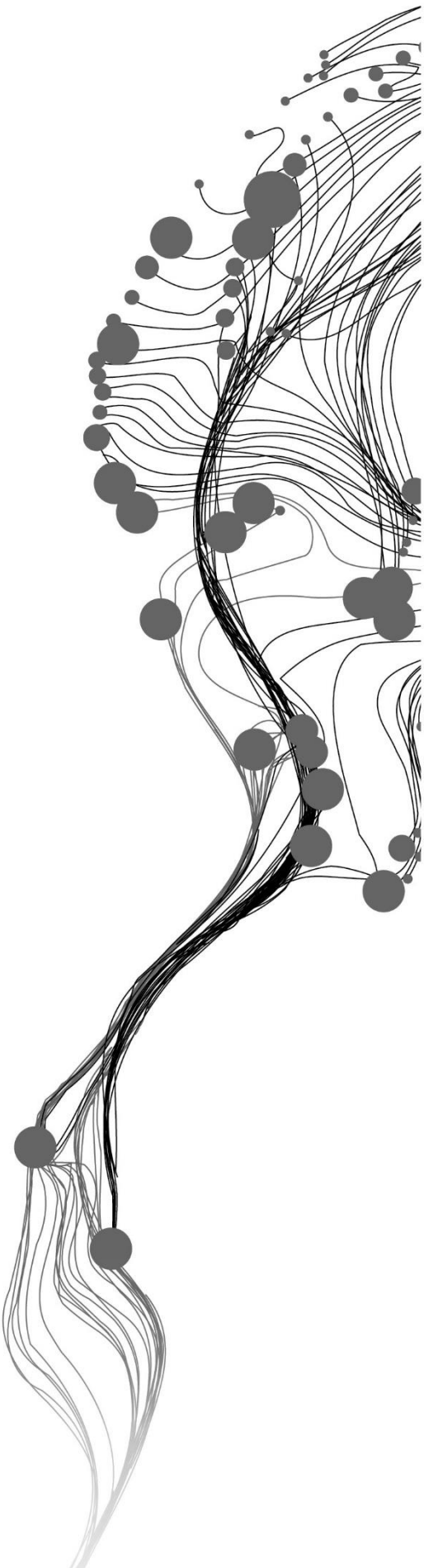


Automated Delineation of Smallholder Farm Fields using Generative Adversarial Network

QIUYU YAN
Enschede, The Netherlands, [06, 2020]

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: [Geoinformation Science and Earth Observation]

SUPERVISORS:
dr. C. Persello
dr. Y. Yang

THESIS ASSESSMENT BOARD:
Prof.dr.ir. A. Stein (Chair)
dr. C. Persello (First supervisor)
dr. Y. Yang (Second supervisor)
dr. M.N. Koeva (External Examiner, ITC, PGM department)
drs. J.P.G. Bakx (Procedural Advisor)

DISCLAIMER

This document describes work undertaken as part of a programme of study at the Faculty of Geo-Information Science and Earth Observation of the University of Twente. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the Faculty.

ABSTRACT

Smallholder farms play a vital role in agricultural production in many developing countries around the world. As basic geographic information of agricultural resources, accurate boundaries of smallholder farm fields are important and indispensable geo-information for farmers, managers and policymakers to help them manage and utilize their agricultural resource. Beyond that, accurate delineation of smallholder farm fields could promote the sustainable development of agriculture. However, traditional manual methods such as image digitisation by visual inspection of satellite images or filed campaigns are inefficient and time-consuming. Therefore, this research aims to propose an automated algorithm by fully convolutional neural networks (FCN) in combination with generative adversarial networks (GAN) to improve the delineation accuracy of smallholder farms using Very High Resolution (VHR) images. This research consists of two parts. In the first part, we investigate three state-of-the-art fully-convolutional deep network architectures (U-Net, PSPNet, SegNet) to find the optimal architecture in the contour detection task of smallholder farm fields. After that, we aim to conduct the optimal FCN architecture in combination with GAN methods to improve the accuracy of contour detection. Thus, the second part explores the potentials of two GAN methods (ContourGAN and pixel2pixelGAN) for this specific task.

The study area is in the Sudano-Sahelian savanna region of northern Nigeria, around the city of Kofa, Bebeji Local Government Area, Kano State. It is a 3×2 km area which composes of abundant small fields and most of them have three or more crops. The VHR image dataset consists of six 1000×1000 pixels tiles extracted from a WorldView-3 image acquired on September 25th 2015. By comparing different methods in this research based on the F1-score, we aim to propose an optimal method for this contour detection task of smallholder farm fields.

Keywords
Smallholder farm, contour detection, fully convolutional neural network, generative adversarial networks

ACKNOWLEDGEMENTS

I am very grateful for my college life and study.
I would like to express my gratitude to my supervisor, Dr. C. Persello, for guiding me in my study, life and paper writing. At every stage of my paper, from the thesis theme to the mid-term review and the later revision, he provided me with huge guidance in the whole process.
Also, I would like to express my gratitude to the teachers and classmates who have taught me and helped me to move forward in the field of geographic information science.
Finally, I would like to express my gratidude to my family. They always support me and care about me unconditionally.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Smallholder farms whose areas are from 1 to 10 hectares play a vital role in agricultural production in many developing countries around the world. Smallholder farms account for almost 75% of the world's agricultural lands, and most of them are less than 2 hectares (Lowder et al., 2016). Smallholder farms often take the family as a unit, and most farmers lack professional knowledge about how to manage their farms. As basic geographic feature information of precision agriculture, the accurate boundaries of smallholder farm fields are indispensable geo-information for farmers, managers, and policymakers to help them manage and utilize their agricultural resources. Beyond that, the precise delineation of smallholder farm fields could promote the sustainable development of agriculture, which is vital for ensuring food security in developing countries.

Based on these objectives, we aim to acquire the spatial delineation information from the Very High Resolution (VHR) satellite images because the traditional manual methods are inefficient and time-consuming (García-Pedrero et al., 2017). Besides, it is a challenging task because their boundaries which are based on vague delineation and irregular shapes, are difficult to delineate. Moreover, the complex textual patterns and mixed-cropping systems between fields also increase the difficulty of standard delineation. Therefore, we need a new automatic and accurate delineation algorithm to deal with these problems.

