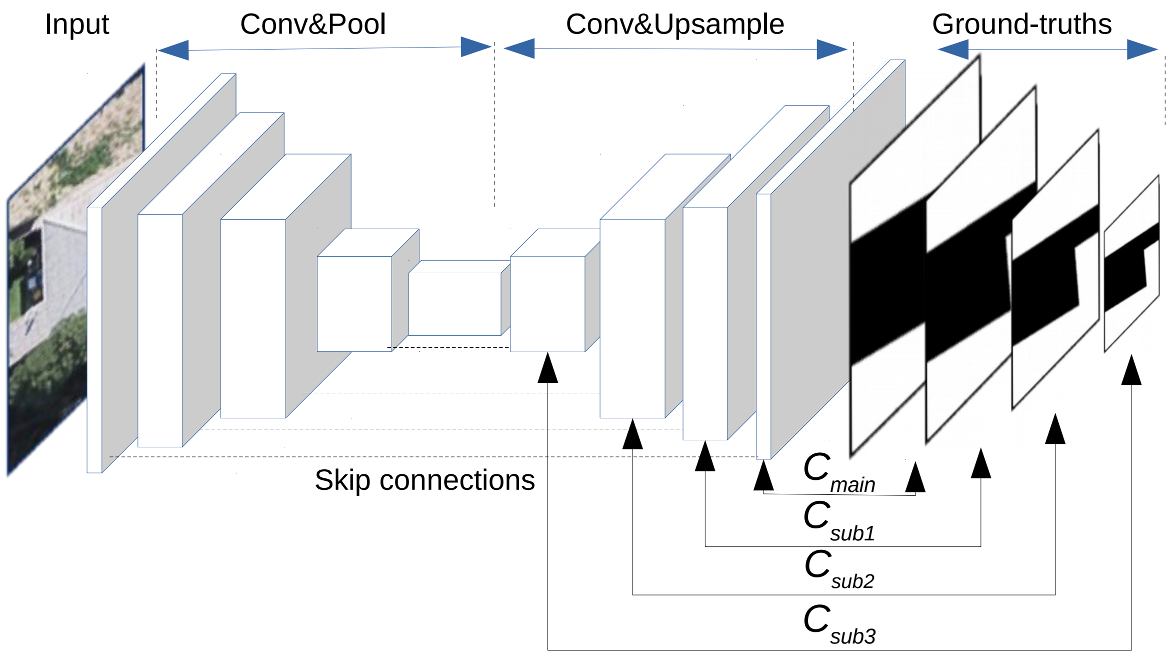


Figure 1.6: Network architecture of the MC–FCN.

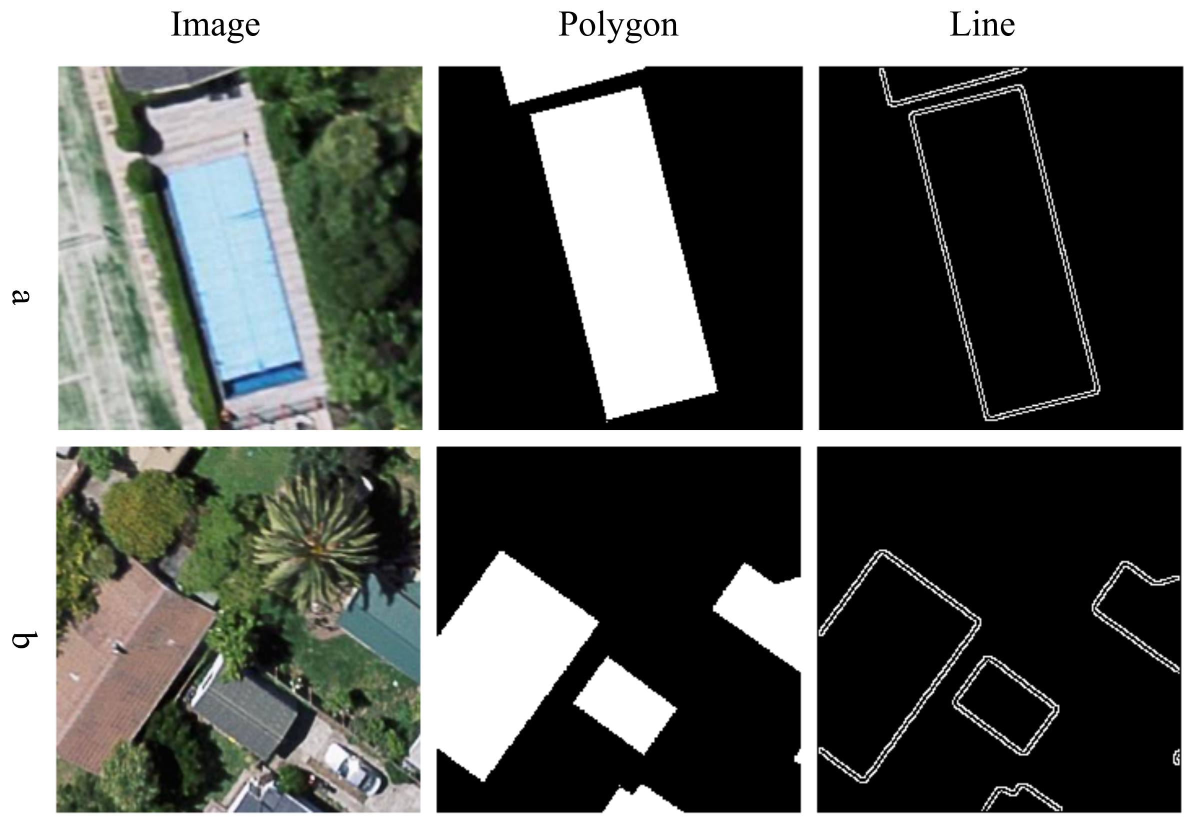


Figure 1.7: Example of polygon feature and line feature from aerial image.

1.2 Line Feature Extraction

Different from polygon features, line features occupy very few pixels(*e.g.*, building outline shown in Figure 1.7). Thus, the automatic extraction of the line feature is more complicated. To extract line features, there are mainly two approaches: (1) indirect approach, and (2) direct approach.

1.2.1 Indirect Approach

For the indirect approach, instead of extracting target line features directly from input aerial or satellite image, it first performance semantic segmentation. With accurately segmented feature maps, outlines, or boundaries among different semantic parts can be easily achieved. In these methods, since the line feature is derived from segmentation maps, their performances highly rely on the robustness of polygon extractions.

In principle, all polygon extraction methods mentioned above can also be used for indirect line extraction. However, due to the sensitivity of outline/boundary, training with only semantic information usually leads to inconsistent outline or boundary. To prevent this, we proposed a boundary regulated network(BR-Net) [49] to achieve building segmentation as well as outline extraction.

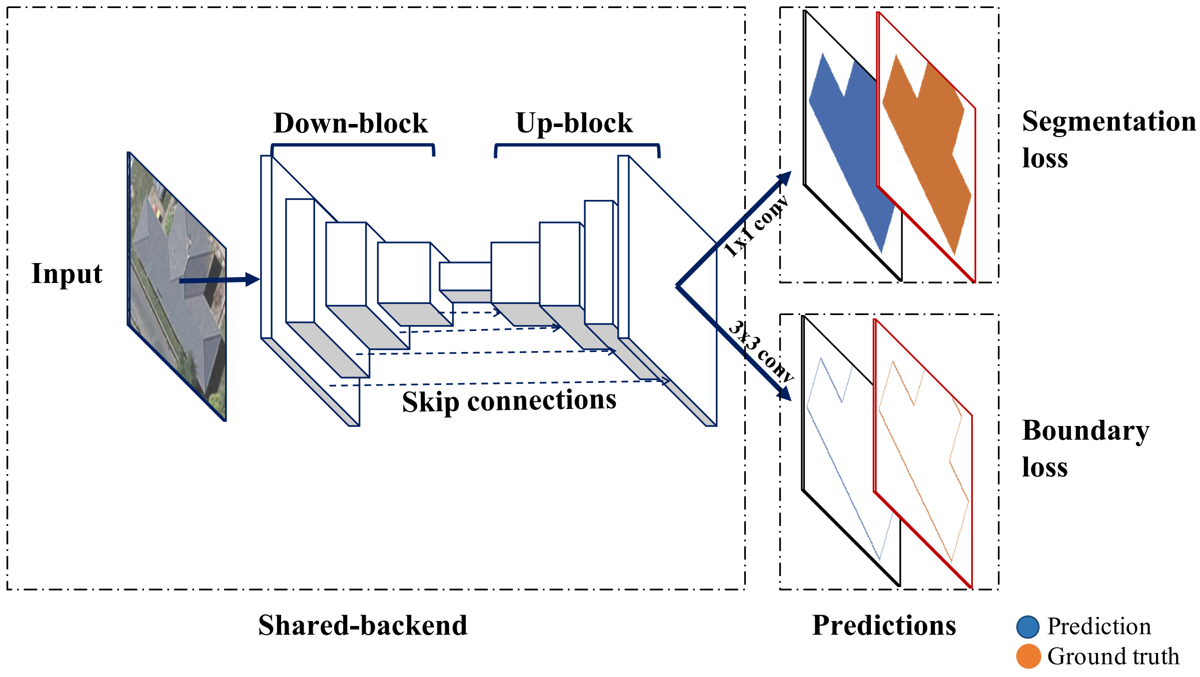


Figure 1.8: Network architecture of the BR-Net.

1.2.1.1 Boundary Regulated Network

As shown in Figure 1.8, the boundary regulated network (BR-Net) method consists of a shared backend utilizing a modified U-Net and a multi-task framework to generate predictions for segmentation maps and building outlines based on a consistent feature representation from the shared backend. In the proposed BR-Net, the optimizer has two main tasks. It must ensure that both the segmentation and outlines of the prediction results are as close as possible to those of the ground truth. In this manner, in every iteration, parameters are updated by considering both segmentation and outlines, which prevents parameters from focusing on surrounding pixels and utilizes a more comprehensive range of global information.

As shown in Figure 1.9, compared to existing methods, the proposed BR-Net can achieve superior performance on both building polygon segmentation and outline extraction. However, even in BR-Net, the outlines are extracted indirectly.

1.2.2 Direct Approach

Different from the indirect approach, direct approach extracts line features directly from input aerial or satellite images. Traditionally, line feature extraction is done by edge operators that estimate the gradient of the image intensity function.

1.2.2.1 Edge Operators

With the operators, different finite-difference approximations of the gradient are computed to generate boundaries among different zones [50]. Due to their simplicity, these methods are fast and unsupervised. However, when faced with a slightly complicated task, such as extracting

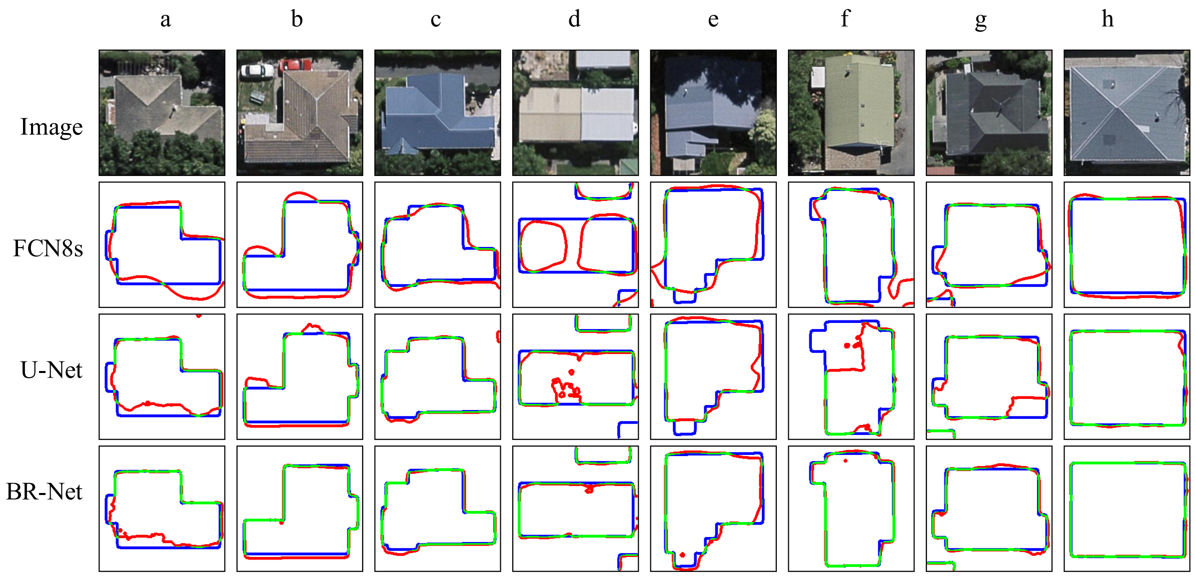


Figure 1.9: Representative results of single-building-level outline extraction by FCN8s, U-Net and, BR-Net. The green, red, blue, and white channels in the results represent true positive, false positive, false negative, and true negative predictions, respectively.

building outlines from an aerial image, extracted outlines from these methods are filled with noises(*e.g.*, building outline extracted by canny operator [51] in Figure 1.10).

1.2.2.2 Deep Convolutional Networks

Even with much fewer positive examples, line feature extraction can also be considered as a semantic segmentation or pixel-level classification problem [52]. With the rapid development of deep learning algorithms, many efficient deep learning networks, such as RSRCNN [53], ResUNet [54], and D-LinkNet [55], are proposed for automatic extraction of line features(*e.g.*, roads).

However, these models are focusing on deeper network architectures to achieve better utilization of the feature representation capability of hidden layers. Another critical issue in line feature extraction still exists. Regardless of how these models generate predictions, their loss functions are computed directly from the pixel-to-pixel similarity of the ground truth. Due to the extremely biased distribution of positive and negative pixels, gradient exploding during training becomes a severe problem. Additionally, because of the occasional human errors, there are inevitable several or tens of pixels misalignment between annotation and corresponding aerial image. Due to the much fewer pixels of line features, the pixel-to-pixel losses are very sensitive to these misalignments.

In light of this issue, we propose a nearest feature selector(NFS) module, which enables dynamic re-alignment between ground truth and prediction. At every iteration, a dynamic matching between ground truth and prediction is performed to determine the matched position. Then, the overlapped areas of both ground truth and prediction are used for further loss computation. Since the NFS is used for the upper stream, it can be seamlessly integrated into all existing loss functions. The effectiveness of the proposed NFS module is demonstrated by a VHR image

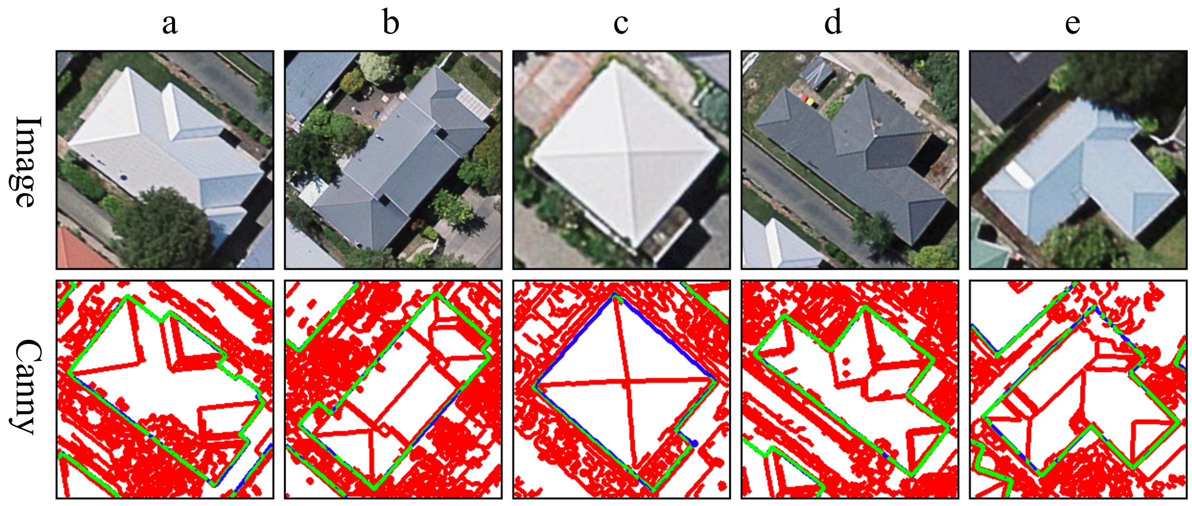


Figure 1.10: Example of building outlines extracted by Canny operator($\sigma = 1$).

dataset located in New Zealand(refer to Figure 3.2 and Figure 3.3). In comparative experiments, under different network architecture as well as loss functions, addition of NFS shows significantly higher values of F1-score, jaccard index [47], and kappa coefficient [48].

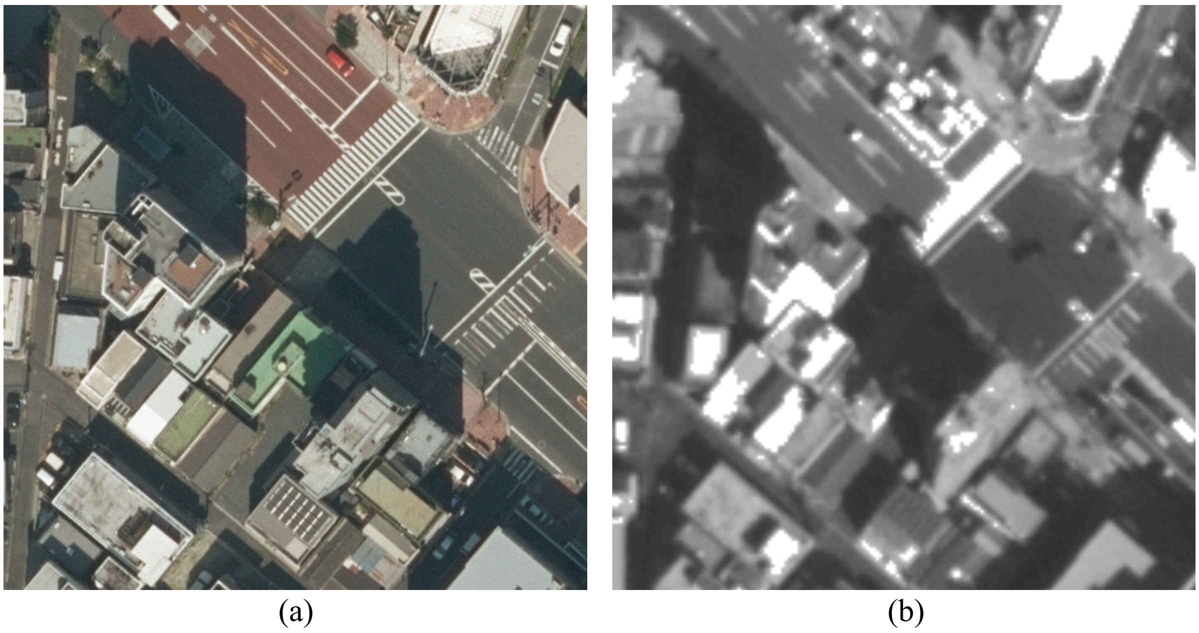


Figure 1.11: Sample patch of RGB aerial and panchromatic satellite images from some location. (a) High resolution (0.16 m) RGB aerial image, and (b) Low resolution (0.5 m) panchromatic satellite image.

1.3 Model Transfer

Ideally, if there is sufficient training data, we can train a perfect model that can work for all conditions. However, in practice, only limited annotations are provided. Thus, to achieve large-scale automatic mapping, the model built for one location should also work at another location or the same location but from different data sources. In remote sensing, transferring an aerial model to satellite images is a frequent demand of model transfer.

Typically, the aerial images have higher spatial resolution and more abundant chrominance information. Due to the difference in resolution and color space, the model trained on aerial images undergoes performance degradation significantly on satellite images. To avoid this, we propose a model transfer pipeline that integrated both resolution and color transferring to synthesized high resolution, colorized satellite images.

1.3.1 Resolution transfer

To achieve high resolution(HR) images, a simple solution is to upscale low resolution(LR) image through bilinear or bicubic interpolations. However, interpolation would generate insufficient large gradients along edges and high-frequency regions by only weighted averaging neighboring LR pixel values. For example, small buildings would not be the same as larger ones, even if up-scaled.

Rather than interpolation, super-resolution (SR) has offered a promising alternative solution [56]. Aimed at increasing the image resolution while providing finer spatial details than those captured by the original acquisition sensors, SR could balance the size and detail of land features between the training and testing datasets to a certain degree [57].

1.3.2 Color transfer

Transferring chrominance information from an aerial image to a panchromatic image can be viewed as a particular form of image colorization. The goal of transferring is to reconstruct RGB color from panchromatic input. According to whether the usage of reference images or not, existing methods can mainly be categorized into two groups: (i) scribble-based methods, and (ii) example-based methods.

As for scribble-based methods, user-specific scribbles and corresponding optimization frameworks are integrated to correctly propagated color information the whole input image [58–60]. The performances of these methods are affected by scribbles provided by users.

Different from scribble-based methods, example-based methods learn a colorization pattern from existing examples. Liu *et al.* [61] proposed an example-based colorization which transfers color in an illumination-independent domain using online color references. Other than color statistics, Wu *et al.* [62] introduced a content- based method that establishes semantic correspondences of the regions between source and target images. In recent years, deep convolutional networks, which learn the colorization pattern directly from large-scale image pairs, provide a more convenient yet effective option [63, 64]. Zhang *et al.* [65] turned image colorization as a classification task which estimates chrominance values in CIE LAB colorspace. Later, Iizuka *et al.* [66] proposed a multi-task network which combines both color prediction and scene classification to achieve more natural result.

Chapter 2

Methods

Contents

2.1 Polygon Feature Extraction

2.1.1 Scheme

Figure 2.1 presents the workflow for polygon extraction. The orthophoto, as well as their corresponding ground truth, are divided into two sets for training and testing. A sliding window with a stride of 224 pixels is applied to each tile of the training set to generate image patches with a size of 224 $\times$ 224 pixels. After data preprocessing, the image patches are shuffled and split into two groups that include training (70%), and cross-validating (30%). The number of samples in training and cross-and validation are 744 and 312, respectively. Through several cycles of training and cross-validation, the hyper-parameters are determined and optimized. Then, the predictions generated by the optimized model are further evaluated by the tiles in the test set. For performance evaluations, we choose three commonly used evaluation metrics, namely, jaccard index, f1-score, and kappa coefficient. These metrics are computed without post-processing operations [27, 67] for better estimation of experimental methods.

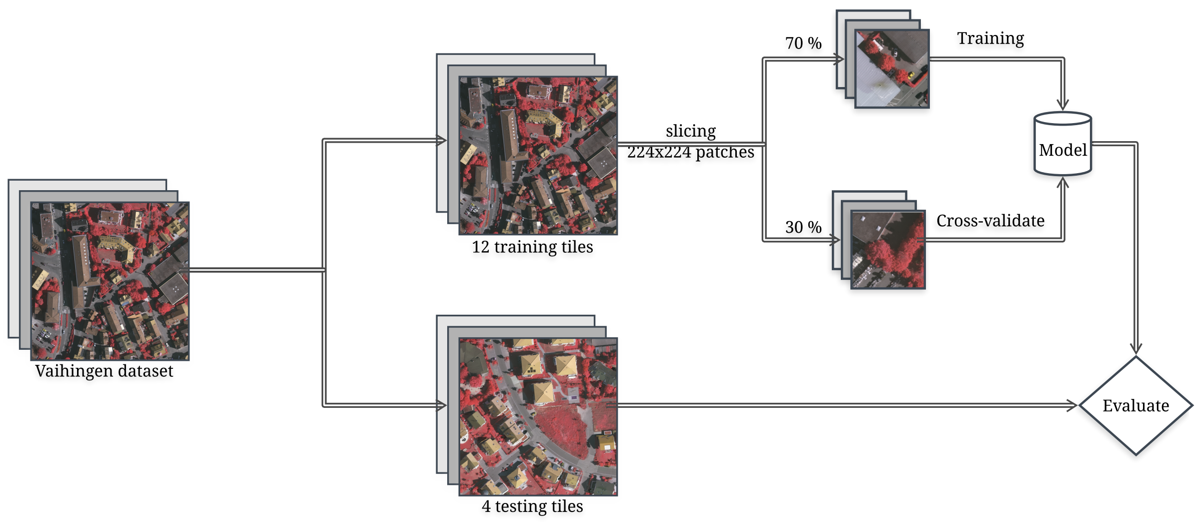


Figure 2.1: Experimental workflow of this research. The existing methods, as well as the proposed model, are trained and evaluated by 224 x 224 image patches extracted from original dataset.

2.1.2 Stacked Fully Convolutional Networks(SFCN)

After the invention of FCN in 2015, FCN and FCN-based methods have become a gold standard for many image segmentation tasks [68, 69]. Compared to conventional patch-based CNN methods, FCN-based models significantly improve computational efficiency and performance. Advanced FCN-based models further enhance feature representation capabilities and improve model performance through various approaches. These approaches include various combinations of skip-connections (U-Net & FPN), replacing bilinear upsample with unpooling (SegNet) or convolution transpose (DenconvNet), multi-constraints (MC-FCN), and additional boundary information (BR-Net). However, the increased representation capability and the complexity of the models usually lead to the overfitting of training data, especially for small or biased datasets.

To avoid overfitting, approaches including early stopping, data augmentation, regularization, and ensemble learning, are widely adopted. Of them, owing to its ability to utilize the representation capability of different models, the ensemble learning approach shows better performance and generalization capability. However, ensemble learning is currently used for patch-based CNN architectures, but not for FCN-based architectures. Additionally, research on ensemble learning mainly focuses on adding numbers or trying different combinations of basic models. To our best knowledge, research on methods to discern better combinations of various FCNs in ensemble learning does not exist.

Thus, we design stacked fully convolutional networks (SFCNs), and propose a feature alignment method, which enhances the relations between basic models. For ensemble learning, if the predictions from two models are completely different (in extreme cases, one of them is all zeros, and the other is all ones), the ensemble result is just an average of both biased predictions that cannot yield better performances. Therefore, to have better results, the predictions of different models should contain a certain level of variety as well as similarity. Compared with traditional ensemble learning approaches, the proposed method introduces an extra alignment loss to control similarity as well as consistency between features that are extracted separately from different basic models. In contrast to common segmentation loss, which is computed as the difference between a ground truth and its corresponding prediction, proposed alignment loss is computed among extracted features from stacked basic models. To make sure the alignment loss can be applied to a various number of basic models (*e.g.*, two models of FCN and U-Net, three models of FCN, U-Net, and FPN), the alignment loss is computed as the mean square error(MSE) between the maximum and minimum values of the extracted features (see details in Equation (2.1)). The value of alignment loss becomes zero when all the extracted features are similar. The value of the alignment loss reflects the consistency of extracted features. During iterations, the optimizer is required to update parameters to reduce the value of the weighted sum of segmentation loss and alignment loss. Thus, the optimized network is capable of generating a normalized prediction from various basic models. Through feature alignment framework, the SFCNs can achieve a balance of similarity and variety using different basic models, and improve performance.

Figure 2.2 presents the design of the proposed stacked fully convolutional networks(SFCN). The SFCN consists of two parts: (1) a framework for feature extraction using various fully convolutional networks and (2) a framework for feature alignment and output generation.

In the feature extraction framework, different numbers or combinations of FCN-based models are implemented to extract features from the same input image separately. For each FCN-based model, there are several universal operations and model specific layers. For universal operations, there are convolution, nonlinear activation, and subsampling operations. For backend models, various model specific layers, such as skip-connection (U-Net & FPN) and unpooling (SegNet), are included.

For universal operations, element-wise multiplication within the kernel is computed through the convolutional operation. The size of the kernel determines the receptive field and the computational efficiency of the convolution operation. Later, the output of convolution is handled by the rectified linear unit (ReLU) [70], which returns the original value if the value is larger than zero and sets values less than zero to zero. To accelerate network training, most models adopt

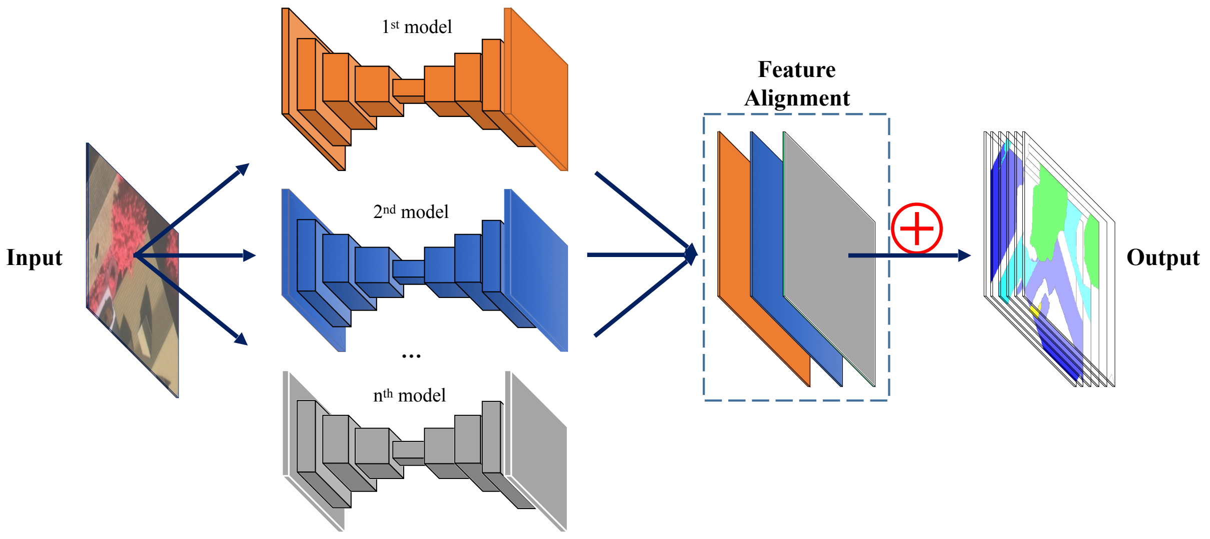


Figure 2.2: Proposed stacked fully convolutional networks(SFCN). The SFCN contains a framework for feature extraction using n number of fully convolutional networks(1^{st}, 2^{nd}, ..., n^{th} model), and a framework for feature alignment and final output generation.

batch normalization (BN) [71] layers before (*e.g.*, SegNet) or after non-linear activations (*e.g.*, FPN). To reduce the width and height of features, max-pooling [72] is chosen for subsampling in this study.

As for model specific layers, sequential bilinear upsampling [73] is commonly used for upsampling the width and height of the features. By contrast, SegNet backend uses unpooling which applies corresponding pooling indices of max-pooling to achieve upsampling. In FPN and U-Net backends, skip-connection, which concatenates two layers with consistent height and width across the channel axis, is applied between downward and upward layers.

In the framework for feature alignment and output generation, alignment loss that restricts the consistency of extracted features from various models and multi-class segmentation loss is computed sequentially.

- Alignment loss ($Loss_{align}$)

  Through the n^{th} FCN-based model, extracted features (denoted as Xn) with size of $W \times H \times D$ are generated. W and H are consistent with the height and width of the input. The value of D is the same as the number of classes of land covers. The maximum and minimum value for each position from the 1^{st} to the n^{th} feature are computed. The final alignment loss ($Loss_{align}$) is calculated by the mean square error between the corresponding maximum and minimum values of all positions.

$$\begin{aligned} Xmax_{i,j,k} &= max(X1_{i,j,k}, X2_{i,j,k}, ..., Xn_{i,j,k}) \\ Xmin_{i,j,k} &= min(X1_{i,j,k}, X2_{i,j,k}, ..., Xn_{i,j,k}) \\ Loss_{align} &= \frac{1}{W \times H \times D} \sum_{i=1,j=1,k=1}^{W,H,D} (Xmax_{i,j,k} - Xmin_{i,j,k})^2 \end{aligned} \tag{2.1}$$

- Segmentation loss ($Loss_{seg}$)

From all extracted features $(X1, X2, ..., Xn)$, the final output/prediction(Y) of the network is computed by taking the average value of all features. Then, the binary cross entropy [74], which calculates the difference between ground truth(G) and its corresponding prediction, is used as segmentation loss$(Loss_{seg})$. The calculation can be formulated as

$$Y = \frac{1}{N}\sum(X1, X2, ..., Xn)$$
$$Loss_{seg} = -\frac{1}{W \times H \times D}\sum_{i=1,j=1,k=1}^{W,H,D} g_{i,j,k} \times \log(y_{i,j,k}) + (1 - g_{i,j,k}) \times \log(1 - y_{i,j,k}) \tag{2.2}$$

Where $y_{i,j,k}$ and $g_{i,j,k}$ represent the (i,j,k) element of model output(Y) and ground truth (G). The value of $y_{i,j,k}$ is the predicted probability of the pixel category.

Therefore, the total loss of the network can be formulated as

$$\mathit{Loss}_{final} = \mathit{Loss}_{seg} + \lambda \times \mathit{Loss}_{align} \tag{2.3}$$

Where λ is the weight of the alignment loss (Loss_{align}). By controlling the value of λ, we are able to adjust the balance between Loss_{align} and Loss_{seg}.

During iterations, Adam optimizer [75] will minimize Loss_{final} to driven proposed network to generate pixel-to-pixel predictions for multi-label land-cover segmentation.

2.1.2.1 Network Specification

Three classic FCN-based architectures, including FCN-8s, U-Net, and FPN, are chosen as the basic models. All these models are implemented by Geoseg [2] using PyTorch (`https://pytorch.org/`, version=0.4.1) as backend.

The number and the size of convolutional kernels have a significant impact on model performance. To minimize their effect, basic models used in this research are implemented with the consistent number of kernel size at corresponding layers(see details in Figure 2.3).

2.1.2.2 Model Setup

To analyze the importance or significance of the proposed alignment loss, four versions of stacked fully conventional networks (SFCNs) are setup. Three variants are utilizing different combinations of two basic models, and a variant utilizing all three basic models. The variants using two basic models, SFCN$_{f\&p}$, SFCN$_{f\&u}$ and SFCN$_{u\&p}$ consist of FCN-8s&FPN, FCN-8s&U-Net, and U-Net&FPN, respectively (as shown in Table 2.1). All combinations are separately trained with different values of λ ($\lambda \in$ [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]). In all experiments, the models are trained, cross-validated, and tested though the same dataset. To prevent random bias, each set of experiments is repeated five times. After removing the best and worst performances of each method, their average performance with the testing dataset is carefully evaluated.

Table 2.1: Network setting of stacked fully convolutional networks from FCN-8s, U-Net, and FPN.

Version	**No. of basic models**	**FCN-8s**	**U-Net**	**FPN**
SFCN$_{f\&p}$	2	$+$	$-$	$+$
SFCN$_{f\&u}$	2	$+$	$+$	$-$
SFCN$_{u\&p}$	2	$-$	$+$	$+$
SFCN$_{f\&u\&p}$	3	$+$	$+$	$+$

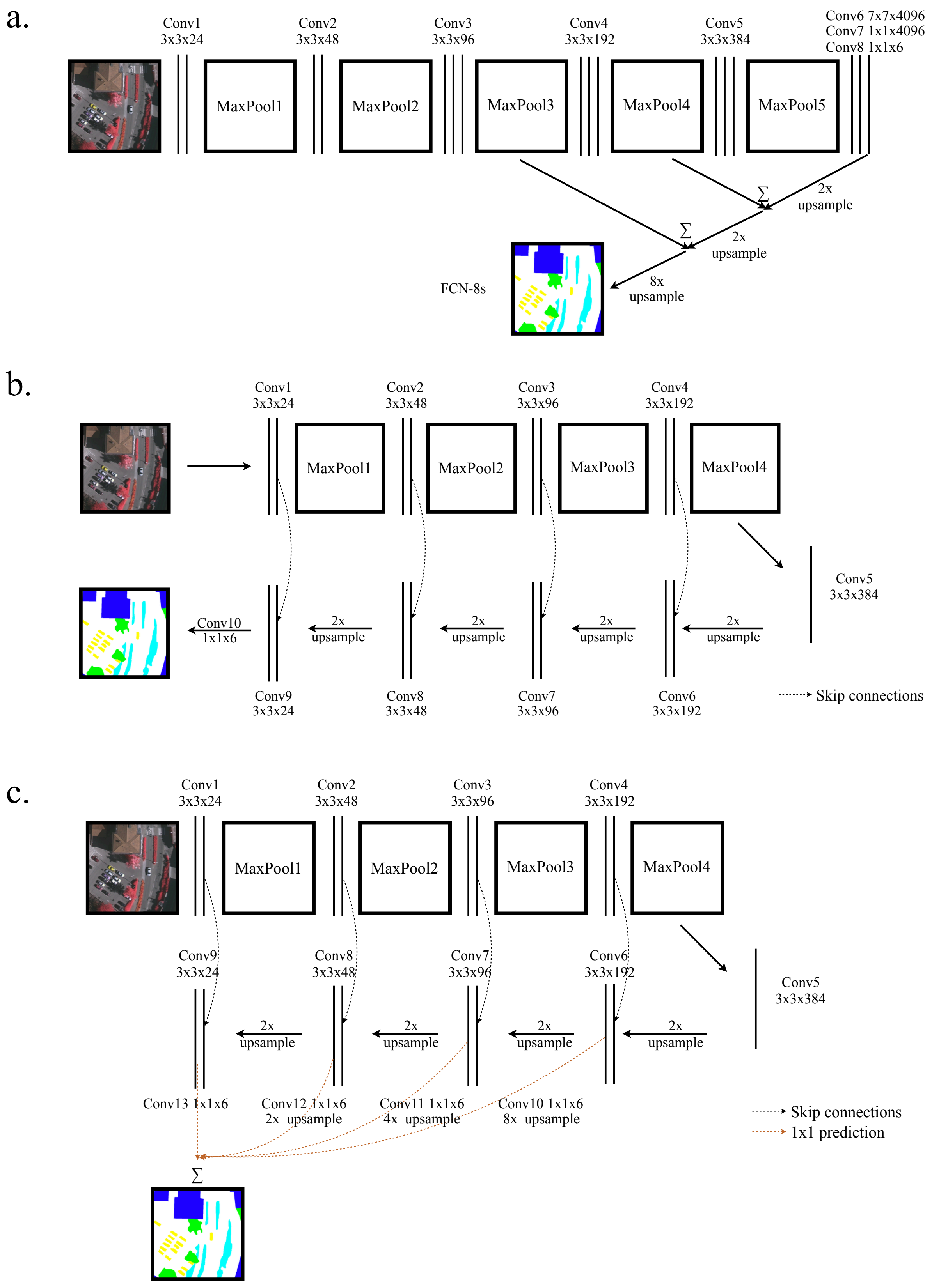


Figure 2.3: Specification of three basic models: (**a**) FCN-8s, (**b**) U-Net, and (**c**) FPN.

2.2 Line Feature Extraction

2.2.1 Scheme

Figure 2.4 presents the workflow for line feature extraction. The orthophotos, as well as their corresponding ground truth, are divided into two sets for training and testing. According to the building locations, a sliding window with a stride of 224 pixels is applied to the whole area to extract image patches with a size of 224 × 224 pixels. After data preprocessing, there are 16,635 and 14,834 image patches extracted from training and testing areas. Later, the image patches within the training area are shuffled and split into two groups that include training (70%), and cross-validating (30%). Through several cycles of training and cross-validation, the hyper-parameters are determined and optimized. Then, the predictions generated by the optimized model are further evaluated by the patches within the test set. For performance evaluations, we choose six commonly used evaluation metrics, namely, overall accuracy, precision, recall, jaccard index, f1-score, and kappa coefficient. These metrics are computed without post-processing operations [27, 67] for better estimation of experimental methods.