In recent years, deep learning networks are widely applied in contour detection tasks. Compared with other methods, deep learning approaches could learn and extract contextual features better at different layers based on spatial information (Bergado et al., 2016). However, deep learning models including fully convolutional neural networks (FCNs) usually face a thorny problem because these models theoretically need an enormous quantity of training samples to learn general models, and this problem will have a bad influence on the classification results. In the remote sensing field, it is impractical and time-consuming to label all the training images because of its tremendous volume and abundant geographic information, especially the contours. Thus, how to increase the number of training data and samples is a topical problem. This research aims to introduce generative adversarial networks (GANs) to tackle this problem and improve the accuracy of boundary delineation techniques. Since GANs were proposed by Ian Goodfellow in 2014 (Goodfellow et al., 2014), GANs give researchers a new and effective method for automatic delineation. It is proved that the adversarial training method could be used for semantic segmentation models because it could detect and correct the inconsistencies between ground truth and segmentation results (Luc et al., 2016). The GANs consist of the a generator and a discriminator, which could keep the adversarial improvements. Different from the other generative models, the GANs use the generators and discriminators as adversaries with respect to each other to produce the samples so that they can improve during the adversarial process (Hong et al., 2017). As an effective learning method, GANs could deal with the problems such as limited training samples because its generator could create virtual samples for training data, which could improve the classification performance (Lin et al., 2017). Based on the strategy of Persello et. al (2019), this research aims to pay more attention to the first step which is sparse contour detection in contour detection tasks by introducing GAN.

Based on the above, how to extract accurate contours of smallholder farm fields is a challenging topic and here this research aims to propose a high-accuracy algorithm using FCNs in combination with GAN to improve the delineation accuracy of smallholder farms from VHR images.

1.1. Research objectives and questions

Firstly, VHR satellite images of smallholder farms include complex geo-information and the delineation is difficult due to the specific attributes. In this research, we want to use FCN to extract the preliminary boundaries of smallholder farm fields in the VHR images. It is prepared for the following study. However, due to the limited training samples in remote sensing data, the accuracy of output still can be improved. Thus, we want to combine it with the GAN and improve the accuracy in boundary detection tasks after adversarial training. Finally, an effectual method to evaluate the accuracy of automatic delineation is required such as precision, error and degree of confidence because the aim is to improve the segmentation accuracy.

Therefore, the improved accuracy of delineation is essential. The main objective of this research is to develop an automated technique based on GANs to automatically delineate smallholder farms from VHR satellite images. This aim could be achieved through several specific objectives as follows:

1. To perform an automatic delineation by using FCN.
2. To perform an automatic delineation by introducing GAN.
3. To choose and improve the FCN and GAN architecture for the delineation of agricultural fields.
4. To evaluate and compare the delineation results of different methods.

Based on the research gap and the main goal, here we can list four specific research questions as follows:

1. How to perform automatic delineation by FCN and GAN?
2. How to combine FCN with GAN for the delineation of agricultural fields?
3. Which architecture of FCN and GANs could have a better performance in terms of smallholder farm fields?

1.2. Background and related work

Segmentation algorithms

Image segmentation aims to divide the image into connected regions or categories which correspond to different objects or parts of objects. Some typical algorithms such as mean shift, split and merge, region growing and multiresolution segmentation are applied in many fields. Mean shift algorithm seeks modes of the given set of points. Given specific kernel and bandwidth, it repeatedly centers the search window by mean of the data until convergence and assigns points that lead to nearby modes to the same cluster (Comaniciu & Meer, 2002). In split and merge algorithm, 'split' means that the whole image is divided iteratively in elementary homogeneous regions on the basis of a predefined criterion and 'merge' means that adjacent regions are merged if they satisfy the predefined homogeneity criterion (Chen & Pavlidis, 1979). As to region growing algorithm, it means repeatedly label the neighbour of the seed point if their attributes are similar until there is no more pixel that could be labelled (Adams & Bischof, 1994).

Contour detection algorithms

Contour detection aims to detect object contours. For efficient and accurate delineation, many researchers proposed a variety of methods based on image segmentation. Before the deep learning models, contour detection has a long history in image processing and computer vision fields and it used to be realized by Roberts (Roberts, 1963), Sobel (Sobel, 1972), Canny and other operators. Some researches aim to extract the colour, brightness and texture as features and train an edge classifier for pixels (Martin et al., 2004a). The delineation of large and man-made objects can be well extracted with edge detection and region segmentation from satellite images (Mueller et al., 2004). From the attributes of the images, a segmentation method based on tonal and textural gradients of each region was proposed and Snakes Algorithm can improve the detection of their field boundaries (Tiwari et al., 2009). Besides, the field-based sub-boundaries are used to perform boundary analysis by the perceptual grouping (Turker & Kok, 2013).

Martin et. Al (2014) proposed features which are extracted from colour, brightness and texture to train a contour based on pixels. In conclusion, traditional segmentation methods are sensitive to intrinsic variability and dependent on parameter selection which will cause an extra error in segmentation processing (García-Pedrero et al., 2017).

Based on the above traditional automatic and semiautomatic delineation, more and more researchers turn sight to machine learning and neural networks as artificial intelligence technology becomes mature. Short and long skip connections are added to extend FCNs to build highly deep FCNs and the results show the noticeable improvements in biomedical image segmentation (Drozdzal et al., 2016). The combination of FCN and conditional random fields approve its high expansibility and accuracy in CT abdomen images (Christ et al., 2016). Since the FCN could be widely applied in biological image processing, it could work for delineation of remote sensing images as well (Maggiori et al., 2016). An encoder-decoder FCN called FCED was proposed by image-to-image architecture for contour detection (J. Yang et al., 2016). Xie and Tu (2015) propose an edge detection algorithm that uses a Fully Convolutional Network (FCN) with multiple side outputs, named holistically-nested architecture, for a deeply supervised training and it has a promising performance (Xie & Tu, 2015). In this framework, raw images are transformed into contour information and nested contours are generated by feature maps. Besides, the FCN has high computational efficiency and less information loss because of upsampling and skip structure. Irrelevant edges will be discarded when detecting field boundaries by trained FCN and it is proved that FCN for pixel-wise classification has good performance in boundary detection tasks (Shelhamer et al., 2017). Persello et. Al (2019) proposed an approach which consists of sparse contour detection, closed segment extraction and final contour generation for delineation of smallholder farm fields by FCN and combinatorial grouping (Persello et al., 2019).

Generative Adversarial Networks

In recent researches, GANs are one of the exciting methods in many fields including pixel classification. Although GAN has shown its great potential in image synthesis and pixel classification, it still faces many problems such as training instability and optimization difficulties. More and more architectures were proposed for these issues. Deep convolutional GAN is proved as an effective approach to data synthesis and pixel classification (Radford et al., 2015). The discriminator of Auxiliary classifier GAN (AC-GAN) is modified to be a softmax classifier which means it could output multiclass label probabilities (Zhu et al., 2018). Zhu et al. (2018) also proposed 1D-GAN and 3D-GAN which are modified by the theory of AC-GAN for image classification based on hyperspectral satellite images. Multiple-layer feature-matching generative adversarial networks (MARTA GANs) which are based on DCGAN is also proposed for unsupervised image classification in remote sensing fields (Lin et al., 2017). The MARTA GANs introduces a multiple-feature-matching layer by perceptual loss and feature matching loss for high-resolution remote sensing images. Moreover, Wasserstein GAN (WGAN) and WGAN-Gradient penalty (WGAN-GP) gain promising results in hyperspectral image classification (Sun & Bourennane, 2019). In ContourGAN architectures, they find the training instability is not only from the generator or discriminator but also from the adversarial training procedure (H. Yang et al., 2019). In other study fields, Perceptual GAN is used to improve the detection rate of small objects by generating super-resolved representations for small objects (Li et al., 2017). FCN and GAN are applied in some specific segmentations such as sclera segmentation (Lucio et al., 2018).

1.3. Main contributions

Inspired by these methods and architecture in the above sections, this research aims to propose an automated algorithm by combining FCNs with GANs to improve the delineation accuracy of smallholder farms using VHR images. This research consists of two parts. In the first part, we investigate three state-of-the-art fully-convolutional deep network architectures (U-Net, PSPNet, SegNet) to find the optimal architecture in the contour detection task of smallholder farm fields. The second part explores the potentials of two GAN methods (ContourGAN and pixel2pixelGAN) for this specific task.
The main contributions of this research can be summarized as follows:

- An improved approach which performs optimal FCN architecture in combination with GAN methods for automatic delineation of smallholder farm fields
- The introduction of GAN-based technique for VHR satellite images in contour detection tasks.

2. STUDY AREA AND DATA

In this research, the study area is in the Sudano-Sahelian savanna region of northern Nigeria, around the city of Kofa, Bebeji Local Government Area, Kano State. It is a 3×2 km² area which composes of abundant small fields and most of them have three or more crops. The VHR image dataset consists of six 1000×1000 tiles extracted from a WorldView-3 image acquired on September 25th 2015. As figure 2-1 shows, TR data (TR1, TR2 & TR3) is for training and TS data (TS1, TS2 & TS3) is for accuracy assessment.

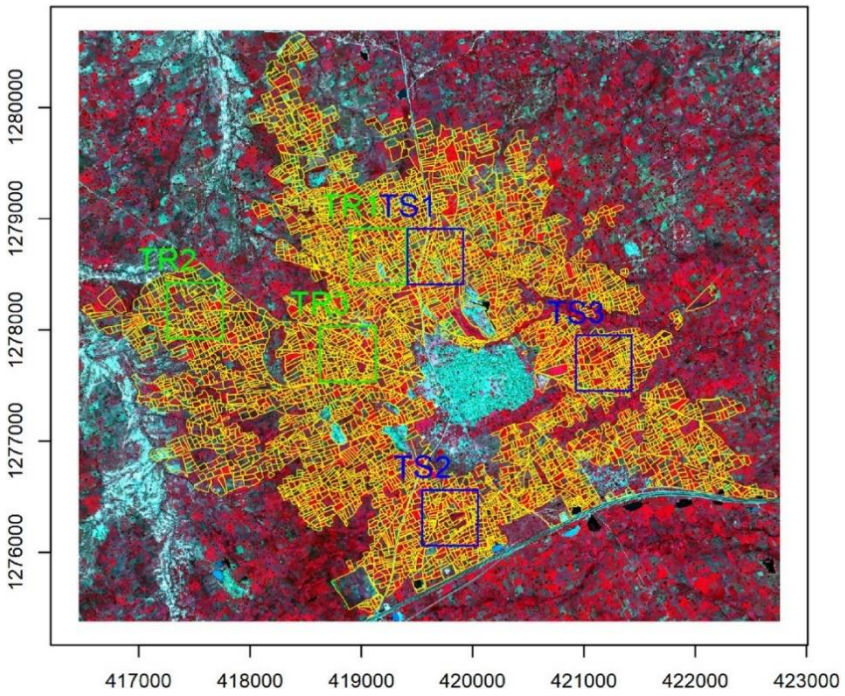


Figure 2-1. study area of Kofa (Persello et al., 2019)

In this dataset, WorldView-3 satellite has one panchromatic band (resolution: 0.31m), 8 multispectral visible and near-infrared bands (resolution: 1.24m), 8 short-wave infrared bands (resolution: 3.7m) and 12 CAVIS bands (resolution: 30m).

3. METHODS

In this section, we aim to present some methods by GAN-based techniques to improve the delineation accuracy of smallholder farms from VHR images. From many state-of-the-art architectures of FCN, we select three models which are U-Net, PSPNet and SegNet to do the pixel classification. Then, we will explain the details of these methods in the following sub-sections. Some of them will get better results and some get worse, but all of them are worthwhile experiments in this new field.

3.1. FCN

3.1.1. U-Net

U-net was originally used for biomedical image segmentation, and its architecture consists of one contraction network (encoder) and one expansion network (decoder) (Ronneberger et al., 2015). The main idea is to add a network similar to the previous one behind the shrinking network, where the pooling operator will be replaced by the up-sampling operator. As a result, these layers increase the resolution of the output. For positioning, the high-resolution features from the contraction network are combined with the up-sampled output. The continuous convolution layer can then learn to assemble more accurate outputs based on this information.

- The encoder part is similar to a typical convolution network structure like VGG. It consists of two repeated 3×3 convolutional kernels (unpadded convolution). The encoder uses the modified linear unit (rectified linear unit, ReLU) as activation function and downsampling (step is 2 with 2×2 convolutional kernel) as biggest pooling operation. Under each sampling steps, the number of all channels will be double.
- In the decoder part, each step includes upsample of the feature graph. Then 2×2 convolutional kernels are used for convolution operation (up-convolution) to reduce the number of feature channels by half. Then the corresponding clipped feature graph in the cascade contraction network; Then, two 3×3 convolutional kernels are used for convolutional operation, and both of them use ReLU activation function. In the last layer, the 1×1 convolutional kernel is used for a convolutional operation to map the output layer.