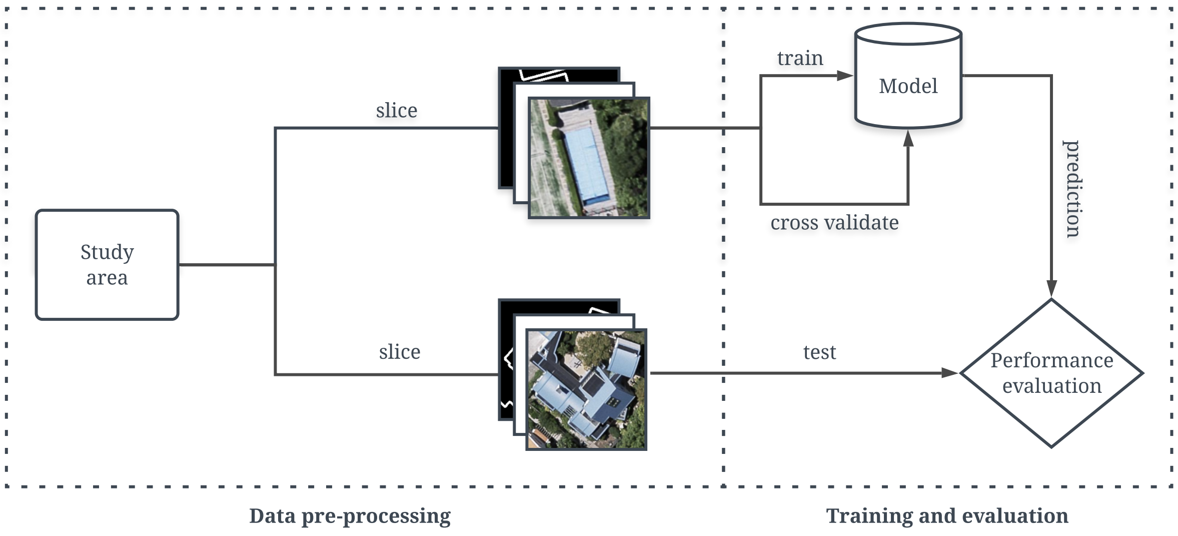


Figure 2.4: Experimental workflow for line feature extraction. The existing loss functions, as well as the proposed nearest feature selector, are trained and evaluated by 224 x 224 image patches extracted from original dataset.

2.2.2 Proposed Model

For efficient line feature extraction, we used the SegNet [36] for feature extraction and the nearest feature selector(NFS) for dynamic alignment between ground truth and prediction.

2.2.2.1 Nearest Feature Selector(NFS)

Figure 2.6 shows the mechanisms of the nearest feature selector(NFS). The ground truth is sliding over corresponding prediction along both X- and Y-axis to generated overlaps that used for

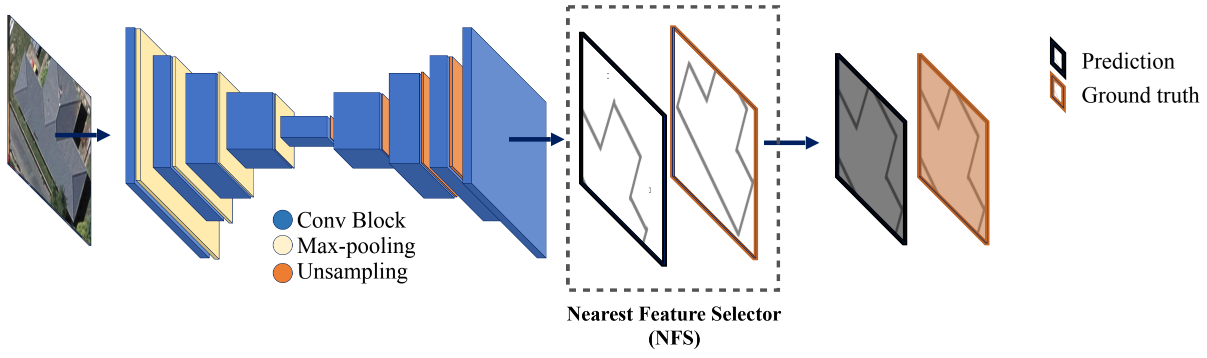


Figure 2.5: Proposed method for line feature extraction. The pipeline is consist of a SegNet for feature extraction and a nearest feature selector(NFS) for re-alignment.

similarity estimation. The overlap with the closest distance between ground truth and prediction are cropped for further loss computation. Since the NFS is computed dynamically, it can be seamlessly integrated into existing loss without further modification.

To evaluate the similarity of the overlaps, different measurements are adopted. For prediction and ground truth that contains a single channel, classic L1 distance is used. Thus, the distance of overlaps can be formulated as

$$\boldsymbol{D_{L1}} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} ||\boldsymbol{X}_{i,j} - \boldsymbol{Y}_{i,j}|| \tag{2.4}$$

where, $\boldsymbol{X}$ is the prediction and $\boldsymbol{Y}$ is the corresponding ground truth. The W and H are the width and height, respectively.

For prediction and ground truth that contains multiple channels, average cosine similarity along channels will be calculated. In such cases, the distance of overlaps can be formulated as

$$\boldsymbol{D_{cs}} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} \frac{\boldsymbol{X}_{i,j} \bullet \boldsymbol{Y}_{i,j}}{||\boldsymbol{X}_{i,j}|| \times ||\boldsymbol{Y}_{i,j}||} \tag{2.5}$$

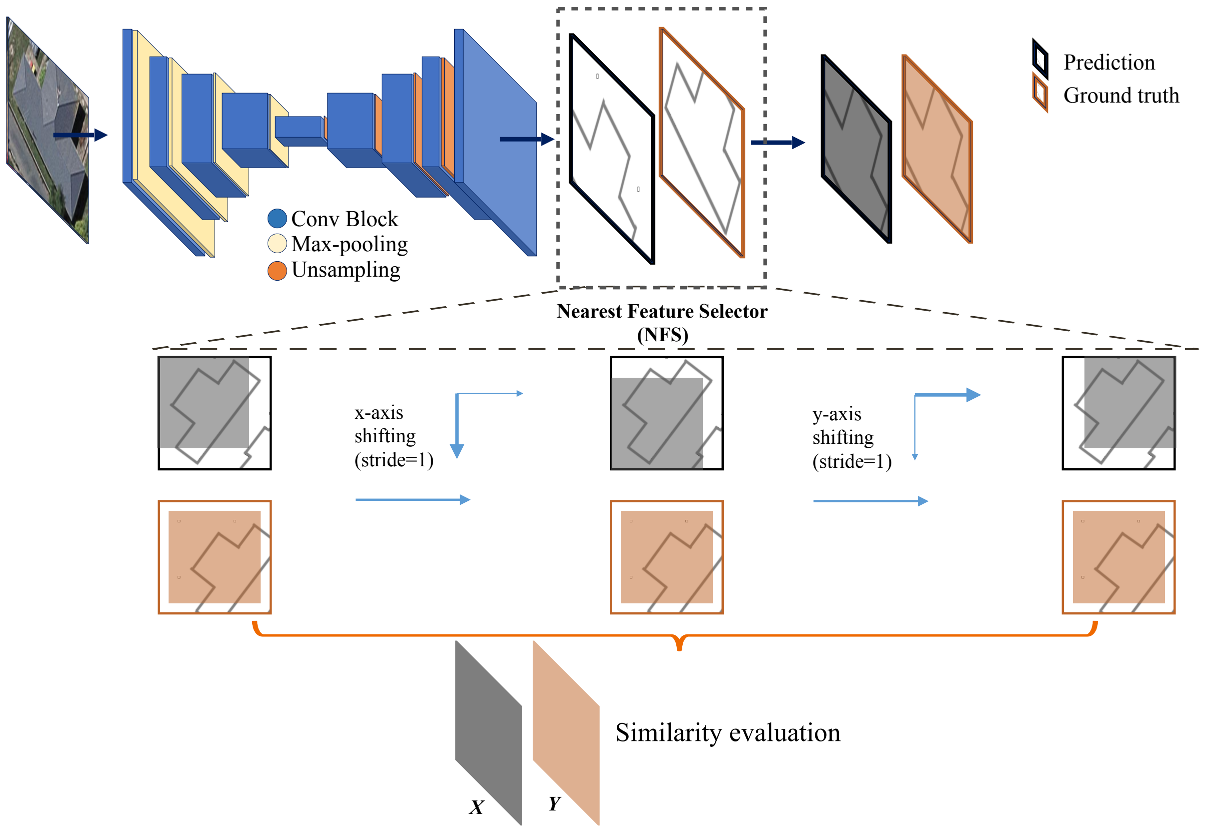

Conv Block
Max-pooling
Unsampling
Prediction
Ground truth
Nearest Feature Selector (NFS)
x-axis shifting (stride=1)
y-axis shifting (stride=1)
Similarity evaluation
X
Y

Figure 2.6: Overview of the nearest feature selector(NFS). The ground truth is sliding over prediction along X- and Y-axis to generated overlaps that used for similarity estimation.

2.3 Model Transfer

2.3.1 Scheme

Figure 2.7 presents the pipeline for model transfer. The super-resolution model, colorization network as well as the extraction model are trained on aerial images that with higher resolution and RGB color. The trained models, including the super-resolution model and colorization network, are applied to panchromatic low-resolution satellite images to generated high resolution, colorized satellite images. Finally, the quality of the generated images is evaluated by the extraction model trained on aerial images. The models are both trained in the training area of the Tokyo dataset(see Figure 3.4).

Similar to the polygon feature extraction scheme, a sliding window with a stride of 224 pixels is applied to the training area to extract image patches with a size of 224 $\times$ 224 pixels. Later, the image patches within the training area are shuffled and split into two groups that include training (70%) and cross-validating (30%). The number of samples in training and cross-and validation are 1,578 and 677, respectively. Through several cycles of training and cross-validation, the hyper-parameters are determined and optimized. Then, the extraction accuracy of the model is evaluated by the testing area.

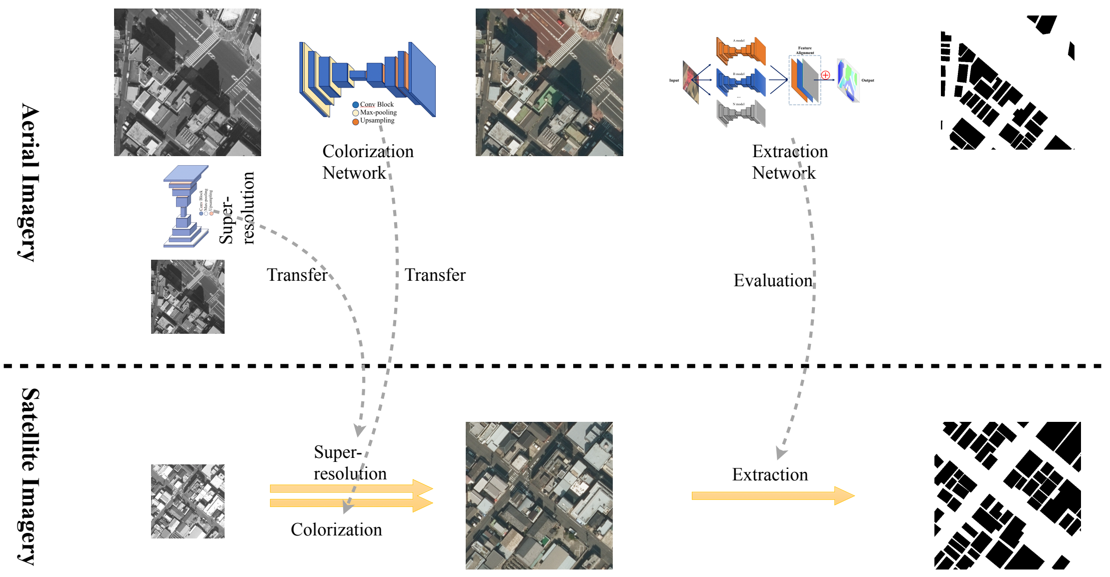


Figure 2.7: Experimental workflow of model transfer. The super-resolution model, colorization network and extraction model, are firstly trained by image patches extracted from training area of aerial images. Later, these models are applied to panchromatic satellite image to generated high-resolution, color image for further extraction evaluation.

2.3.2 Super-resolution Model

To achieve an HR image, a high-performance super-resolution model termed efficient sub-pixel convolutional neural network (ESPCN) [76] is selected. The ESPCN model is trained and optimized by image patches extracted from the training area of aerial imagery(see Figure 3.4).

2.3.3 Colorization Network

As shown in Figure 2.8, the proposed colorization network(ColorNet) is consist of one encoder and one decoder. The encoder follows the design of classic ResNet-18 [77] using sequential basic residual blocks and max-pooling layers. The parallel decoders share the identical architecture except for the final prediction layer. Similar to the encoder, the decoder applies sequential deconvolutional layers [35], residual blocks, and skip connections [34] to refine to original height and width gradually. To avoid interference within batch samples, batch normalization layers [71] are replaced by instance normalization [78].

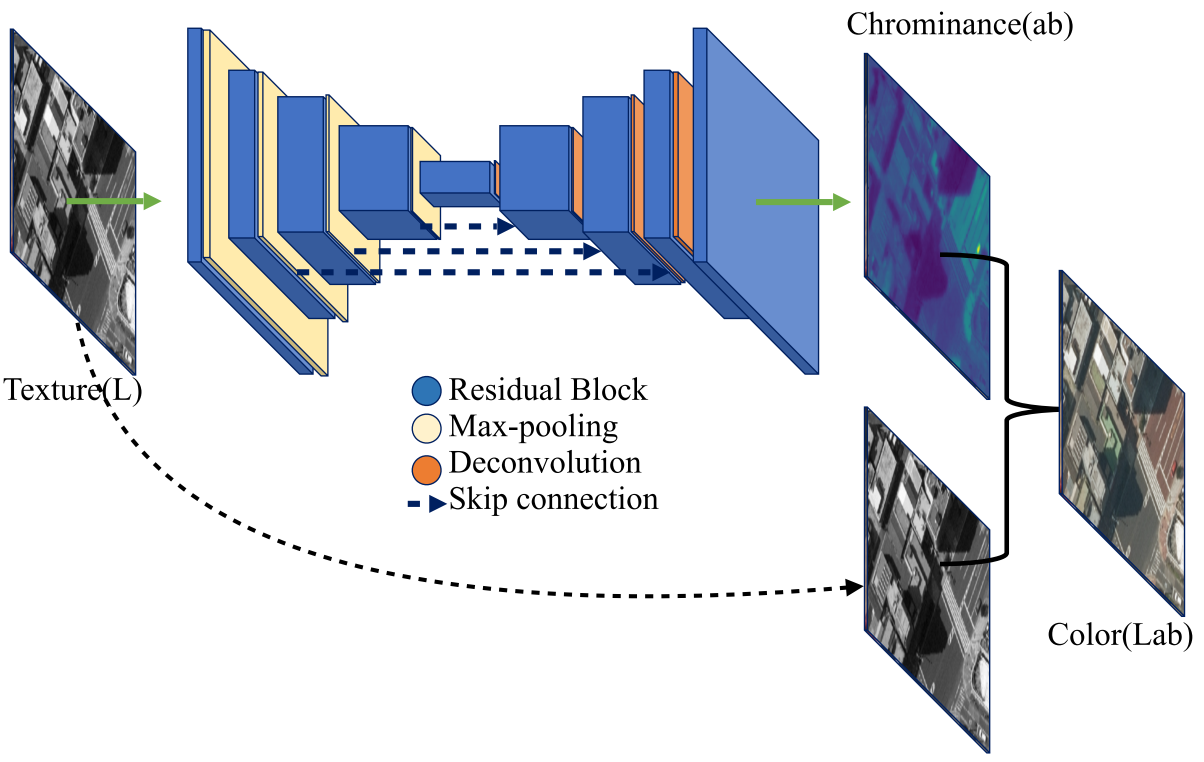


Figure 2.8: Network architecture of the Colorization Network(ColorNet).

2.3.4 Extraction Networks

As in Section 2.1, we trained stacked fully conventional network (SFCN) on testing datasets from an aerial image and then tested with generated high-resolution, colorized satellite images.

Chapter 3

Materials

Contents

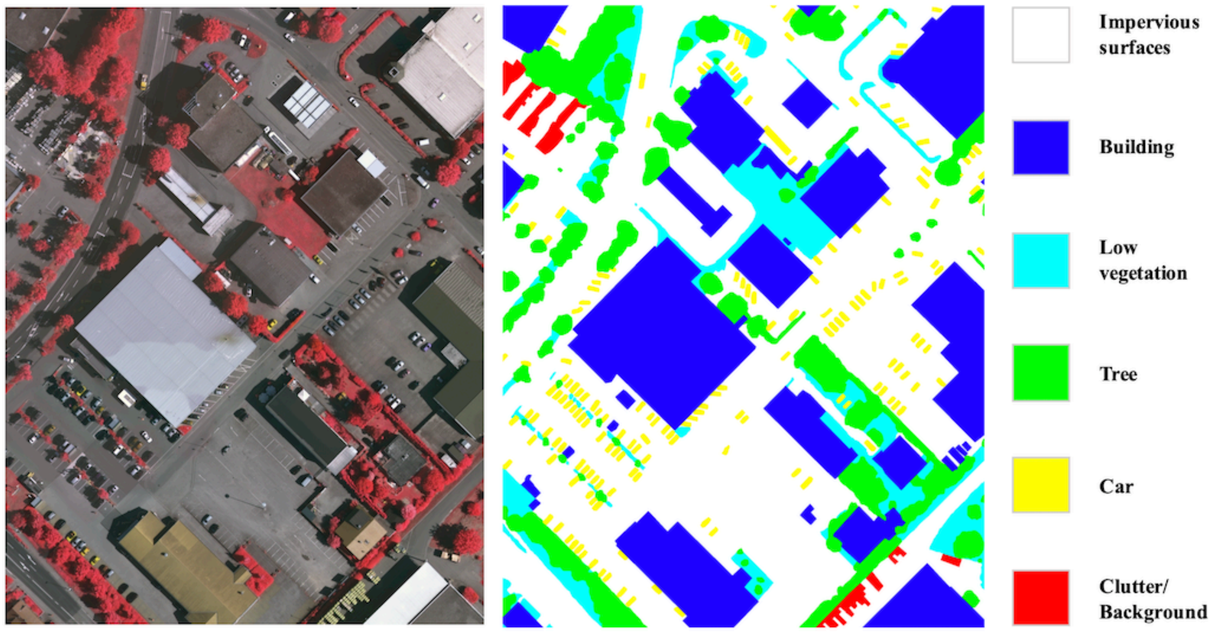
3.1 Datasets

3.1.1 Vaihingen Dataset

For estimating the effectiveness of the designed SFCNs and proposed feature alignment framework, we conduct our polygon feature extraction experiments on ISPRS 2D semantic labeling dataset–Vaihingen dataset.The dataset is an open benchmark, which is available online (`http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html`).

Within the Vaihingen dataset, there are 33 tiles, including 16 tiles for training and 17 tiles for testing. Only the tiles used for training are provided with images of annotated ground truth. The size of each tile ranges from 1388 x 2555 to 2006 x 3007 pixels. The ground sampling distance (GSD) of orthophoto is about 9 cm.

As shown in Figure 3.1, each tile of the dataset contains an orthophoto and its corresponding annotated ground truth. The orthophoto is an 8-bit image with three bands, which correspond to the near-infrared, red, and green bands delivered by the camera. The image of annotated ground truth utilizes six different colors to represent land-covers of impervious surfaces, buildings, low vegetation, trees, cars, and clutter/background (see color map in Table 3.1).



(a) Orthophoto (b) Ground truth (c) Legend

Figure 3.1: Example of Vaihingen 2D semantic labeling dataset. (**a**) true orthophoto, (**b**) annotated ground truth, and (**c**) legend. The ground truth contains six types of land-covers.

3.1.2 New Zealand Dataset

To evaluate the performance of different methods, a study area that covers 32 km^2 in Christchurch, New Zealand, is chosen for this study. The aerial image dataset and corresponding building outlines (polygons in .shp format) are downloaded from Land Information of New Zealand

Table 3.1: Reference of color map of the Vaihingen dataset.