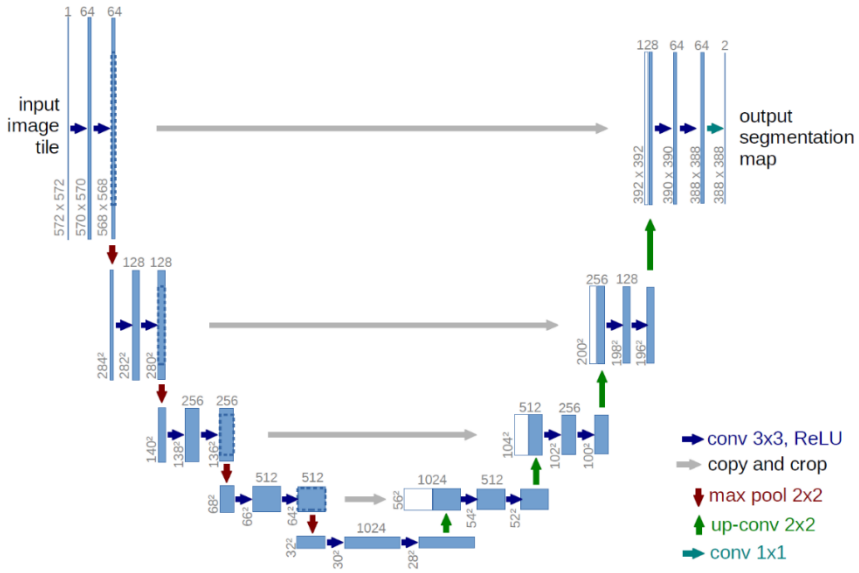


Figure 3-1. Architecture of U-Net (Ronneberger et al., 2015)

In the figure above, each blue block represents a multi-channel feature graph, the number of channels in the feature graph is marked at the top, the x-y size is set at the bottom left edge of the block, and the arrows of different colours represent different operations.
The main idea of FCN in semantic segmentation is to use continuous layers to complement the usual shrinkage network, adding sampling layers to the location of the discriminant output, which increases the resolution of the output layer for positioning (Long et al., 2014). The high separation rate from the contraction path is combined with the up-sampled output, and based on this information, a continuous convolution layer can learn to assemble more accurate outputs. The difference between U-net and common FCN is that the up-sampling of U-net still has a large number of channels, which enables the network to spread the context information to a higher resolution. As a result, the expansion path and the contraction path are symmetric, forming a u-shaped shape. The network has no full connection layer, but only an effective part of each convolution layer.

3.1.2. PSPNet

The main problem with the current fcn-based model is the lack of a suitable strategy to take advantage of the category clues in the global scenario. For the understanding of typical complex scenes, spatial pyramid pooling has been widely used in the past to obtain global image-level features. This spatial statistical method provides a good descriptor for the overall scene analysis. In order to combine the appropriate global characteristics, the pyramid scenario resolution network (PSPNet) was proposed (Zhao et al., 2017). In addition to the traditional convolutional FCN for pixel prediction, it extended the pixel level features to the pyramid pooling which is shown in figure 3-2 (c). Local and global clues work together to make the final prediction more reliable.

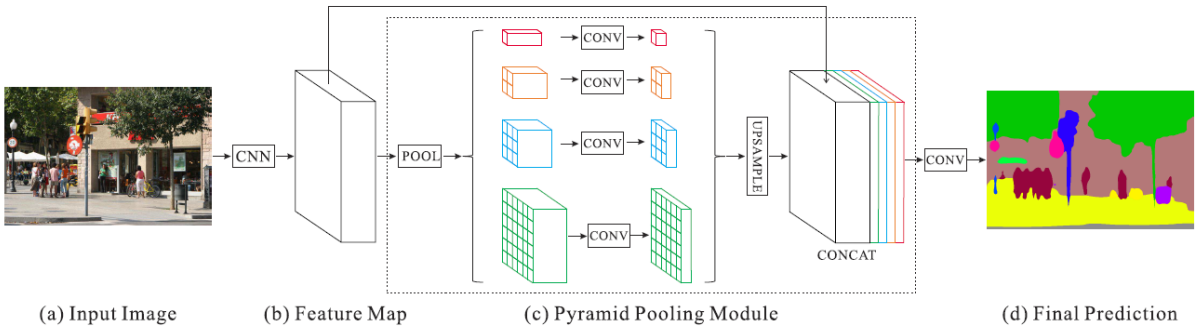


Figure 3-2. Architecture of PSPNet (Zhao et al., 2017)

From the figure 3-2, we could understand the network architecture of PSPNet. After the input image, the feature map is extracted by using the pre-trained image with ResNet. The size of the feature map is 1/8 of the input image, as shown in (b). On the feature map, it uses the pyramid pooling module in (c) to collect context information. Using a 4-tier pyramid structure, the pooled kernel covers all, half, and small portions of the image. They are merged into global prior information. In the last part of (c), it concats the previous pyramid feature map with the original feature map. Then convolution is performed to generate the final prediction graph in (d).
PSPNet provides a valid global context prior for pixel-level scene resolution. Pyramid pooling modules can collect hierarchical information and are more representative than global pooling. In terms of computation, PSPNet is not much more than the original empty convolution FCN network (Zhao et al., 2017).

3.1.3. SegNet

As shown in the figure above, SegNet is a symmetric network composed of an encoder network (left) and a decoder network (right). After inputting an RGB image, the network classifies the objects in the image (for example, "road", "car", "building", etc.) according to the semantic information of the objects in the image, and finally generates a segmentation image (Badrinarayanan et al., 2017).