Land-covers	RGB Values
Impervious surfaces	[255, 255, 255]
Building	[0 , 0, 255]
Low vegetation	[0 , 255, 255]
Tree	[0 , 255, 0]
Car	[255, 255, 0]
Clutter/Background	[255, 0, 0]

(`https://data.linz.govt.nz/layer/53413-nz-building-outlines-pilot/`). The spatial resolution of the aerial images is 0.075 m. The original images are captured during the flying seasons of 2015 and 2016. Later, they are converted into orthophotos and divided into tiles by the provider. The size of each tile is 3200 × 4800 pixels (240 × 360 m^2). Prior to conducting our experiments, we merge the 370 tiles within the study area into a single mosaic. Additionally, for accurate roof segmentation, we manually adjust vectorized building outlines to ensure that all building polygons are strictly aligned with their corresponding roofs.

As shown in Figure 3.2, the study area is primarily covered by residential or manufacturing buildings with sparsely distributed patches of grassland. Before conducting our experiments, the study area is evenly divided into two areas for training (Figure 3.2, left) and testing (Figure 3.2, right).

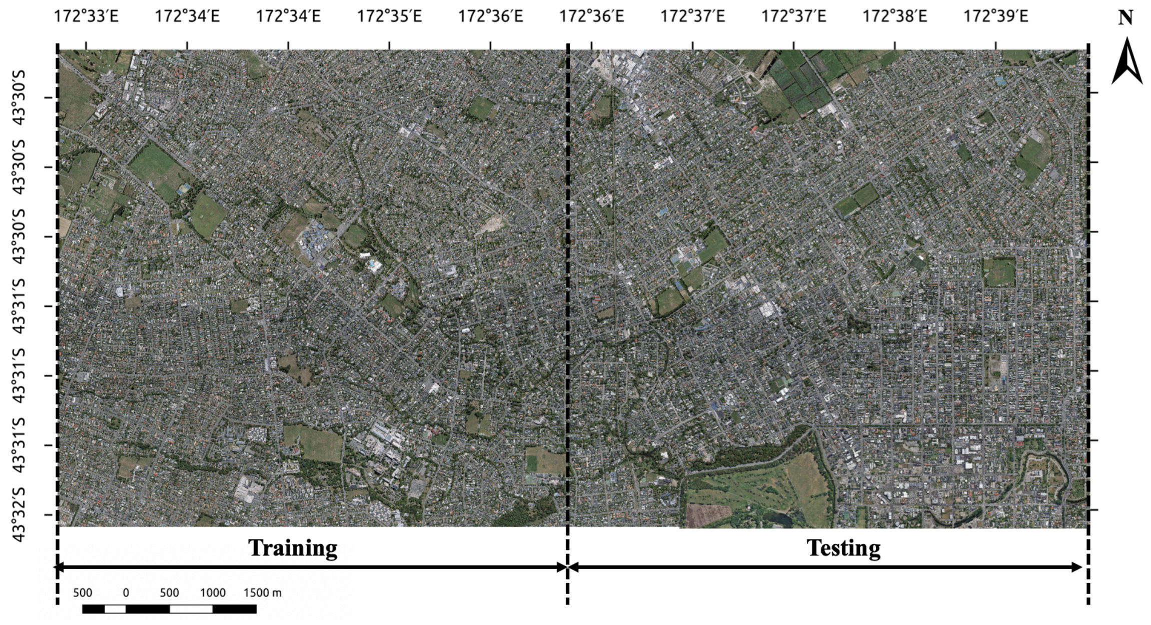


Figure 3.2: Aerial imagery of the study area ranging from $172°33'E$ to $172°40'E$ and $43°30'S$ to $43°32'S$.

From training and testing areas, 17,228 and 14,952 patches were extracted for experiments.

The size of the patch is 224x224 pixels. As shown in Figure 3.3, within each patch, there are buildings located in the central area.

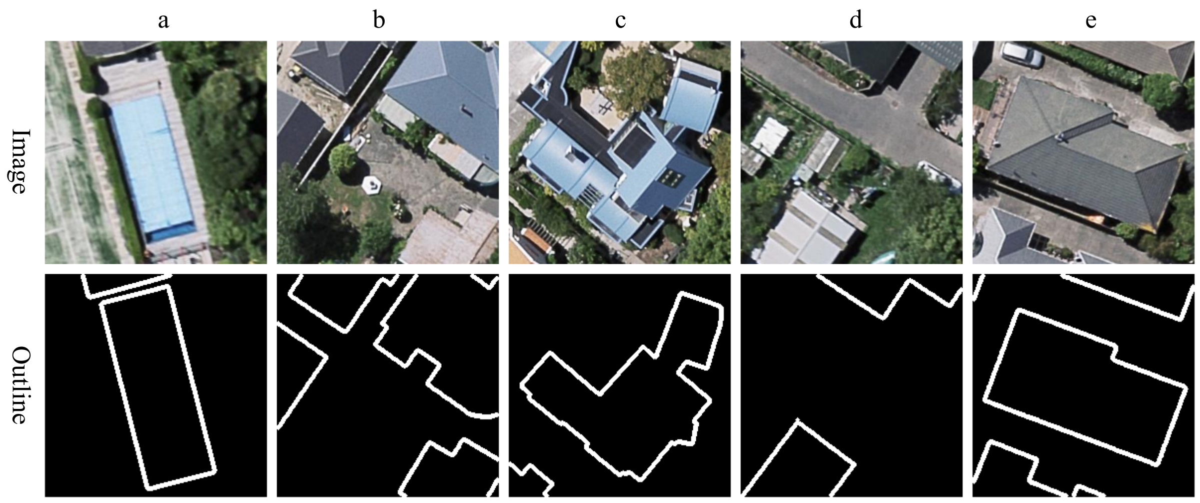


Figure 3.3: Sample patches of RGB input and corresponding edge.

3.1.3 Tokyo Dataset

For transfer learning experiments, a study area that covers 6 km^2 in Tokyo, Japan, is chosen for this study. The RGB aerial images and corresponding panchromatic satellite are provided by NTT GEOSPACE(`https://www.ntt-geospace.co.jp/`). The spatial resolutions of the aerial and satellite images are 0.16 m and 0.5 m, respectively. Both images are captured during the flying seasons of 2016. Even they are captured from the same location and similar period, they are misalignments with tens of pixels or even hundreds of pixels. Note that a typical building occupies about (50-200)^2 pixels, annotations for the aerial and satellite images should be prepared separately. The annotations for aerial images are come from NTT GEOSPACE. In contrast, the annotations for satellite images are created by ourselves.

As shown in Figure 3.4 and Figure 3.5, the study area is covered mainly by residential or manufacturing buildings with sparsely distributed patches of grassland. Before conducting our experiments, the study area is evenly divided into two areas for training (left) and testing (right).

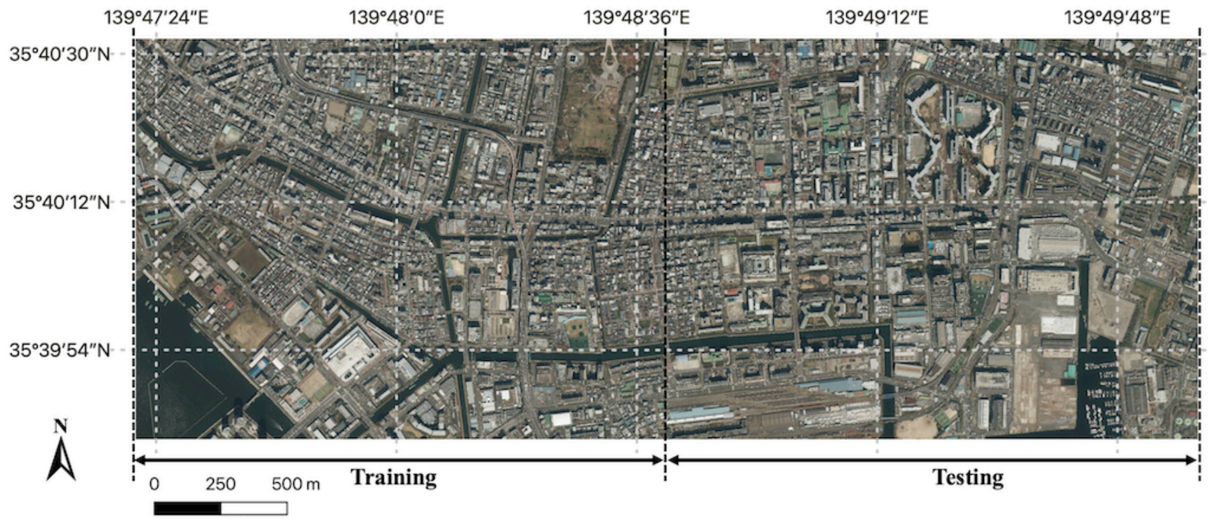


Figure 3.4: Aerial imagery of the study area ranging from $139°47'E$ to $139°50'E$ and $35°40'S$ to $35°39'S$.

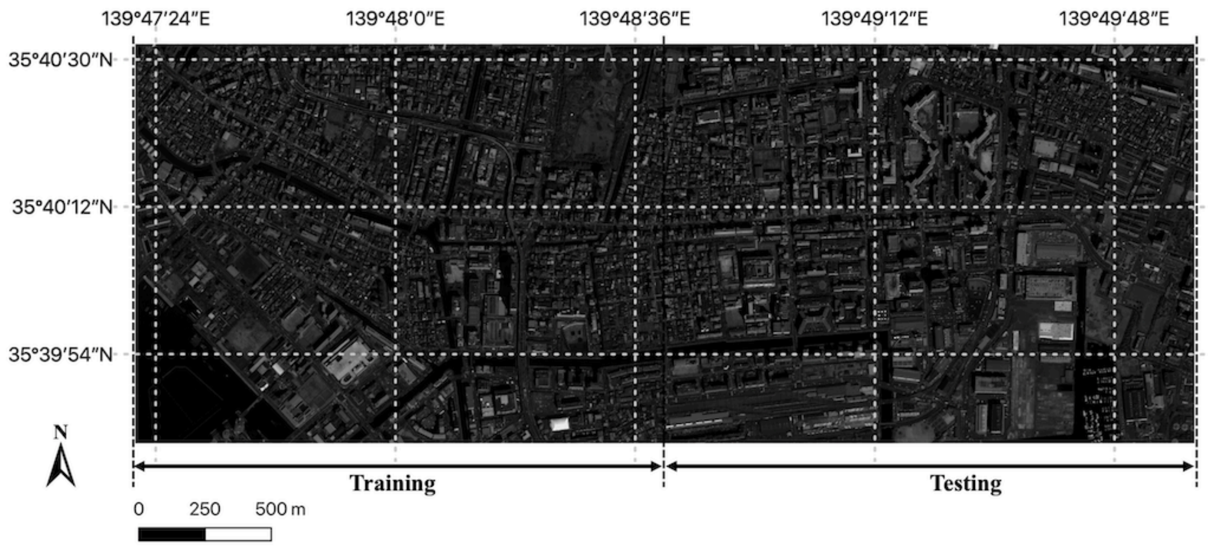


Figure 3.5: Satellite imagery of the study area ranging from $139°47'E$ to $139°50'E$ and $35°40'S$ to $35°39'S$.

3.2 Source Codes

3.2.1 Geoseg

Recently, deep learning algorithms, especially fully convolutional network based methods, have become very popular in the field of remote sensing. Generally, these methods achieve state-of-the-art accuracy or computational efficiency for their corresponding datasets. However, since these methods are trained and evaluated through different datasets, creating an in-depth comparison of the performance of various models is difficult. Additionally, although the datasets are open-access, the implemented models or algorithms are usually not revealed in detail by the authors.

Facing this problem, we introduce Geoseg, a computer vision package that is focused on implementing the state-of-the-art methods for automatic binary or multi-labels segmentation. The Geoseg package implements more than 9 FCN-based models including FCNs [32], U-Net [34], SegNet [36], FPN [79], ResUNet [80], MC-FCN [37], and BR-Net [49]. For in-depth comparisons, balanced and unbalanced evaluation metrics, such as precision, recall, overall accuracy, f1-score, jaccard index or intersection over union and kappa coefficients are implemented.

3.2.1.1 Code arrangement

Geoseg is built on top of PyTorch version 0.4.1. The whole package is organized as in Figure 3.6. There are six sub-directories : $data/, dataset/, checkpoint/, logs/, result/, and src/$. The dataset/ directory contains all samples for training, validating, and testing. The logs/ directory records learning curves and training and validation performance during model iterations. The src/ directory contains scripts implemented with various network architectures of the models. The quantitative and qualitative results are saved in the result/ directory.

In Geoseg, we implemented 9 FCN-based models according to the reports from their original papers. Since the original methods were implemented for various platforms and used for various sizes of input, Geoseg introduces a few modifications on several models for unification.

3.2.2 GeoSR

Similar to Geoseg, we present GeoSR, a PyTorch based computer vision package for deep learning based single-frame remote sensing imagery super-resolution. The GeoSR provides integrated modules, including data preprocessing, training, evaluation, and visualization.

3.2.2.1 Code arrangement

GeoSR is built on top of PyTorch version 0.4.1. The whole package is organized as in Figure 3.7. There are seven sub-directories: $src/, dataset/, logs/, model\_zoo/, archs/, utils/, and result/$. The src/ and dataset/ directory contains all samples for training, validating, and testing. The logs/ directory records learning curves and training and validation performance during model iterations. The model_zoo/ directory contains trained models at every experiment. The archs/ directory implements different network architectures. The utils/ directory implements scripts for handling datasets, running instructions, evaluation metrics, and visualization tools. The visualization results are saved in the result/ directory.

```
Geoseg
  ├── data/
  │   └── original image tiles
  ├── dataset/
  │   └── image&mask slices from data
  ├── checkpoint/
  │   └── pre-trained models
  ├── logs/
  │   ├── curve
  │   └── raw
  │   └── snapshot
  │   speed.csv
  ├── result/
  │   └── quantitative & qualitative result
  ├── src/
    ├── __init__.py
    ├── models
    │   └── network archs. FCNs, UNet, etc.
    ├── estrain.py
    ├── losses.py
    ├── metrics.py
    ├── runner.py
    ├── test.py
    ├── train.py
    └── vision.py
...
```

Figure 3.6: The code organization of the Geoseg package. The package implements model constructing, training, logging, evaluating, and result visualization modules.

```
GeoSR
├── src
│   └── data_dir
├── dataset
│   ├── train
│   ├── test
│   └── val
├── logs
│   ├── curve
│   ├── raw
│   └── statistic
├── model_zoo
│   └── trained_model
├── archs
│   ├── blockunits.py
│   ├── drcn.py
│   ├── espcn.py
│   ├── fsrnn.py
│   ├── rednet.py
│   ├── srcnn.py
│   ├── srdensenet.py
│   └── vdsr.py
├── utils
│   ├── combiner.py
│   ├── extractor.py
│   ├── loader.py
│   ├── metrics.py
│   ├── preprocessor.py
│   ├── trainer.py
│   ├── tester.py
│   └── vision.py
├── result
│   ├── raw
│   └── generate
│       ├── combined
│       ├── figures
│       └── tables
├── main.py
│
...
```

Figure 3.7: The code organization of the GeoSR package. The package implements model constructing, training, logging, evaluating, and result visualization modules.

3.3 Devices

- **Local PC.**The 64-bit Ubuntu 16.04 LTS system equipped with an NVIDIA GeForce GTX 1070 GPU (`https://www.nvidia.com/en-us/geforce/products/10series/geforce-gtx-1070-ti/`) with 8 GB of memory.
- **Cloud Server**.The Sakura Internet Server(`https://www.sakura.ad.jp/`) equipped with one NVIDIA Tesla V100 GPU (`https://www.nvidia.com/en-us/data-center/tesla-v100/`) and installed with 64-bit Ubuntu 16.04 LTS.

Table 3.2: Specifications of local PC and cloud server used for experiment.

Device	**Model**	**CPU**	**GPU**	**RAM**
Local PC	ASUS ROG G20CB	Core i5 6400	GTX 1070, 8 GB	16 GB
Cloud Server	Sakura Koukaryoku	Xeon CPU E5-2623	4x Tesla V100, 16 GB	128 GB

Chapter 4

Results

Contents

4.1 Polygon Feature Extraction

Three well-known FCN-based methods, namely, FCN-8s, U–Net, and FPN, are chosen for basic models in this study. Four SFCN models composed from three basic models (refer to Table 2.1) are trained separately with various weight (λ) of alignment loss ($Loss_{align}$). All experiments are performed on the same dataset and processing platform.

Three commonly used balanced metrics, including f1-score, jaccard index, and kappa coefficient, are selected for quantitative evaluation.

4.1.1 Quantitative Results

Three basic models (FCN-8s, U-Net, and FPN) and four combinations of ensemble models (*i.e.*, SFCNs) with/without optimal feature alignment(FA) are implemented and validated by the testing dataset. To prevent random bias, each set of experiments is repeated five times. After removing the best and worst performances of each method, their mean value and standard deviation (SD) of evaluation metrics are calculated.

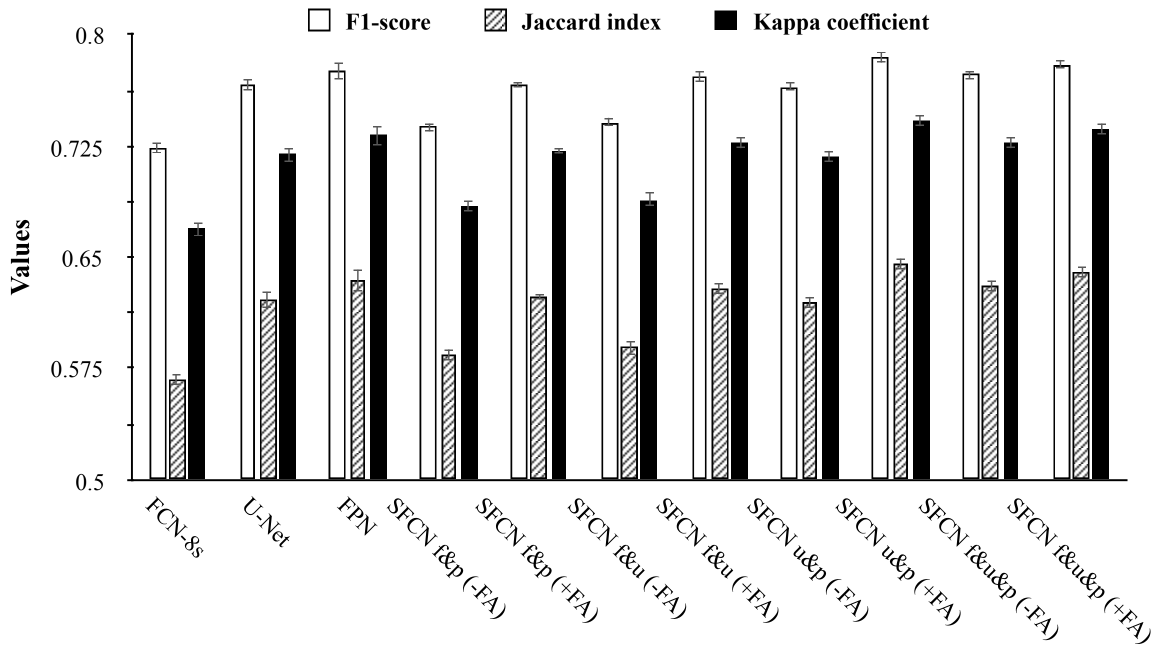
Figure 4.1a shows the relative performances of these models. Among three basic methods, the FPN shows the highest values for all evaluation metrics. For each combination of ensemble learning, methods with optimal feature alignment($+$FA) are generally better than the corresponding methods without optimal feature alignment($-$FA).