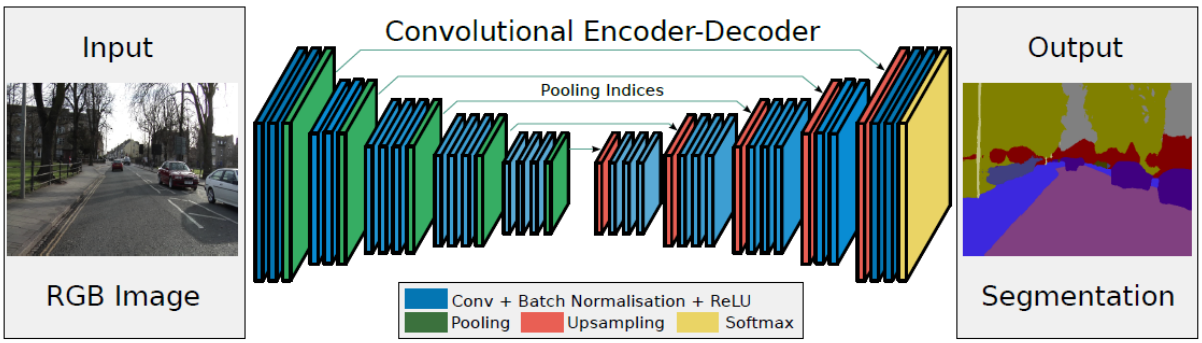


Figure 3-3. Architecture of SegNet (Badrinarayanan et al., 2017)

- The encoder is a series of convolutional networks. The network consists of a volume base layer, a pooling layer and a BatchNormalization layer. The volume base is responsible for obtaining the local features of the image, while the pooling layer samples the image and transmits the scale-invariant features to the next layer, while BN is mainly used to normalize the distribution of the training image and accelerate learning. In a nutshell, encoder classifies and analyzes the low-level local pixel values of an image to obtain higher-order semantic information.
- Decoder collects these semantic information and corresponds the same object to the corresponding pixel, each object is represented by a different colour. Now that the Encoder has all the object information and the general location information, the next step is to map these objects to specific pixels. The work is done by Decoder. Decoder carries out up-sampling on the feature image after the reduction and then carries out convolution processing on the image after the reduction. The purpose is to improve the geometric shape of the object and make up for the loss of detail caused by the object shrinking by pooling layer in Encoder.

The source information for the pooling points is stored in a method called Pooling Indices. In the pooling layer processing of the encoder network, it will record which region the 1x1 feature point comes from the original layer after each Pooling and this information is called Pooling Indices. Pooling Indices will be used in the decoder network. Since SegNet is a symmetric network, when the feature map needs to be upsampled in the decoder network, we can use the Pooling Indices of the corresponding Pooling layer to determine which position a 1x1 feature point should be placed in the 2x2 region after the upsampling.

3.2. GAN

3.2.1. ContourGAN

The traditional GAN used in pixel-wise classification consists of two parts, which are a generator and a discriminator. The generator is to generate realistic samples, and the discriminator is used to determine whether a sample is true or false. Based on the traditional architecture of GAN, a GAN-based method called ContourGAN was proposed to extract contours by foreground texture rather than noise pixels from the background(H. Yang et al., 2019). The traditional image-to-image models only consider the loss between the prediction and the real value of the ground and they ignore the similarity between the result data distribution and the real value of the ground. On this basis, this generative adversarial network is proposed to improve the accuracy of contour detection.

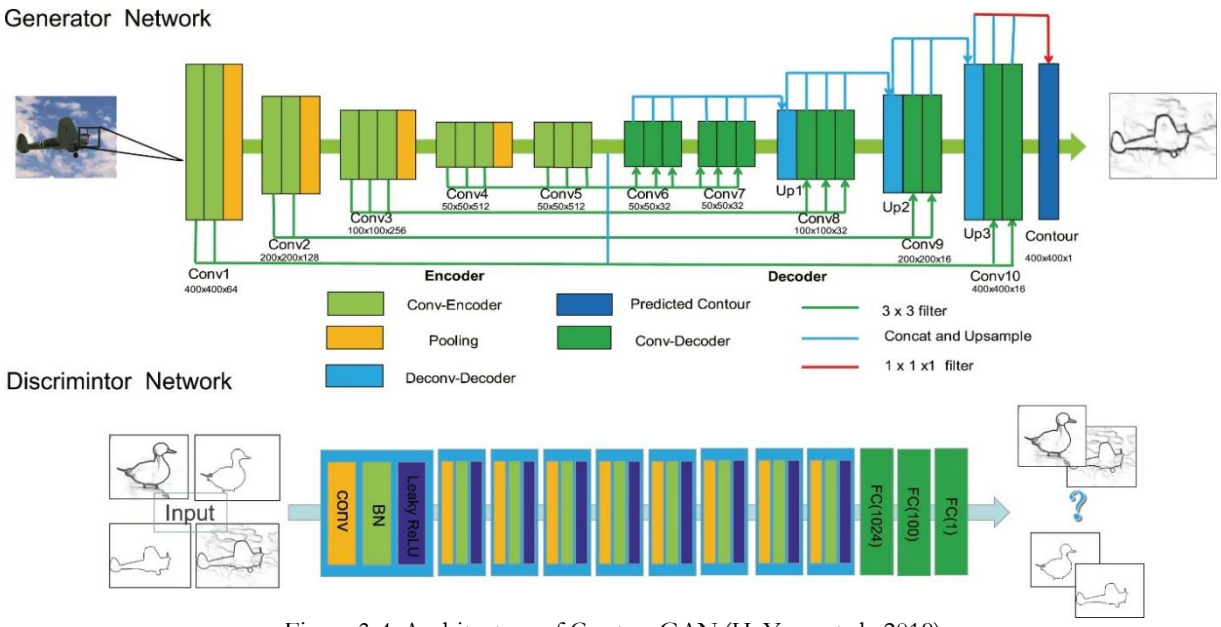


Figure 3-4. Architecture of ContourGAN (H. Yang et al., 2019)

3.2.1.1. Generator and discriminator

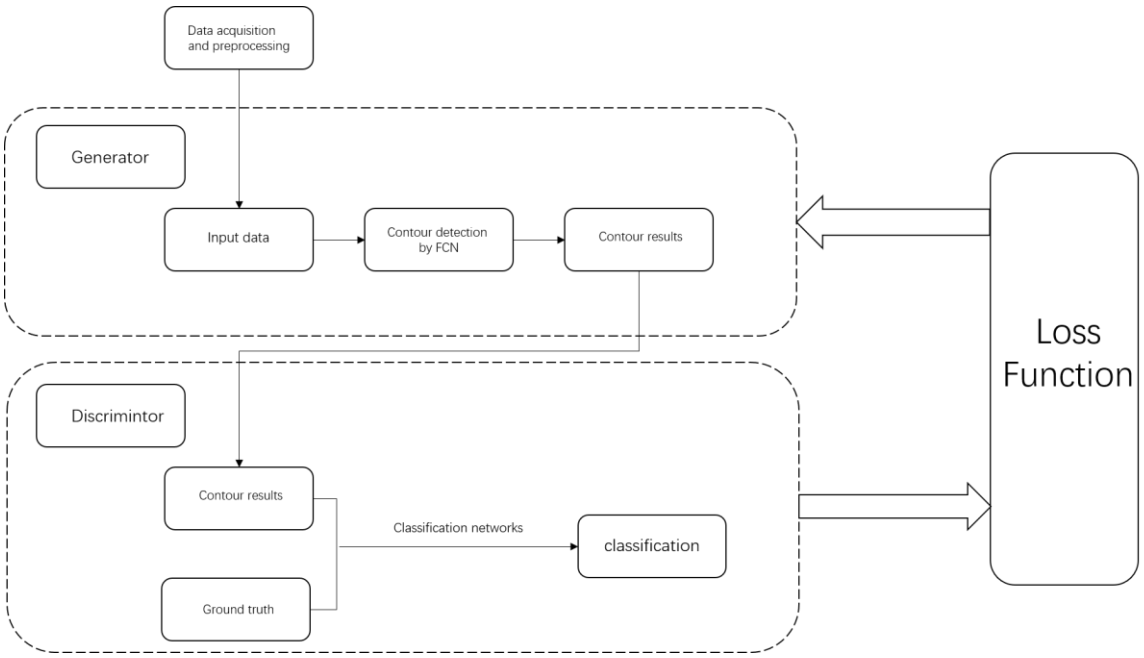


Figure 3-5. Generator and discriminator of ContourGAN

Figure 3-5 shows its architecture. In CountourGAN, the generator is an encoder-decoder model which aims to extract contour information from the input, and the discriminator is a CNN that calculates the loss of contour results based on ground truth. CountourGAN consists of a generator which is an encoder-decoder model to extract input image edge information and a discriminator which is a CNN to distinguish generated contours from the ground truth. The encoder conducts downstream sampling of the input image through the max-pooling layer, while the decoder conducts upstream sampling of the feature map calculated from the last layer of the encoder. The convolution layer is transposed to obtain the mapping to ensure the consistency with the input size. Each convolution layer in the encoder is connected to the corresponding convolution layer in the decoder.

3.2.1.2. Objective and loss function

- **Proposed method**

In the proposed method of ContourGAN, the *I* and *C* denotes the raw input images and labels. D and G denote the discriminator and generator networks which are used to solve the adversarial minimum problem. After we input training images I into the generator $G_{\theta G}$, the results will be input into the discriminator $D_{\theta D}$ to calculate the loss for the adversarial training process.

$$\min_{\theta G} \max_{\theta D} \log D_{\theta D}(C) + \log(1 - D_{\theta D}(G_{\theta G}(I))) \qquad \text{Equation 1}$$

- **Content loss**

Content loss of the ContourGAN is the pixel-wise value which calculates the weights of positive and negative pixels (edge and non-edge). γ and β in Equation 2 respectively denote the weights of non-edge and edge pixels. The classification loss is implemented as binary cross-entropy.

$$\tau(C, \tilde{C}) = -\frac{1}{N}\sum_{n=1}^{N} \gamma C_j \log \tilde{C_j} + \beta(1 - C_j)\log(1 - \tilde{C_j}) \quad \text{Equation 2}$$

- **Adversarial loss**

Adversarial loss of the ContourGAN is used to estimate the similarity between the predicted contour and available contour information. Thus, the adversarial loss will keep increasing if the discriminator could distinguish the predicted contour and ground truth. Equation 4 shows the entire loss function of ContourGAN consists of content loss, adversarial loss and regularization. $\propto$ denotes the weight of the adversarial loss in the Equation 4 and it could be modified by different datasets.

$$\tau_{adv}^{G} = \sum_{n=1}^{N} -\log D_{\theta D}\left(G_{\theta G}(I)\right) \quad \text{Equation 3}$$

$$\tau_{GAN} = \tau(C, \tilde{C}) + \propto \tau_{adv}^{G} + \lambda||\theta||^2 \quad \text{Equation 4}$$

3.2.2. Pixel2pixel GAN

However, one problem with this traditional GAN is that all the random noise may produce the same sample after each training. In order to produce different samples, the condition GAN comes out, namely cGAN. The cGAN is actually exactly the same as GAN in the training process, except that for the input sample, there is also an extra condition which can be a label or other generalized things (generalized label). In pixel2pixel GAN, we should train the generator and discriminator simultaneously during the adversarial process. In this specific case, we aim to take satellite images as real samples and try to generate the ground truth during the adversarial training.

3.2.2.1. Generator and discriminator

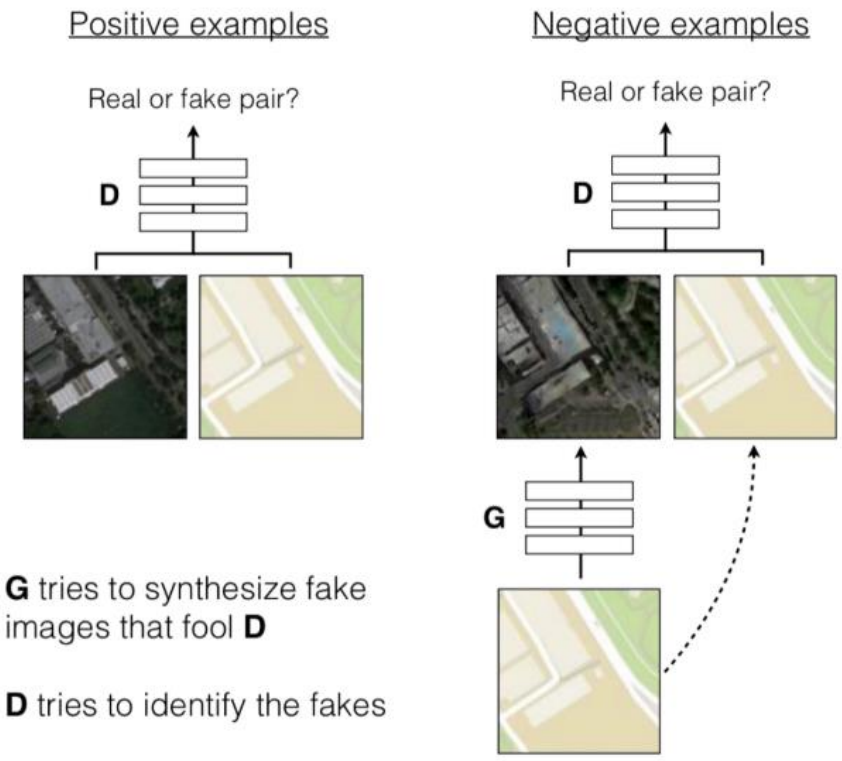


Figure 3-6. Generator and discriminator of pixel2pixel GAN (Isola et al., 2016)

From the figure 3-6, the thought of the entire framework based on cGAN. The input of the network is a data structure of two images which can be thought of as figure and label. From the discriminator part above, our white image acts as the role of noise. In other words, we can change the white image into a black image after the generator, and meanwhile, the black image and the white image are superimposed as false samples for training. So that we can find that for true and false samples if you want to make the discriminator cannot distinguish true and false, the generated black-figure must be more similar to the black image of the real sample to meet the conditions because the white figure (such as label) is the same. This is also why the combination with black and white as the sample input.