Figure 4.1b displays the corresponding mean and standard deviation(SD) values of evaluation metrics from different methods. Among four ensemble models without optimal feature alignment (SFCNs, $-$FA), SFCN$_{f\&u\&p}$($-$FA) shows the higher mean values than SFCN$_{u\&p}$($-$FA), SFCN$_{f\&u}$($-$FA), and SFCN$_{f\&p}$($-$FA) for all metrics. This observation indicates that an ensemble with more models can lead to better performance. For ensemble models using the same number of basic models (SFCN$_{f\&p}$, SFCN$_{f\&u}$, and SFCN$_{u\&p}$), a combination of U-Net and FPN (SFCN$_{u\&p}$) is better than a combination of FCN-8s and U-Net(SFCN$_{f\&u}$) or FCN-8s and FPN (SFCN$_{f\&p}$). Surprisingly, the best basic model (FPN) is better than the best ensemble model without feature alignment (SFCN$_{f\&u\&p}$, $-$FA). This result suggests that a simple ensemble of different basic models does not assure higher performance. As for the four ensemble models with optimal feature alignment (SFCNs, $+$FA), SFCN$_{u\&p}$($+$FA) shows the highest mean values for f1-score (0.785), jaccard index(0.646), and kappa coefficient(0.742). Ensemble methods with feature alignment showed higher values for all three evaluation metrics compared to their counterparts without feature alignment. Among all methods, the SFCN$_{u\&p}$($+$FA) methods achieved the highest performance.

The values for the standard deviation (SD) of three metrics from different models range from 0.001 to 0.008. When compared to the mean values, even the maximum value of SD (0.008) is not significant. Through independent t-test, except for SFCN$_{f\&u\&p}$, methods with optimal feature alignment showed significantly different values for all three evaluation metrics compared to their counterparts without feature alignment(see details in Table 4.1.

4.1.2 Qualitative Results

Figure 4.2 shows the prediction results on testing areas Tile-1, Tile-2, Tile-3, and Tile-4 of three basic models(FCN-8s, U-Net, and FPN), and optimized SFCNs. Generally, these models a.



b.

Method	F1-score		Jaccard index		Kappa coefficient	
	Mean	SD	Mean	SD	Mean	SD
FCN-8s	0.724	0.003	0.568	0.004	0.669	0.004
U-Net	0.766	0.004	0.621	0.006	0.719	0.005
FPN	0.776	0.006	0.635	0.008	0.732	0.007
SFCN$_{f\&p}$ (−FA)	0.738	0.003	0.584	0.004	0.685	0.004
SFCN$_{f\&u}$ (−FA)	0.741	0.003	0.589	0.004	0.689	0.004
SFCN$_{u\&p}$ (−FA)	0.765	0.006	0.619	0.007	0.718	0.006
SFCN$_{f\&u\&p}$ (−FA)	0.773	0.006	0.630	0.007	0.727	0.006
SFCN$_{f\&p}$ (+FA)	0.767	0.002	0.623	0.002	0.721	0.002
SFCN$_{f\&u}$ (+FA)	0.772	0.003	0.629	0.004	0.727	0.004
SFCN$_{u\&p}$ (+FA)	**0.785**	0.004	**0.646**	0.005	**0.742**	0.005
SFCN$_{f\&u\&p}$ (+FA)	0.780	**0.001**	0.640	**0.001**	0.736	**0.001**

Figure 4.1: Comparison of performances of the basic models of FCN-8s, U-Net, and FPN as well as four SFCNs with/without feature alignment. (**a**) Bar chart for comparison of relative performances. (**b**) Table of mean value and standard deviation (SD) of the performance comparison of these methods. For each evaluation metric, the highest mean values and lowest SD are highlighted in **bold**.

Table 4.1: Result of independent t-test of four SFCNs under with/without feature alignment. The p-value is the probability that SFCN has the same performances at both with and without feature alignment.

Group	**F1-score**		**Jaccard index**		**Kappa coefficient**	
	t-value	**p-value**	**t-value**	**p-value**	**t-value**	**p-value**
SFCN$_{f\&p}$ (+FA vs.−FA)	14.438	0.0001	16.546	0.0001	15.123	0.0001
SFCN$_{f\&u}$ (+FA vs.−FA)	11.750	0.0003	11.558	0.0003	11.144	0.0004
SFCN$_{u\&p}$ (+FA vs.−FA)	5.350	0.0059	5.329	0.0060	5.210	0.0065
SFCN$_{f\&u\&p}$ (+FA vs.−FA)	2.234	0.0892	2.271	0.0856	2.415	0.0732

could correctly segment the major parts of different land-covers from the original aerial images. The FCN-8s model tends to misclassify low vegetation as trees (*e.g.*, red rectangle in column 2, Tile 1), and the border area of buildings is usually broken (*e.g.*, the red rectangle in column 2, Tile 3). The result generated by U-Net is unable to discriminate between roads and buildings (*e.g.*, red rectangle in column 3, Tile 1 or row 3, Tile 2). The FPN model is generally better than FCN-8S and U-Net. However, trees and roads are misclassified as buildings (*e.g.*, the red rectangle in column 4, Tile 3). Among SFCN models, results generated from SFCN$_{f\&p}$ and SFCN$_{f\&u}$ tend to miss the buildings in the corner area (*e.g.*, the red rectangles in column 5, Tile 4 and column 5, Tile 4). The SFCN$_{f\&u\&p}$ model outperforms SFCN$_{f\&p}$ and SFCN$_{f\&u}$. However, there are misclassified holes within large buildings (*e.g.*, the red rectangle in column 8, Tile 3). When compared to other methods, even with some misclassification (*e.g.*, the red rectangle in column 7, Tile 3), SFCN$_{u\&p}$ shows better performance in major areas.

4.1.3 Sensitivity Analysis

To investigate the significance of feature alignment, four stacked fully convolutional networks (*i.e.*, SFCNs) using sequential values of lambda ($\lambda \in$ [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]) are implemented and validated on the testing dataset. To prevent random bias, each set of experiments is repeated five times. After removing the best and worst performances of each method, their mean value and standard deviation (SD) of the evaluation metrics are calculated. Figure 4.3 and Table 4.2 present the trends and values of f1-score, jaccard index, and kappa coefficient over various λ of $Loss_{align}$.

Figure 4.3a shows the trend of performances over λ values of $Loss_{align}$ on SFCN$_{f\&p}$. As the value of λ increases, the values of three metrics improve. Even there is small difference in standard deviation, the mean values of f1-score, jaccard index, and kappa coefficient of both cases under λ = 1.0 and λ = 0.8 are identical. The corresponding f1-score, jaccard index, and kappa coefficient are 0.767, 0.623, and 0.721, respectively. The best performance is achieved under value of λ = 1.0 and λ = 0.8. This result indicates that the introduction of feature alignment leads to better performance of the ensemble model. Figure 4.3b and c show the trend of performances on SFCN$_{f\&u}$ and SFCN$_{u\&p}$, respectively. When $\lambda \leq 0.8$, higher λ generally has higher value metrics. By contrast, while $\lambda \geq 0.8$, higher λ leads to weaker performances. In contract to Figure 4.3a–c, there is no significant change in the values of the metrics under various λ in Figure 4.3d, which implies that feature alignment has no significant effect on SFCN$_{f\&u\&p}$.

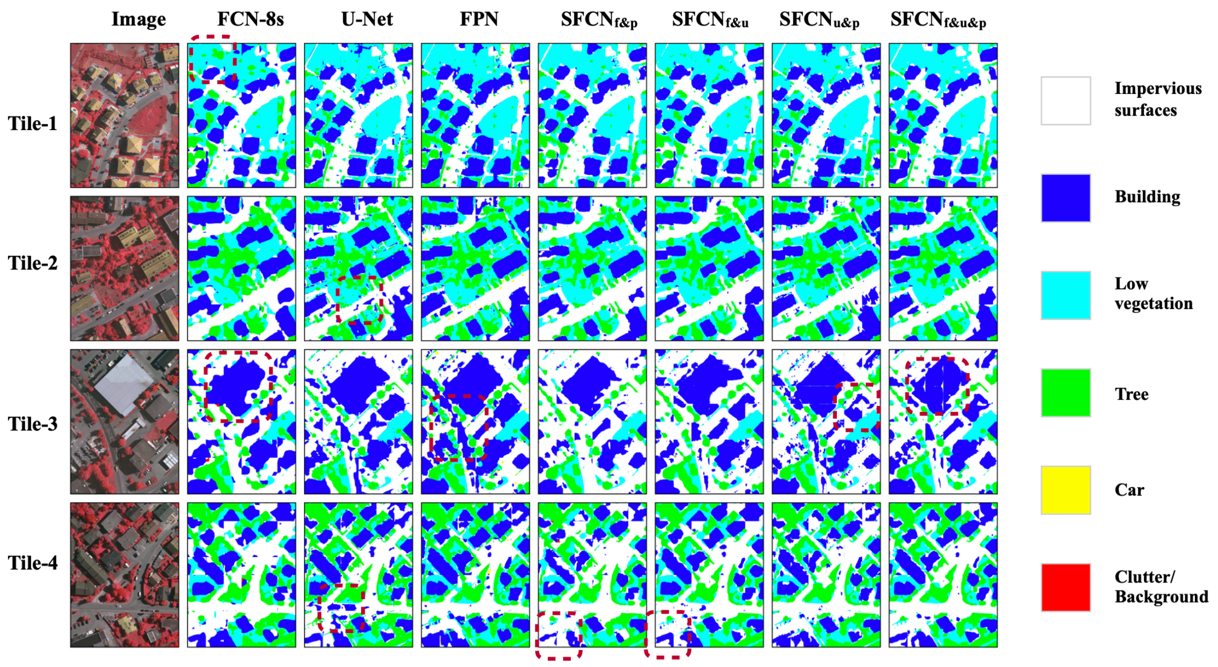


Figure 4.2: Segmentation results of FCN-8s, U-Net, and FPN and optimized SFCNs for testing areas including Tile-1, -2, -3, and -4. Predicted land-covers are represented with six colors.

Table 4.2 reveals the values of evaluations metrics f1-score, jaccard index, and kappa coefficient of four ensemble methods using λ in [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]. For SFCN$_{f\&p}$, the best performances are achieved at λ values 0.8 and 1.0. When compared with no feature alignment(*i.e.*,λ = 0.0), the values of f1-score, jaccard index, and kappa coefficient increase about 3.9% (0.767 vs 0.738), 6.7% (0.623 vs 0.584), and 5.3% (0.721 vs 0.685), respectively. For SFCN$_{f\&u}$, the best performance is achieved at λ value 0.8. When compared to no feature alignment, the highest values of f1-score, jaccard index, and kappa coefficient increase about 4.2% (0.772 vs 0.741), 6.8% (0.629 vs 0.589), and 5.5% (0.727 vs 0.689), respectively. Like SFCN$_{f\&u}$, the best performance of SFCN$_{u\&p}$ is at λ value 0.8. With comparison to the baseline λ = 0.0, the maximum increments of f1-score, jaccard index, and kappa coefficient reach 2.6% (0.785 vs 0.765), 4.4% (0.646 vs 0.619), and 3.3% (0.742 vs 0.718), respectively. By contrast to the above methods, the values of f1-score for SFCN$_{f\&u\&p}$ are almost identical (within [0.773, 0.780]). Under optimal feature alignment condition (*e.g.*, λ = 1.0), the values of jaccard index and kappa coefficient increase about 1.6% (0.640 vs 0.630) and 1.2% (0.736 vs 0.727), respectively. When compared to other methods (*e.g.*, SFCN$_{f\&u}$), the improvement caused by feature alignment of SFCN$_{f\&u\&p}$ is not so significant. The values for the standard deviation (SD) of the three metrics from different models range from 0.001 to 0.006. When compared to the mean values, even the maximum value of SD (0.006) is not significant.

4.1.4 Computational Efficiency

All experiments are implemented and tested on a Sakura Internet Server(`https://www.sakura.ad.jp/`) equipped with one NVIDIA Tesla V100 GPU (`https://www.nvidia.com/en-us/data-center/tesla-v100/`) and installed with 64-bit Ubuntu 16.04 LTS.

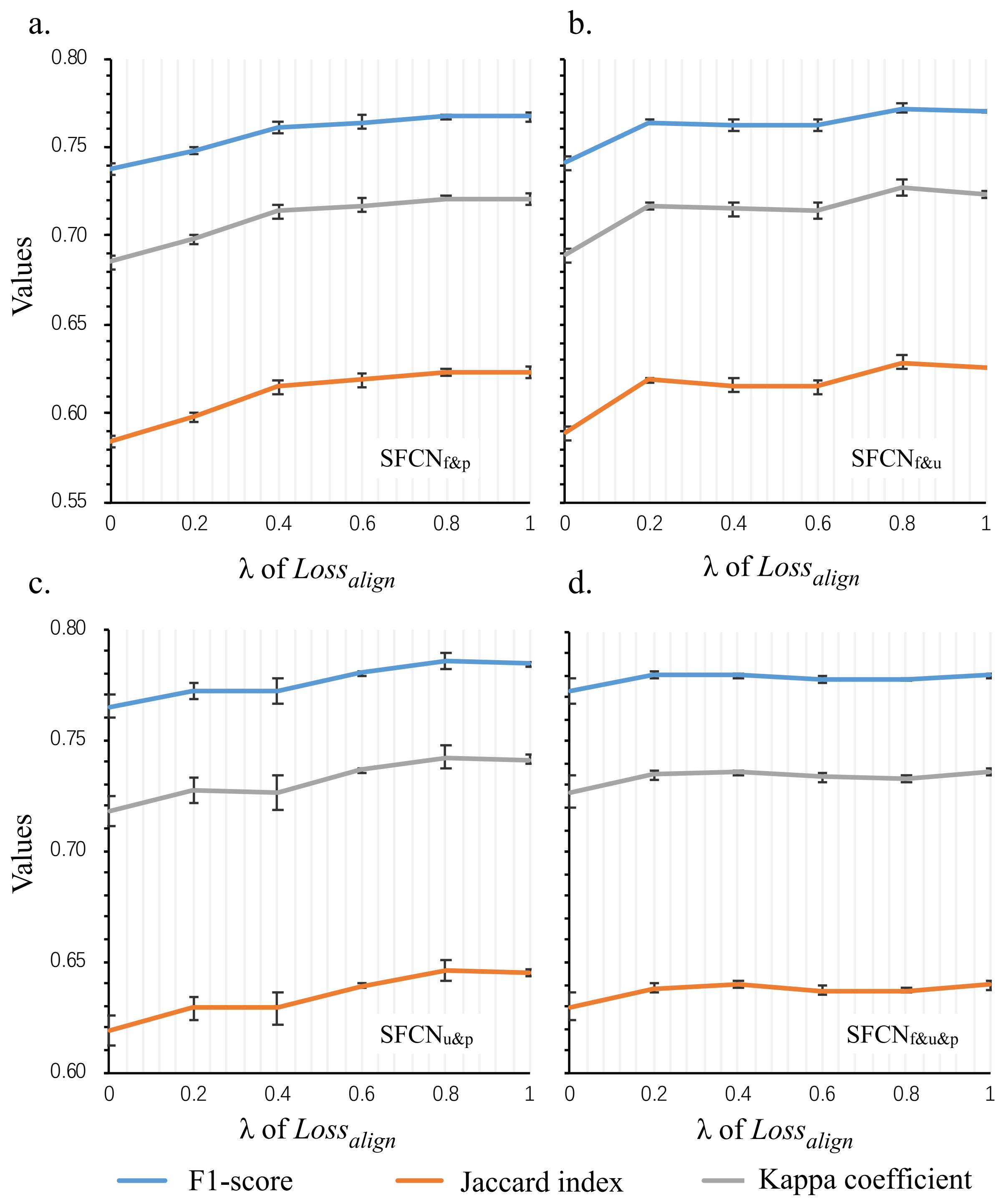


Figure 4.3: Trends of model performances of four SFCNs using lambda values in [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]: (**a**) performances of SFCN$_{f\&p}$ over lambda values; (**b**) performances of SFCN$_{f\&u}$ over lambda values; (**c**) performances of SFCN$_{u\&p}$ over lambda values; and (**d**) performances of SFCN$_{f\&u\&p}$ over lambda values.

Table 4.2: Table of model performances of four SFCNs under lambda values in [0.0, 0.2, 0.4, 0.6, 0.8, 1.0].For each evaluation metric, the highest mean values and lowest SD are highlighted in **bold**.

Method		**F1-score**		**Jaccard index**		**Kappa coefficient**	
Version	**λ-value**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
SFCN$_{f\&p}$	0.0	0.738	0.003	0.584	0.004	0.685	0.004
	0.2	0.748	**0.002**	0.598	0.003	0.698	0.003
	0.4	0.761	0.004	0.615	0.004	0.714	0.004
	0.6	0.764	0.004	0.619	0.004	0.717	0.004
	0.8	**0.767**	**0.002**	**0.623**	**0.002**	**0.721**	**0.002**
	1.0	**0.767**	0.003	**0.623**	0.003	**0.721**	0.003
SFCN$_{f\&u}$	0.0	0.741	0.003	0.589	0.004	0.689	0.004
	0.2	0.764	0.002	0.619	**0.002**	0.717	**0.002**
	0.4	0.762	0.003	0.616	0.004	0.715	0.004
	0.6	0.762	0.004	0.615	0.005	0.714	0.004
	0.8	**0.772**	0.003	**0.629**	0.004	**0.727**	0.004
	1.0	0.770	**0.001**	0.626	**0.002**	0.723	**0.002**
SFCN$_{u\&p}$	0.0	0.765	0.006	0.619	0.007	0.718	0.006
	0.2	0.772	0.004	0.629	0.006	0.727	0.005
	0.4	0.772	0.006	0.629	0.008	0.726	0.007
	0.6	0.780	**0.001**	0.639	**0.002**	0.736	**0.001**
	0.8	**0.785**	0.004	**0.646**	0.005	**0.742**	0.005
	1.0	0.784	**0.001**	0.645	**0.002**	0.741	0.002
SFCN$_{f\&u\&p}$	0.0	0.773	0.006	0.630	0.007	0.727	0.006
	0.2	**0.780**	0.002	0.638	0.002	0.735	0.002
	0.4	**0.780**	**0.001**	**0.640**	**0.001**	**0.736**	**0.001**
	0.6	0.778	0.002	0.637	0.002	0.734	0.002
	0.8	0.778	**0.001**	0.637	0.002	0.733	**0.001**
	1.0	**0.780**	**0.001**	**0.640**	0.002	**0.736**	0.002

Table 4.3: Comparison of the computational efficiencies of FCN-8s, U-Net, FPN and four ensemble fully conventional networks. For each column, the highest mean values and lowest SD are highlighted in **bold**.

Methods	Training FPS		Testing FPS	
	Mean	SD	Mean	SD
FCN-8s	41.4	**0.2**	67.1	0.3
U-Net	**59.4**	0.4	**75.4**	1.3
FPN	54.6	2.0	74.7	4.4
SFCN$_{f\&p}$	31.2	**0.2**	61.7	0.7
SFCN$_{f\&u}$	33.4	0.8	63.2	0.9
SFCN$_{u\&p}$	41.8	0.9	67.7	3.0
SFCN$_{f\&u\&p}$	27.2	**0.2**	57.6	**0.1**

To eliminate the effect of some hyperparameters, for all models, the size of batch and number of the iteration are fixed to 24 and 1000, respectively. The Adam stochastic optimizer, which is running at default setting (lr=2^{-4}, betas = [0.9, 0.999]), is used for training different models.

Table 4.3 shows the computing speeds in frames per second (FPS) of these methods. In training period, three basic models are processed at 41.4 FPS(FCN-8s), 59.4 FPS(U-Net), and 54.6 FPS(FPN), respectively. When compared to basic models, the ensemble methods are much slower. As the number of basic models increases (*e.g.*, 3 in SFCN$_{f\&u\&p}$vs 2 in SFCN$_{f\&p}$,SFCN$_{f\&u}$, and SFCN$_{u\&p}$), the training speed decreases. Even with the same number of basic models, because of the difference in model combination, the training speeds are different. Generally, a combination of fast basic models can form a faster ensemble model (*e.g.*, 41.8 FPS of SFCN$_{u\&p}$ vs. 31.2 FPS of SFCN$_{f\&p}$). In the testing period, these methods achieved 1.3–2.1 times the processing speed. Interestingly, SFCN$_{f\&u\&p}$ has the most significant performance difference (57.6 vs. 27.2, 2.1x) between the training and testing stages.