3.2.2.2. Objective and loss function

The final objective function is expressed as Equation 5, and it consists of common conditional GAN and a distance loss. The distance loss could be considered as a function about rebuilding the errors which mean the error between the white image generated by the generator (G(x,z)) and the original black image.

$$G = \arg\min_{G}\max_{D} \tau_{cGAN}(G, D) + \lambda\tau_{L1}(G) \qquad \text{Equation 5}$$

3.3. Accuracy assessment

To improve the delineation accuracy of smallholder farms using VHR images., an effective method to evaluate the accuracy of automatic delineation is required such as precision, error and degree of confidence because the aim is to improve the segmentation accuracy. In this part, the precision-recall framework could be a better choice which is a common method in contour detection evaluation (Martin et al., 2004b). Precision means how many selected items are relevant and recall means how many relevant items are selected. Beside, it introduces the harmonic mean of precision and recall values as f-measure/score. The equation of f-measure/score is shown as follows and α is usually evaluated as 1. Based on these parameters, precision-recall (PR) curve could be drawn and it visually shows the precision and recall performance of the model in the overall samples. The classification performance of a certain model will be better if the curve of the model is always above the curves of the other one.

$$F_1 = \frac{2*Precision \cdot Recall}{Precision + Recall} \qquad \text{Equation 6}$$

4. EXPERIMENT ANALYSIS

4.1. Experiment setup

This section describes how we prepare the data and what the architectures of these models are in this research. The experiment consists of two parts which are FCN and GAN. The FCN part is to adopt FCN network such as U-Net, PSPNet and SegNet to conduct the pixel classification of smallholder farm images. The GAN part is to find some methods which could combine with FCN to increase the result of the first part. Therefore, we conduct the contour detection based on the U-Net, PSPNet and SegNet to find the optimal architecture of FCN in this task. Then, we compare the results of the optimal architecture with the different training data by random rotation and GAN.

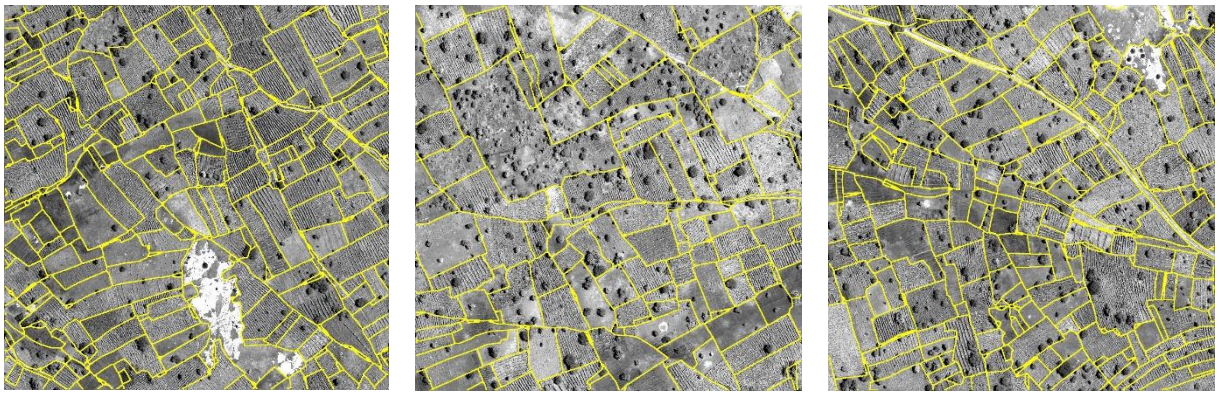
4.1.1. Data preparation

We aim to use FCN and GAN-based techniques to detect the sparse contours by VHR satellite images. Therefore, the first thing we should do is to collect the research data. The VHR image data are six tiles of 1000×1000 pixels from WorldView-3 data (acquired on September 25th 2015). TR data is for training and TS data is for accuracy assessment. In this dataset, the product should be atmospherically corrected, orthorectified, and coregistered. After that, it has been corrected that original dataset by human photo-interpretation and expanded to over 5700 field boundaries.

Table 4-1 Training and testing tiles

Tile	Train	Test
Tile No.1	TR1	--
Tile No.2	--	TS1
Tile No.3	TR2	--
Tile No.4	TR3	--
Tile No.5	--	TS2
Tile No.6	--	TS3

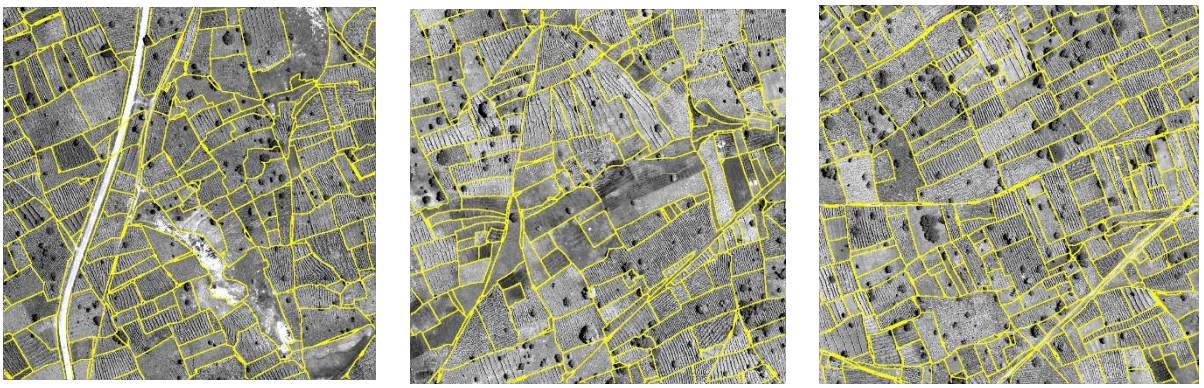
Figure 4-1 shows the satellite images and ground truths of Kofa which are the testing data in this research.