4.2 Line Feature Extraction

Four well-known loss functions, namely, L1, mean square error (MSE), binary cross-entropy (BCE) [74], and focal loss [81], are chosen in this study. These loss functions are trained neither with or without the nearest feature selector(NFS), separately. All experiments are performed on the same dataset and processing platform.

Three commonly used balanced metrics, including f1-score, jaccard index, and kappa coefficient, are selected for quantitative evaluation.

4.2.1 Quantitative Results

Figure 4.4 (a) shows the relative performances of different loss functions. Among all loss functions(*i.e.*, L1, MSE, BCE, and Focal), loss with the nearest feature selector(NFS) shows the higher values for all evaluation metrics.

Figure 4.4 (b) displays the corresponding values of evaluation metrics over various loss functions. Among four loss functions, no matter with or without NFS, Focal loss is generally better than BCE, MSE, and L1 loss. Over all conditions, L1 loss without NFS($\mathrm{L1} - \mathrm{NFS}$) shows the lowest values for all metrics. The best performance is achieved by Focal loss with NFS, showing 0.651 of f1-score, 0.490 of jaccard index, and 0.626 of the kappa coefficient. With all loss functions, the addition of NFS leads to significantly higher values of all evaluation metrics. The result indicates that the proposed NFS can effectively handle the slight misalignments from the annotation and gain more performance. Interestingly, on the weakest L1 loss, addition of NFS results in the most significant increments of three evaluation metrics. The increments of f1-score, kappa coefficient, and jaccard index reach 8.8%, 8.9%, and 9.8%, respectively.

4.2.2 Qualitative Results

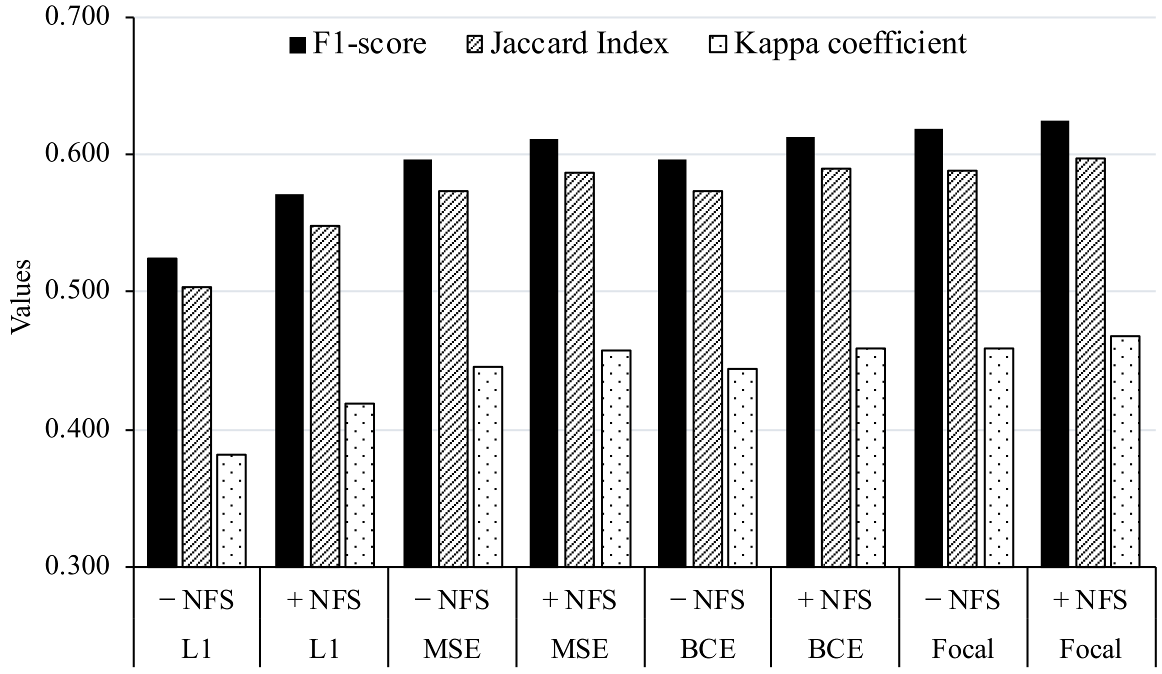
Figure 4.5 presents five representative groups of outline extracted by the L1 loss that either combine the nearest feature selector(NFS) or not. In general, addition of NFS shows better line extraction. With NFS, extracted lines are more intact(*e.g.*, a, b, and e) and accurate(*e.g.*, d). When training a model using L1 only($-$ NFS), the boundary areas are sometimes ignored(*e.g.*, c).

Figure 4.6 reveals five representative groups of outline extracted by mean square error(MSE) loss that with/without the nearest feature selector(NFS). Generally, additional NFS brings slightly better performance in the line extraction task. With the NFS, extracted lines contain fewer false positives(*e.g.*, a, and d) as well as breakpoints(*e.g.*, b, and e).

Figure 4.7 shows five representative groups of outline extracted by binary cross-entropy(BCE) loss that integrated with or without the nearest feature selector(NFS). Even without NFS, a model trained with BCE can successfully extract the significant part of building outlines. The addition of NFS shows slightly better line extraction around corners of the building (*e.g.*, d, and e) as well as areas shadowed by surrounding trees(*e.g.*, a, b, and c).

Figure 4.8 presents five representative groups of outline extracted by the focal loss that combines with or without NFS. Usually, focal loss along can recognize and extract the significant outline from aerial input(*e.g.*, b, c, and e). With additional NFS, generated lines have fewer false positives(*e.g.*, a, and d).

a.



b.

Loss	Condition	F1-score	Jaccard Index	Kappa coefficient
L1	− NFS	0.524	0.503	0.382
L1	+ NFS	**0.571**	**0.548**	**0.419**
MSE	− NFS	0.596	0.573	0.445
MSE	+ NFS	**0.611**	**0.587**	**0.458**
BCE	− NFS	0.596	0.573	0.444
BCE	+ NFS	**0.613**	**0.589**	**0.459**
Focal	− NFS	0.618	0.588	0.459
Focal	+ NFS	**0.624**	**0.597**	**0.468**

Figure 4.4: Performances of different losses which neither with or without of the nearest feature selector (NFS). (**a**) Bar chart for comparison of relative performances. (**b**) Table of performances under different loss functions. For each loss function, the highest values are highlighted in **bold**.

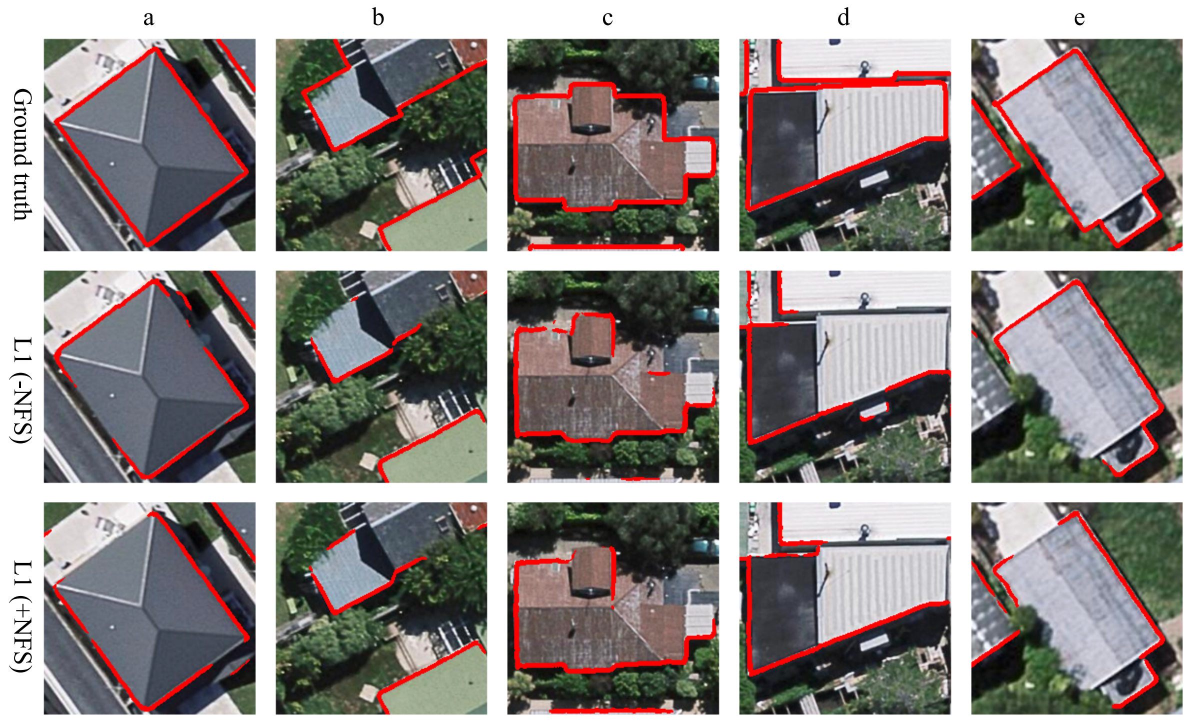


Figure 4.5: Representative results of outline extracted by L1 loss that with/without nearest feature selector(NFS). The background and red line represent aerial input and outline, respectively.

4.2.3 Computational Efficiency

All experiments are implemented and tested on a Sakura Internet Server(`https://www.sakura.ad.jp/`) equipped with 4x NVIDIA Tesla V100 GPU (`https://www.nvidia.com/en-us/data-center/tesla-v100/`) and installed with 64-bit Ubuntu 16.04 LTS. To eliminate the effect of some hyperparameters, for all models, the size of batch and number of the iteration are fixed to 24 and 10000, respectively. The Adam stochastic optimizer, which is running at default setting (lr=2^{-4}, betas = [0.9, 0.999]), is used for training different models.

Table 4.4 shows the computing speeds in frames per second (FPS) of these methods. Among all loss functions, the additional NFS leads to slightly longer precessing time during both training and testing period. However, the declines of the PFS are not significant.

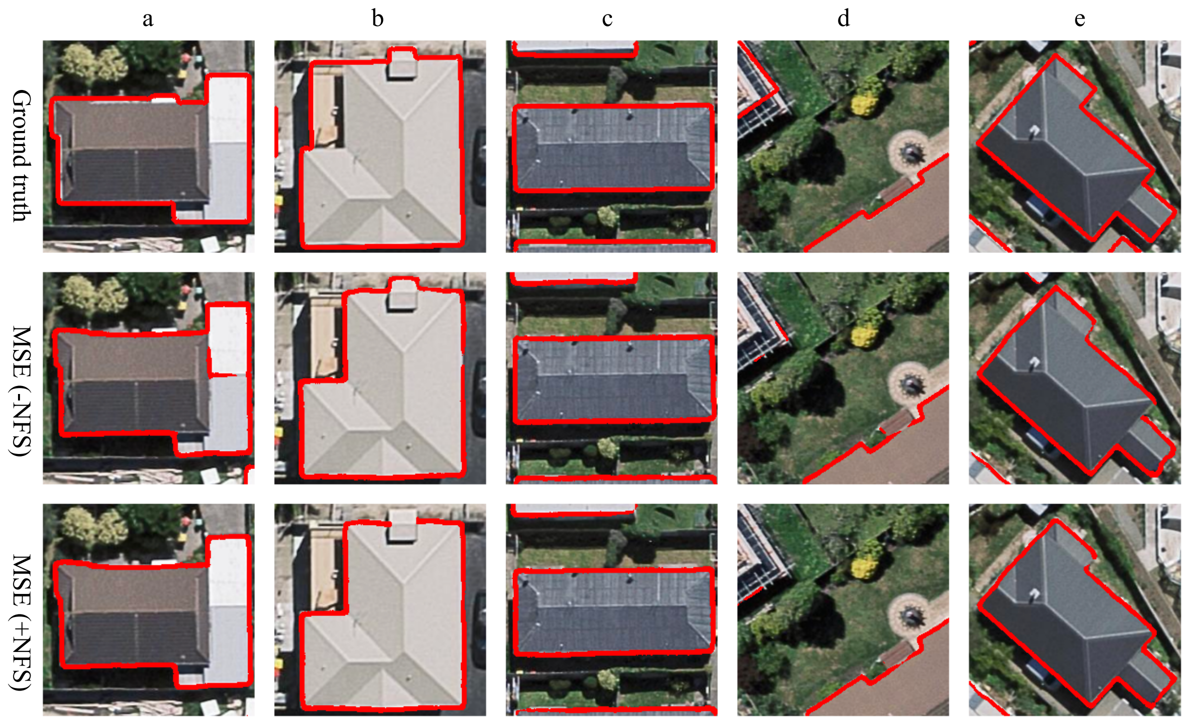


Figure 4.6: Representative results of outline extracted by MSE loss that with/without nearest feature selector(NFS). The background and red line represent aerial input and outline, respectively.

Table 4.4: Comparison of the computational efficiencies of different loss functions under conditions that with or without NFS.

Loss	**Condition**	**Training FPS**	**Testing FPS**
L1	− NFS	102.3	264.4
L1	+ NFS	98.5	236.1
MSE	− NFS	101.9	265.9
MSE	+ NFS	98.4	236.2
BCE	− NFS	102.1	266.8
BCE	+ NFS	98.7	236.6
Focal	− NFS	101.6	268.5
Focal	+ NFS	97.9	236.3

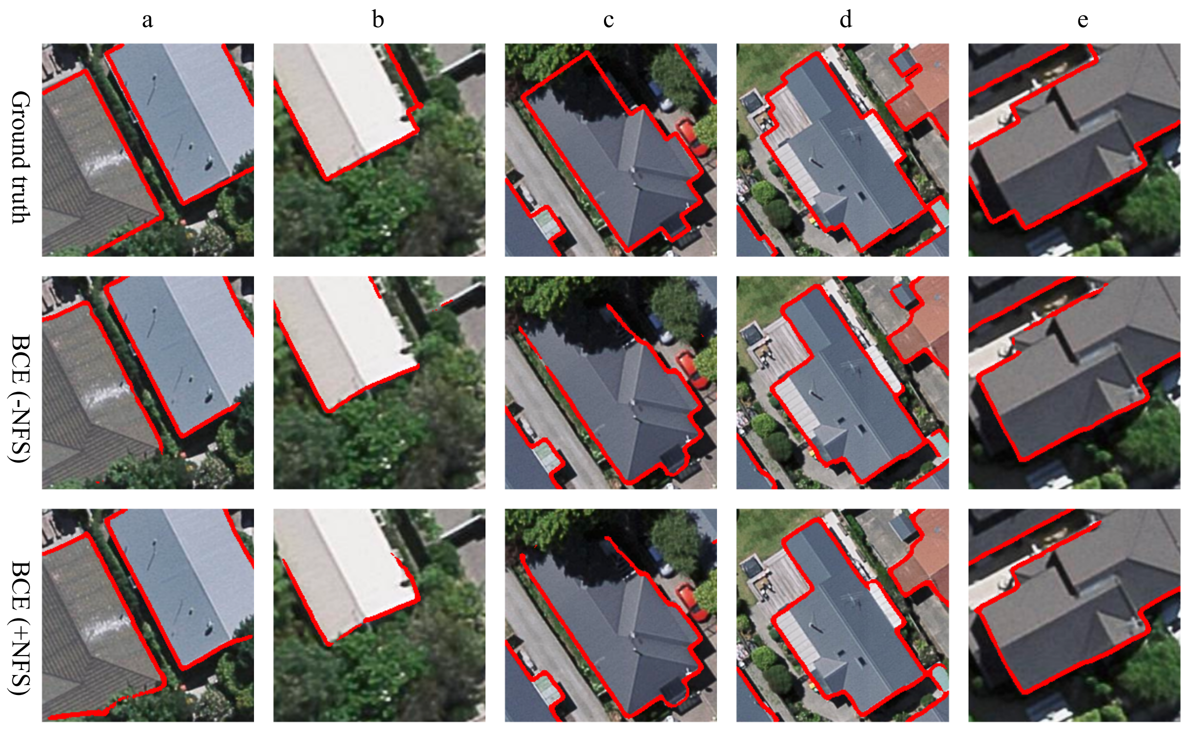


Figure 4.7: Representative results of outline extracted by BCE loss that with/without nearest feature selector(NFS). The background and red line represent aerial input and outline, respectively.

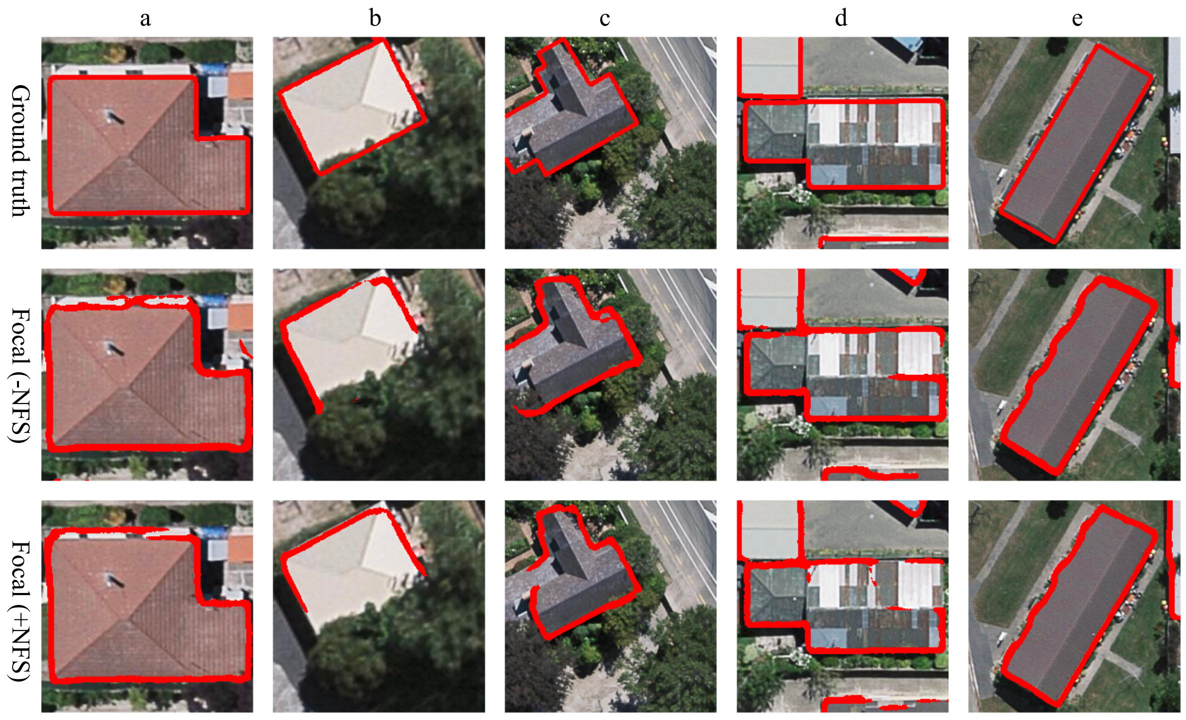


Figure 4.8: Representative results of outline extracted by focal loss that with/without nearest feature selector(NFS). The background and red line represent aerial input and outline, respectively.

Table 4.5: Performances on satellite images by SFCN model trained with aerial images.

Model	Data	SR/Colorized	Precision	Recall	F1-score	Kappa	Jaccard
SFCN	Aerial	-	0.888	**0.907**	**0.897**	**0.625**	**0.814**
	Satellite	$-/-$	**0.909**	0.366	0.521	0.178	0.353
	Satellite	$+/-$	0.900	0.525	0.663	0.278	0.496
	Satellite	$-/+$	0.867	0.837	0.852	0.479	0.742
	Satellite	$+/+$	0.860	0.859	0.859	0.491	0.753

4.3 Model transfer

The stacked fully conventional network(SFCN), which is consists of FCN-8s and FPN, is chosen as the extraction model in this study. The model is trained on the training area of the high-resolution, color aerial imagery. Besides, the pre-trained super-resolution(SR) model and colorization network(ColorNet), are applied to panchromatic satellite to generate SR, colorized or both SR and colorized satellite imagery for extraction evaluation. All experiments are performed on the same dataset and processing platform.

Five commonly used evaluation metrics, including precision, recall, f1-score, kappa coefficient, and jaccard index, are selected for quantitative evaluation.

4.3.1 Quantitative Results

Table 4.5 shows the performance of a pre-trained SFCN model on aerial images as well as satellite images with different image enhancement techniques. On the testing area of aerial imagery, the pre-trained SFCN model achieves the highest values of all evaluation metrics except precision. In the contrast, on low-resolution(LR), panchromatic satellite, the pre-trained model presents the lowest values of all evaluation metrics except precision. The pre-trained undergoes significant performance degradation when it is directly applied to new data with different resolution and color space. Compared with original LR image(*i.e.*, 3^{rd} Row), SFCN presents significantly higher values in all evaluation metrics except precision on SR satellite images(*i.e.*, 4^{th} Row). The SR model helps the pre-trained model to better recognize and extract information from satellite images. From 3^{rd} and 5^{th} Rows, evaluation on colorized satellite image shows significantly higher values in evaluations metrics, especially balanced metrics of f1-score, kappa coefficient, and jaccard index. With both SR and colorization(*i.e.*, 6^{th} Row), the pre-trained SFCN shows the highest values in all evaluation metrics except precision among all satellite images. The result indicates that the proposed SR and colorization pipeline can help the pre-trained extraction model to perform properly on new data with different resolutions and colorspace.

4.3.2 Qualitative Results

4.3.2.1 Result Comparisons at Region Level

Figure 4.9 reveals that the pre-trained stacked fully conventional network(SFCN) performances better on aerial image than on satellite images. On grayscale satellite images(*i.e.*, 2^{nd} and 3^{rd}

Rows), there are more false negatives(Blue). From 2^{nd} and 4^{th} Rows, the result on colorization satellite shows significantly fewer false negatives(Blue) but slightly more false positives(Red). With both SR and colorization(*i.e.*, 5^{th} Rows), performance on satellite image becomes much closer to the one aerial image.

4.3.2.2 Result Comparisons at Patch Level

To further explore the effectiveness of the proposed SR and colorization methods, several representative samples are selected for additional comparison.

Figure 4.10 presents nine representative groups of segmentation results generated by pre-trained SFCN. Even with some false positives (Red) as well as false negatives (Blue), the pre-trained achieve the best performance on patches from aerial image (*i.e.*, 1^{st} Row). For patches from satellite images without color(2^{nd} and 3^{rd} Rows), there are lots of false negatives (Blue). Compared to original satellite patches, results from colorized patches show significantly fewer false negatives but slightly more false positives(2^{nd} vs. 4^{th} Rows). With both SR and colorization(5^{th} Rows), the patches present fewer false positives as well as false negatives that becomes closer to the result from aerial patches.

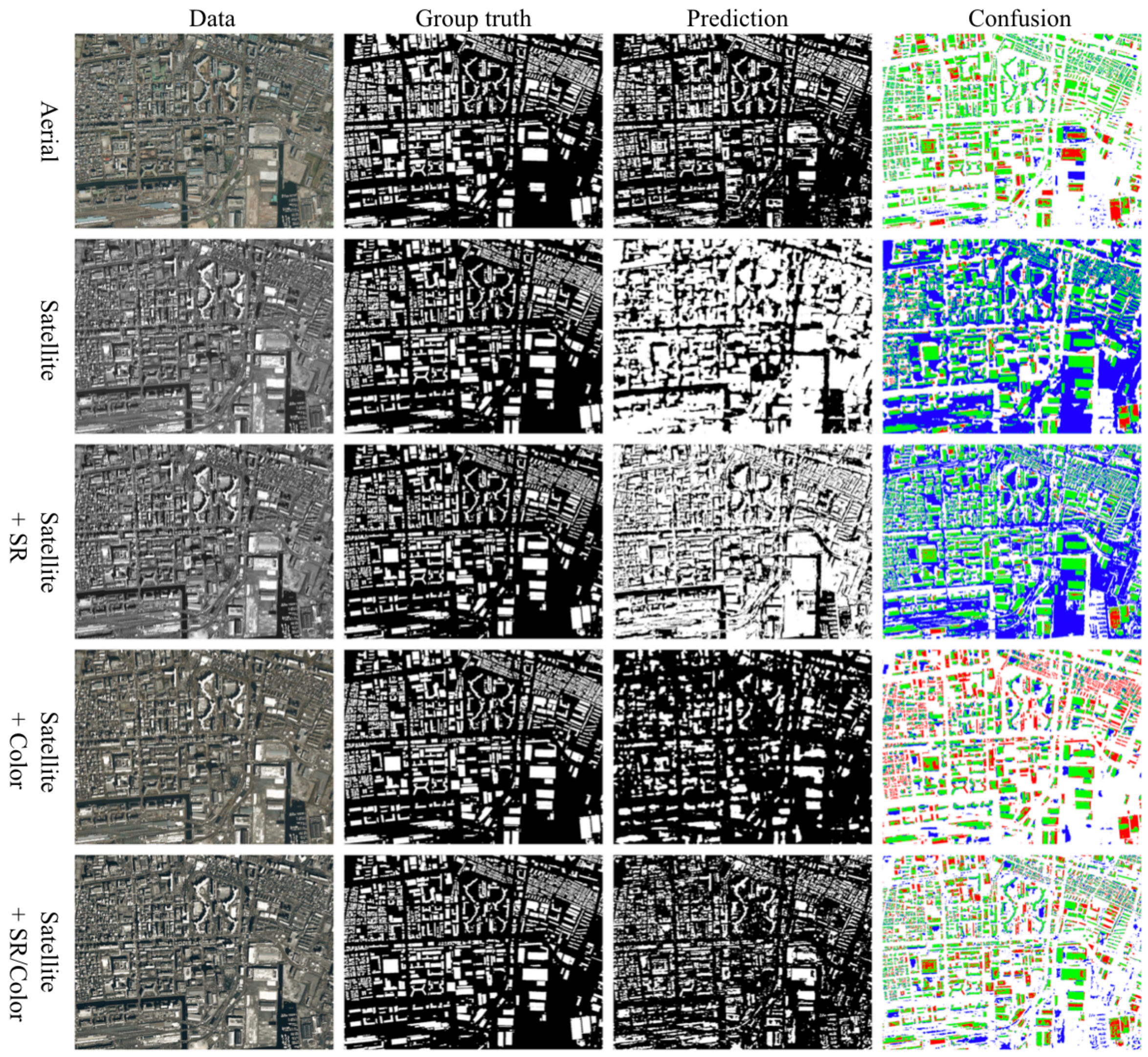


Figure 4.9: Results region-level segmentation of aerial image as well as satellite images. The green, red, blue, and white channels in the confusion results represent true positive, false positive, false negative, and true negative predictions, respectively.

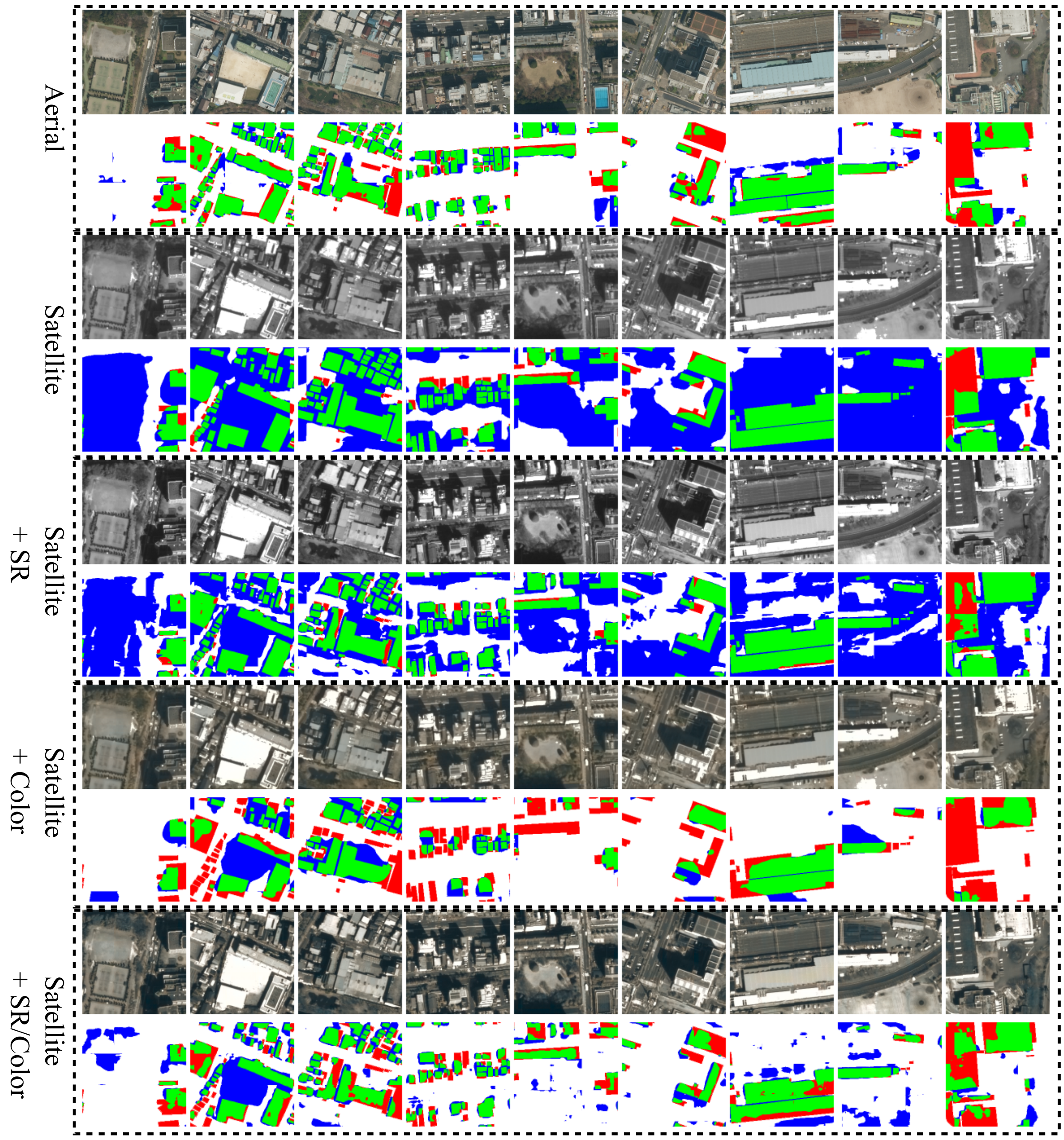


Figure 4.10: Representative results of patch segmentation of aerial image as well as satellite images. The patch size is 256 x 256 pixels. The green, red, blue, and white channels in the results represent true positive, false positive, false negative, and true negative predictions, respectively.

Chapter 5

Discussion

Contents

Figure 5.1 shows the overall tasks required for the complete automatic mapping that enables the automatic generation of a digital map from input aerial or satellite image. In this study, we have conducted (1) polygon feature extraction, (2) line feature extraction, and (3) model transfer. Because of the insufficient computational resource as well as datasets, there are some limitations:

- *Data sources*. The current researches are based on aerial or satellite images. Other types of data, such as SAR, multi-spectral imagery, or polarization imagery, are necessary.

- *Time-series analysis.* The data used in these researches are taken within a limited period. For real applications, such as building change detection, disaster mapping, or informal settlement monitoring, time-series analysis have to be further study.

- *Merge&Vectorization.* Rather than pixel-level semantic segmentation/extraction, vectorized maps are more prefer in daily usage. Thus, in future studies, we will try to merge and vectorize the polygon and line features extracted by our models to achieve a daily used map.

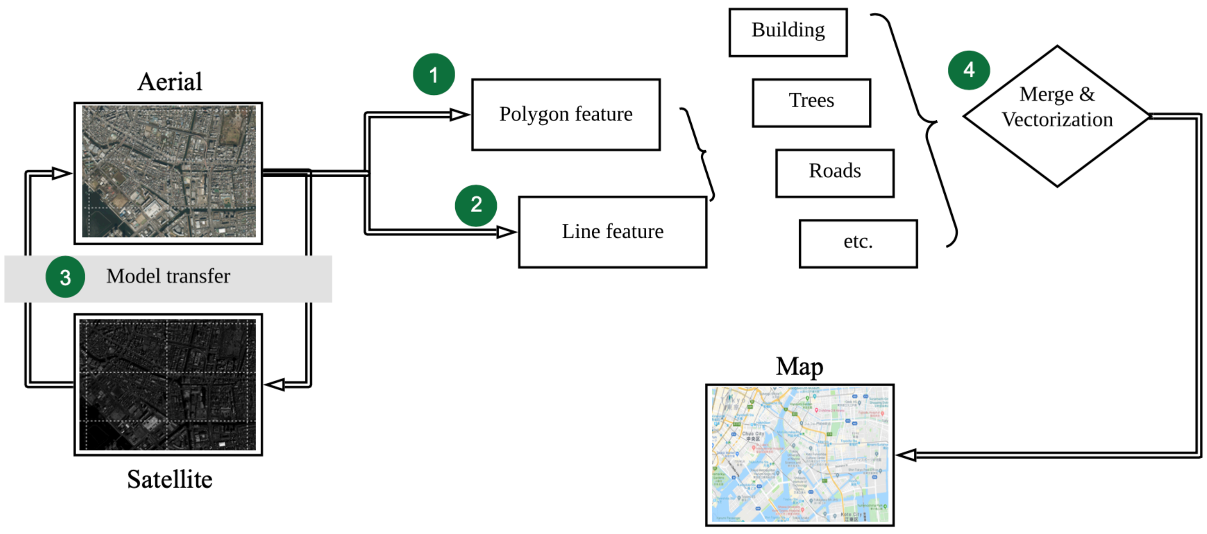


Figure 5.1: Scheme of the complete automatic mapping. Input aerial or satellite image can be converted as digital map through sequential polygon feature extraction, line feature extraction, model transfer and vectorization.

5.1 About Polygon Feature Extraction

5.1.1 Regarding the Stacked Fully Convolutional Network

Deep-learning methods, especially FCN-based models, are widely adopted for automatic polygon feature extraction from large-scale aerial images [82, 83]. Compared to conventional methods, the FCN-based models significantly improve segmentation performance when tested on various benchmark datasets [7,84]. Recently, more advanced FCN-based models have enhanced feature representation capabilities to achieve better model performance (*e.g.*, FPN, MC-FCN, and BR-Net). However, the increased representation capability and as the complexity of the models usually lead to overfitting. Ensemble learning, which utilizes several different networks to generate a weighted prediction, is a promising option to avoid overfitting.

In this study, we designed four SFCNs and proposed a novel feature alignment framework to enhance the performance of the ensemble framework. In contrast to existing ensemble approaches which mainly focus on adding numbers or trying different combinations of basic models, the proposed framework introduces alignment loss to control the similarity and consistency of features extracted from different basic models. Through feature alignment, the proposed ensemble method can achieve a balance between variety and similarity so that better predictions can be achieved from weaker basic models. Qualitative and quantitative results on the testing tiles demonstrated the effectiveness of our proposed stacked fully convolution networks as well as feature alignment framework. Additionally, because of its flexibility, this framework can easily extend to ensemble learning architectures using different numbers of basic models.

5.1.2 Accuracies, Uncertainties, and Limitations

From the sensitivity analysis, different ensemble models show a similar trend that as the weight of alignment loss increases, the performance of the model will increase first and, after a certain level, decline (see details in Figure 4.3). The interpolation of this trend is: (1) When there is no feature alignment(λ=0), features extracted from different basic models are so diverse that they might have different predictions for specific locations. An ensemble of these features doesn't bring better results. (2) When feature alignment is added, at early stages, a higher value of λ forces features extracted from different basic models to be closer to each other so that they can compromise on specific locations and generate better overall predictions. However, if λ rises beyond the optimal value, the extracted features might be too similar to each other, and there will not be enough variety. Thus, the performance of the ensemble method will regress to that of a single basic model. This observation indicates that the feature alignment framework can help achieve a balance in similarity and variety of features in ensemble learning.

Among the methods, the proposed SFCNs with feature alignment (SFCN$_{u\&p}$, +FA) show the highest values for all evaluation metrics. The values of the f1-score, jaccard index, and kappa coefficient are 0.785, 0.646, and 0.742, respectively. SFCN models using two basic models (SFCN$_{f\&p}$, SFCN$_{f\&u}$, and SFCN$_{u\&p}$), with or without feature alignment (*i.e.*, +/− FA), show significantly different performances. Ensemble models with proper weights for alignment loss are generally better than their counterparts without alignment loss. Especially for SFCN$_{f\&u}$, optimal feature alignment gains increments of 4.2% (0.772 vs 0.741) for f1-score, 6.8% (0.629 vs 0.589) for jaccard index, and 5.5% (0.727 vs 0.689) for kappa coefficient. These results indicate that introducing feature alignment leads to better performance of the ensemble model.

However, for ensemble models using three basic models (SFCN$_{f\&u\&p}$), the values of the jaccard index and kappa coefficient only increase about 1.6% (0.640 vs. 0.630) and 1.2%(0.736 vs. 0.727), respectively. Additionally, when compared to the best basic model (FPN), the optimized ensemble model does not show big improvements. (see details in Figure 4.1 b)

Through analysis of computing speed, we observed a significant decrease in computational efficiency at the training stage when applying ensemble learning. Of four SFCN models, the model with three basic models (SFCN$_{f\&u\&p}$) is much slower than the models with two basic models (SFCN$_{f\&p}$, SFCN$_{f\&u}$, and SFCN$_{u\&p}$). Because of the decrease in computational efficiency, even though feature alignment can be easily extended to the ensemble model with all basic models, the proposed ensemble model might not be suitable for the analysis of vast areas (*e.g.*, automatic mapping of the entire country).

5.2 About Line Feature Extraction

5.2.1 Regarding the Nearest Feature Selector

In recent years, fully convolutional networks have shown their power in automatic extraction of the line features, including roads and building outlines [49, 54, 85]. However, these methods are mainly focused on designing deeper or more complex network architectures to enhance the representation capability for better prediction. Their loss functions are not capable of handling misalignments or rotations between inputs and manually created annotations. Since the line feature occupies a typically tiny portion of pixels, the misalignment and rotation will severely interfere with the line extraction accuracy.

In our study, we proposed a nearest feature selector(NFS) module to dynamically re-alignment the prediction and corresponding annotation. The proposed framework be easily integrated into existing loss functions, such as L1, mean square error(MSE), and focal loss. Through dynamic re-alignment, the addition of NFS can locate the correct position of annotation for proper loss calculation. Qualitative and quantitative results on the testing data demonstrated the effectiveness of our proposed NFS.

5.2.2 Accuracies, Uncertainties, and Limitations

Among all methods, the focal loss with NFS shows the highest values for all evaluation metrics. The values of the f1-score, jaccard index, and kappa coefficient are 0.624, 0.597, and 0.468, respectively. Compared with naive L1 loss, the addition of NFS leads to significant increments of all evaluation metrics. The increments of f1-score, kappa coefficient, and jaccard index reach 8.8%, 8.9%, and 9.8%, respectively. For robust loss functions(*e.g.*, focal, and binary cross-entropy loss), the improvement caused by NFS is not significant (see details in Figure 4.4 b). Besides, due to the sliding-and-matching mechanism, the proposed NFS can not be applied to annotations that require rotation correction.

Through the analysis of computational efficiency, we observed a very slight decrease in processing speed when applying NFS. Considering the performance gain by NFS, computational efficiency degradation is ignorable.

5.3 About Model Transfer

5.3.1 Regarding the Super-resolution and Colorization Pipeline

To achieve large-scale automatic mapping, the model built for one location should also be working at another location or the same location but from different data sources. In practice, due to the lack of variety in training data, the pre-trained model undergoes severe performance degradation when applying to new areas or different data sources from the same location.

To avoid overcome this, we propose a model transfer pipeline that integrated both resolution and color transferring to synthesized high resolution, colorized satellite images. Through trained SR and colorization model, the low-resolution, panchromatic image can be converted to high-resolution, color image. Furthermore, the generated image can be appropriately recognized and segmented by the pre-trained SFCN model. Qualitative and quantitative results on the testing area of the satellite imagery demonstrated the effectiveness of our proposed pipeline.

5.3.2 Accuracies, Uncertainties, and Limitations

Among satellite images, the SFCN trained on aerial imagery shows the best performance on images with SR and colorization. The values of precision, recall, f1-score, jaccard index, and kappa coefficient are 0.860, 0.859, 0.859, 0.491, and 0.753, respectively. Compared to the performance on original satellite images, the addition of SR and colorization leads to -4.4%, 63.6%, 29.6%, 76.6%, and 51.8% increments of precision, recall, f1-score, jaccard index, and kappa coefficient, respectively. The improvement caused by SR is less significant than the one caused by colorization(see details in Table 4.5).

Chapter 6

Potential Applications

Figure 6.1 shows the potential applications using the automatic mapping pipeline. Semantic or quantitative information extracted from aerial or satellite images can be used for large-scale spatial/economic analysis such as solar panels availability, flood damage estimation, informal settlement mapping.

- *Solar panels availability.* To achieve zero-emission city, green energies, especially solar energy, are very critical. To estimate the potential capability of solar panels installed on roofs, practical coverage estimation of the building roof is required. With our integrated automatic mapping pipeline, automatic semantic segmentation of the building roof can be applied to the city or country scale. Instead of the time-consuming and labor-intensive field survey, our methods can produce automatic extraction result in several days when there is enough computational resources as well as data.
- *Flood damage estimation.* As an inevitable natural phenomenon, floods bring severe damage to not only local but also global society. In developing countries, because of the insufficient information, rescuing and rebuilding after the damage is slow and ineffective. Using our automatic mapping pipeline, coverage of the buildings from before and after the flood can be automatically extracted using images captured by satellite or unmanned aerial vehicle(UAV). Then, by comparison of the information of before and after, we can focus on the damaged area and estimate the damage-level for a better policy decision.
- *Informal settlement mapping.* Understanding the formation, development, and vanishing of informal settlements are important topics in urban planning and social-economic analysis. Like many social studies, informal settlement mapping mainly relies on field survey, which only covers limited areas or cities. Through our automatic mapping pipeline, large-scale semantic information of land-covers such as buildings, roads, and trees can be extracted. Later, image processing algorithms, including cluttering and vectorization, can be used for further informality estimation.

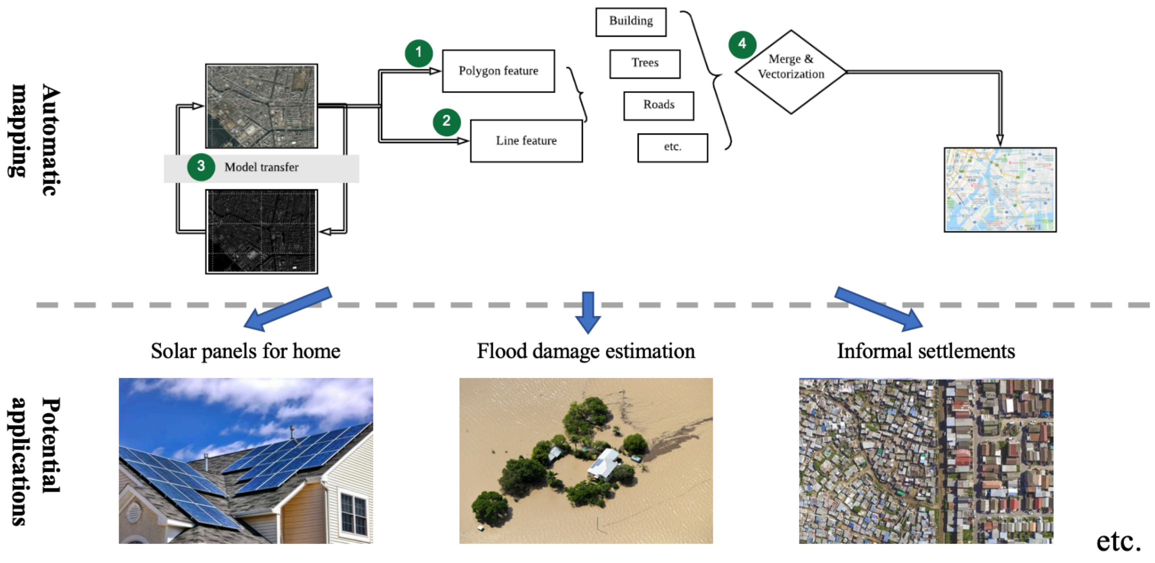


Figure 6.1: Potential applications through automatic mapping. Information extracted from aerial or satellite images can be used for large-scale spatial/economic analysis such as solar panels availability, flood damage estimation, and informal settlement mapping.

Chapter 7

Conclusion

In this study, we provide an integrated automatic mapping solution consisting of polygon feature extraction, line feature extraction, and model transfer.

For robust polygon feature extraction, we propose a novel feature alignment framework for efficient ensemble learning of fully convolutional networks. The proposed framework can be seamlessly integrated with ensemble learning models with a various number of basic models to regulate a balance in similarity and variety of the features extracted from different branches. Their performances are verified by the VHR image dataset with multi-label semantic information. The ensemble models with proposed feature alignment show significantly better performance than existing methods. In SFCN$_{f\&u}$, optimal feature alignment gains increments of 4.2% (0.772 vs 0.741), 6.8% (0.629 vs 0.589), and 5.5% (0.727 vs 0.689) for f1-score, jaccard index, and kappa coefficient, respectively. Sensitivity analysis demonstrated that feature alignment plays an important role in controlling the balance between similarity and variety of the ensemble model. In future studies, we will further optimize our feature alignment framework to achieve better performance in more complex ensemble learning architectures.

For accurate line feature extraction, we design a nearest feature selector(NFS) module to dynamically re-align the prediction and slightly misaligned annotation. The proposed module can be easily combined with existing loss functions to better handle sub-pixel or pixels level misalignments of the manually created annotations. For all loss functions, the addition of proposed NFS show significantly better performance in all evaluation metrics. For classic L1 loss, increments gained by additional NFS are 8.8% of f1-score, 8.9% of kappa coefficient, and 9.8% of jaccard index.

In model transfer, we introduce an integrated super-resolution(SR) and colorization pipeline to facilitate SFCN trained on high-resolution(HR), color aerial image to work appropriately on low-resolution(LR), panchromatic satellite image. With SR and colorization module, the original satellite will be converted to HR and color image that can be successfully recognized and segmented by a pre-trained model. The performance of the pipeline is verified by Tokyo dataset with both aerial and satellite from the same location. The segmentation result on satellite image with additional SR and colorization is significantly better than those on original satellite images. Compared to SR, colorization shows a more significant impact on segmentation results.

List of Publications

[1] Guangming Wu, Yimin Guo, Xiaoya Song, Zhiling Guo, Haoran Zhang, Xiaodan Shi, Ryosuke Shibasaki, and Xiaowei Shao. A stacked fully convolutional networks with feature alignment framework for multi-label land-cover segmentation. *Remote Sensing*, 11(9):1051, 2019.

[2] Guangming Wu, Zhiling Guo, Xiaowei Shao, and Ryosuke Shibasaki. Geoseg: A computer vision package for automatic building segmentation and outline extraction. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 158–161. IEEE, 2019.

[3] Guangming Wu, Zhiling Guo, Xiaodan Shi, Qi Chen, Yongwei Xu, Ryosuke Shibasaki, and Xiaowei Shao. A boundary regulated network for accurate roof segmentation and outline extraction. *Remote Sensing*, 10(8):1195, 2018.

[4] Guangming Wu, Xiaowei Shao, Zhiling Guo, Qi Chen, Wei Yuan, Xiaodan Shi, Yongwei Xu, and Ryosuke Shibasaki. Automatic building segmentation of aerial imagery using multi-constraint fully convolutional networks. *Remote Sensing*, 10(3):407, 2018.

[5] Zhiling Guo, Guangming Wu, Xiaodan Shi, Mingzhou Sui, Xiaoya Song, Yongwei Xu, Xiaowei Shao, and Ryosuke Shibasaki. Geosr: A computer vision package for deep learning based single-frame remote sensing imagery super-resolution. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 3376–3379. IEEE, 2019.

[6] Zhiling Guo, Guangming Wu, Xiaoya Song, Wei Yuan, Qi Chen, Haoran Zhang, Xiaodan Shi, Mingzhou Xu, Yongwei Xu, Ryosuke Shibasaki, et al. Super-resolution integrated building semantic segmentation for multi-source remote sensing imagery. *IEEE Access*, 7:99381–99397, 2019.

[7] Qi Chen, Lei Wang, Yifan Wu, Guangming Wu, Zhiling Guo, and Steven L Waslander. Aerial imagery for roof segmentation: A large-scale dataset towards automatic mapping of buildings. *ISPRS journal of photogrammetry and remote sensing*, 147:42–55, 2019.

[8] Zhiling Guo, Qi Chen, Guangming Wu, Yongwei Xu, Ryosuke Shibasaki, and Xiaowei Shao. Village building identification based on ensemble convolutional neural networks. *Sensors*, 17(11):2487, 2017.

Bibliography

[1] Eric Abbott and David Powell. Land-vehicle navigation using gps. *Proceedings of the IEEE*, 87(1):145–162, 1999.

[2] Douglas A Stow, Allen Hope, David McGuire, David Verbyla, John Gamon, Fred Huemmrich, Stan Houston, Charles Racine, Matthew Sturm, Kenneth Tape, et al. Remote sensing of vegetation and land-cover change in arctic tundra ecosystems. *Remote sensing of environment*, 89(3):281–308, 2004.

[3] Gerald E Heilman, James R Strittholt, Nicholas C Slosser, and Dominick A Dellasala. Forest fragmentation of the conterminous united states: Assessing forest intactness through road density and spatial characteristics: Forest fragmentation can be measured and monitored in a powerful new way by combining remote sensing, geographic information systems, and analytical software. *AIBS Bulletin*, 52(5):411–422, 2002.

[4] Liv Norunn Hamre, Stein Tage Domaas, Ingvild Austad, and Knut Rydgren. Land-cover and structural changes in a western norwegian cultural landscape since 1865, based on an old cadastral map and a field survey. *Landscape ecology*, 22(10):1563–1574, 2007.

[5] Ismael Colomina and Pere Molina. Unmanned aerial systems for photogrammetry and remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 92:79–97, 2014.

[6] Lei Ma, Manchun Li, Xiaoxue Ma, Liang Cheng, Peijun Du, and Yongxue Liu. A review of supervised object-based land-cover image classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130:277–293, 2017.

[7] Chris A Glasbey. An analysis of histogram-based thresholding algorithms. *CVGIP: Graphical models and image processing*, 55(6):532–537, 1993.

[8] Jer-Sen Chen, Andres Huertas, and G Medioni. Fast convolution with laplacian-of-gaussian masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (4):584–590, 1987.

[9] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2):358–367, 1988.

[10] Judith MS Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

[11] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11):1101–1113, 1993.

[12] Keh-Shih Chuang, Hong-Long Tzeng, Sharon Chen, Jay Wu, and Tzong-Jer Chen. Fuzzy c-means clustering with spatial information for image segmentation. *computerized medical imaging and graphics*, 30(1):9–15, 2006.

[13] Ding Zhen, Hu Zhongshan, Yang Jingyu, and Tang Zhenming. Fcm algorithm for the research of intensity image segmentation [j]. *Acta Electronica Sinica*, 5:39–43, 1997.

[14] Thrasyvoulos N Pappas. An adaptive clustering algorithm for image segmentation. *IEEE Transactions on signal processing*, 40(4):901–914, 1992.

[15] Alain Tremeau and Nathalie Borel. A region growing and merging algorithm to color segmentation. *Pattern recognition*, 30(7):1191–1203, 1997.

[16] Ali Ozgun Ok. Automated detection of buildings from single vhr multispectral images using shadow information and graph cuts. *ISPRS journal of photogrammetry and remote sensing*, 86:21–40, 2013.

[17] Konstantinos Karantzalos and Nikos Paragios. Recognition-driven two-dimensional competing priors toward automatic and accurate building detection. *IEEE Transactions on Geoscience and Remote Sensing*, 47(1):133–144, 2009.

[18] Sedat Ozer, Deanna L Langer, Xin Liu, Masoom A Haider, Theodorus H van der Kwast, Andrew J Evans, Yongyi Yang, Miles N Wernick, and Imam S Yetik. Supervised and unsupervised methods for prostate cancer segmentation with multispectral mri. *Medical physics*, 37(4):1873–1883, 2010.

[19] Miao Li, Shuying Zang, Bing Zhang, Shanshan Li, and Changshan Wu. A review of remote sensing image classification techniques: The role of spatio-contextual information. *European Journal of Remote Sensing*, 47(1):389–411, 2014.

[20] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[21] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[22] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[23] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[24] Jordi Inglada. Automatic recognition of man-made objects in high resolution optical remote sensing images by svm classification of geometric image features. *ISPRS journal of photogrammetry and remote sensing*, 62(3):236–248, 2007.

[25] Örsan Aytekin, U Zöngür, and U Halici. Texture-based airport runway detection. *IEEE Geoscience and Remote Sensing Letters*, 10(3):471–475, 2013.

[26] Yanni Dong, Bo Du, and Liangpei Zhang. Target detection based on random forest metric learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(4):1830–1838, 2015.

[27] Er Li, John Femiani, Shibiao Xu, Xiaopeng Zhang, and Peter Wonka. Robust rooftop extraction from visible band images using higher order crf. *IEEE Transactions on Geoscience and Remote Sensing*, 53(8):4483–4495, 2015.

[28] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[29] Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.

[30] Zhiling Guo, Xiaowei Shao, Yongwei Xu, Hiroyuki Miyazaki, Wataru Ohira, and Ryosuke Shibasaki. Identification of village building via google earth images and supervised machine learning methods. *Remote Sensing*, 8(4):271, 2016.

[31] Michael Kampffmeyer, Arnt-Borre Salberg, and Robert Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–9, 2016.

[32] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[33] Lin Li, Jian Liang, Min Weng, and Haihong Zhu. A multiple-feature reuse network to extract buildings from remote sensing imagery. *Remote Sensing*, 10(9):1350, 2018.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[35] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[36] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[37] Guangming Wu, Xiaowei Shao, Zhiling Guo, Qi Chen, Wei Yuan, Xiaodan Shi, Yongwei Xu, and Ryosuke Shibasaki. Automatic building segmentation of aerial imagery using multi-constraint fully convolutional networks. *Remote Sensing*, 10(3):407, 2018.

[38] Igor V Tetko, David J Livingstone, and Alexander I Luik. Neural network studies. 1. comparison of overfitting and overtraining. *Journal of chemical information and computer sciences*, 35(5):826–833, 1995.

[39] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.

[40] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[41] Sebastien C Wong, Adam Gatt, Victor Stamatescu, and Mark D McDonnell. Understanding data augmentation for classification: When to warp? In *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on*, pages 1–6. IEEE, 2016.

[42] Markus Grasmair, Otmar Scherzer, and Markus Haltmeier. Necessary and sufficient conditions for linear convergence of l1-regularization. *Communications on Pure and Applied Mathematics*, 64(2):161–182, 2011.

[43] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[45] Bruce E Rosen. Ensemble learning using decorrelated neural networks. *Connection science*, 8(3-4):373–384, 1996.

[46] Zhiling Guo, Qi Chen, Guangming Wu, Yongwei Xu, Ryosuke Shibasaki, and Xiaowei Shao. Village building identification based on ensemble convolutional neural networks. *Sensors*, 17(11):2487, 2017.

[47] Mark Polak, Hong Zhang, and Minghong Pi. An evaluation metric for image segmentation of multiple objects. *Image and Vision Computing*, 27(8):1223–1227, 2009.

[48] Jean Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational linguistics*, 22(2):249–254, 1996.

[49] Guangming Wu, Zhiling Guo, Xiaodan Shi, Qi Chen, Yongwei Xu, Ryosuke Shibasaki, and Xiaowei Shao. A boundary regulated network for accurate roof segmentation and outline extraction. *Remote Sensing*, 10(8):1195, 2018.

[50] Bernd Jähne, Horst Haussecker, and Peter Geissler. *Handbook of computer vision and applications*, volume 2. Citeseer, 1999.

[51] John Canny. A computational approach to edge detection. In *Readings in Computer Vision*, pages 184–203. Elsevier, 1987.

[52] Volodymyr Mnih and Geoffrey E Hinton. Learning to detect roads in high-resolution aerial images. In *European Conference on Computer Vision*, pages 210–223. Springer, 2010.

[53] Yanan Wei, Zulin Wang, and Mai Xu. Road structure refined cnn for road extraction in aerial image. *IEEE Geoscience and Remote Sensing Letters*, 14(5):709–713, 2017.

[54] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

[55] Lichen Zhou, Chuang Zhang, and Ming Wu. D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction. In *CVPR Workshops*, pages 182–186, 2018.

[56] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *IEEE signal processing magazine*, 20(3):21–36, 2003.

[57] Hong Zhu, Xinming Tang, Junfeng Xie, Weidong Song, Fan Mo, and Xiaoming Gao. Spatio-temporal super-resolution reconstruction of remote-sensing images based on adaptive multi-scale detail enhancement. *Sensors*, 18(2):498, 2018.

[58] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM transactions on graphics (tog)*, volume 23, pages 689–694. ACM, 2004.

[59] Yong Li, Tao Ju, and Shi-Min Hu. Instant propagation of sparse edits on images and videos. In *Computer Graphics Forum*, volume 29, pages 2049–2054. Wiley Online Library, 2010.

[60] Daniel Sỳkora, John Dingliana, and Steven Collins. Lazybrush: Flexible painting tool for hand-drawn cartoons. In *Computer Graphics Forum*, volume 28, pages 599–608. Wiley Online Library, 2009.

[61] Xiaopei Liu, Liang Wan, Yingge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng. Intrinsic colorization. In *ACM Transactions on Graphics (TOG)*, volume 27, page 152. ACM, 2008.

[62] Fuzhang Wu, Weiming Dong, Yan Kong, Xing Mei, Jean-Claude Paul, and Xiaopeng Zhang. Content-based colour transfer. In *Computer Graphics Forum*, volume 32, pages 190–203. Wiley Online Library, 2013.

[63] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423, 2015.

[64] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*, pages 577–593. Springer, 2016.

[65] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[66] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (TOG)*, 35(4):110, 2016.

[67] Mary L Comer and Edward J Delp. Morphological operations for color image processing. *Journal of electronic imaging*, 8(3):279–290, 1999.

[68] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[69] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[70] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[71] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[72] Jawad Nagi, Frederick Ducatelle, Gianni A Di Caro, Dan Cireşan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jürgen Schmidhuber, and Luca Maria Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pages 342–347. IEEE, 2011.

[73] Kurt Novak. Rectification of digital imagery. *Photogrammetric engineering and remote sensing*, 58:339–339, 1992.

[74] John Shore and Rodney Johnson. Properties of cross-entropy minimization. *IEEE Transactions on Information Theory*, 27(4):472–482, 1981.

[75] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[76] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[78] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[79] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017.

[80] Yongyang Xu, Liang Wu, Zhong Xie, and Zhanlong Chen. Building extraction in very high resolution remote sensing imagery using deep learning and guided filters. *Remote Sensing*, 10(1):144, 2018.

[81] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[82] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):645–657, 2017.

[83] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. fully convolutional networkss for remote sensing image classification. In *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*, pages 5071–5074. IEEE, 2016.

[84] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raskar. Deepglobe 2018: A challenge to parse the earth through satellite images. *ArXiv e-prints*, 2018.

[85] Songbing Wu, Chun Du, Hao Chen, Yingxiao Xu, Ning Guo, and Ning Jing. Road extraction from very high resolution images using weakly labeled openstreetmap centerline. *ISPRS International Journal of Geo-Information*, 8(11):478, 2019.