TR1 TR2 TR3

Figure 4-1. Training tiles of Kofa

Figure 4-2 shows the satellite images and ground truths of Kofa which are the testing data in this research.



TS1 TS2 TS3

Figure 4-2. Testing tiles of Kofa

4.1.2. Model parameters

In this section, we will discuss about the parameters of different models in this research.

In U-Net, PSPNet and SegNet, we adopt them in this research to conduct pixel-wise classification. Due to the limitation of the software usage, the number of training epochs is 200. Patch size of training tiles is 96×96 and batch size is 32. In addition, the learning rate is 0.0001 and loss function is Categorical cross-entropy which is often used in single label categorization.

In ContourGAN, we set the global learning rate to 0.00001, and weight decay to 0.00002. Due to the limitation of the software usage, the number of training epochs is set to 100. As to the weight of the adversarial loss, we set it to 0.01 as initial value.

In pixel2pixel GAN, we set the weight of updating effect to 0.5. The learning rate of this discriminator is set to 0.0002. In additional, we set the weight of the adversarial loss to 0.01 as initial value and the training epochs is set to 1000 to get a better model.

4.1.3. software

Google Colaboratory is an open research tool for machine learning development and research. This tool is now free to provide free GPU usage to most AI developers. The neural network framework of this research we used is Tensorflow and Keras. The name of GPU is Tesla T4 and the the memoery of it is 15079 MiB.

4.2. Results and analysis

This section will provide all the results of this research. The ground truth is shown as figure 8. Section 4.2.1 shows the results of FCN architectures (PSPNet, U-Net and SegNet). Section 4.2.2 shows the results of optimal FCN architecture in combination with GAN methods. Section 4.2.3 compares and analyze all the results of different methods in this research.

4.2.1. Results of FCN

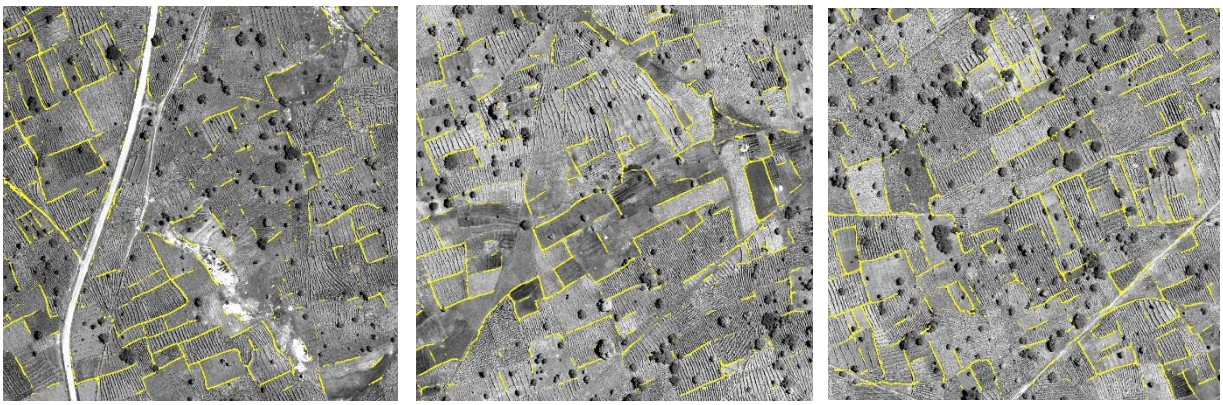
4.2.1.1. PSPNet

Table 4-2 shows the numerical results for PSPNet including precision, recall and F-score.

Table 4-2 Numerical results of PSPNet

Tiles	Precision	Recall	F1 score
TS1	0.746	0.542	0.602
TS2	0.734	0.543	0.601
TS3	0.748	0.548	0.611
Average	0.743	0.544	0.605

Figure 4-3 shows the classification results of PSPNet by three test tiles. The results of three test tiles look not very good here because the PSPNet provides global context prior for scenes in pixel-level resolution but it is not appropriate for complex satellite images here.



TS1 TS2 TS3

Figure 4-3. Results of PSPNet

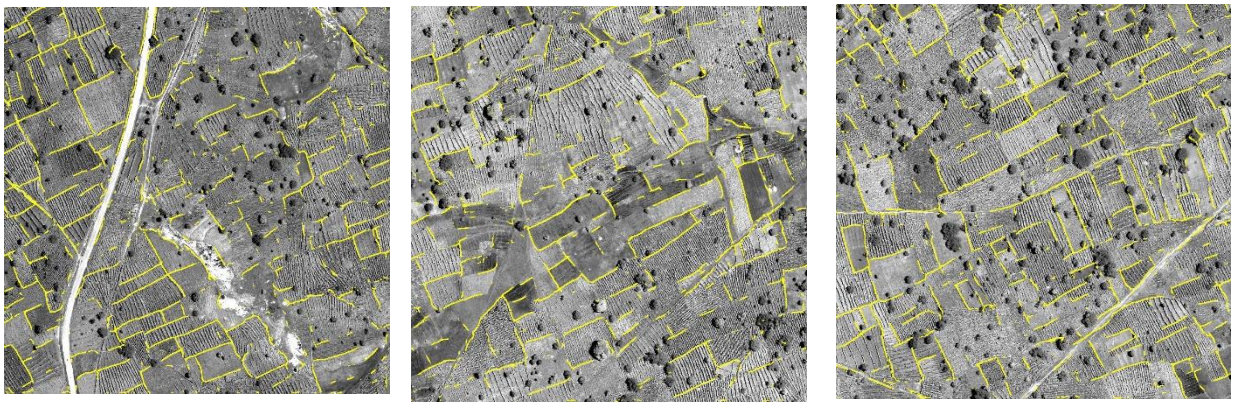
4.2.1.2. U-Net

Table 4-3 shows the numerical results for U-Net including precision, recall and F-score.

Table 4-3 Numerical results of U-Net

Tiles	Precision	Recall	F1 score
TS1	0.714	0.572	0.623
TS2	0.715	0.560	0.613
TS3	0.729	0.567	0.624
Average	0.719	0.566	0.620

Figure 4-4 shows the classification results of U-Net by three test tiles. The results of three test tiles look better than PSPNet because it is originally used for complex biomedical image segmentation and it could combine the low-resolution and high-resolution information.



TS1 TS2 TS3

Figure 4-4. Results of U-Net

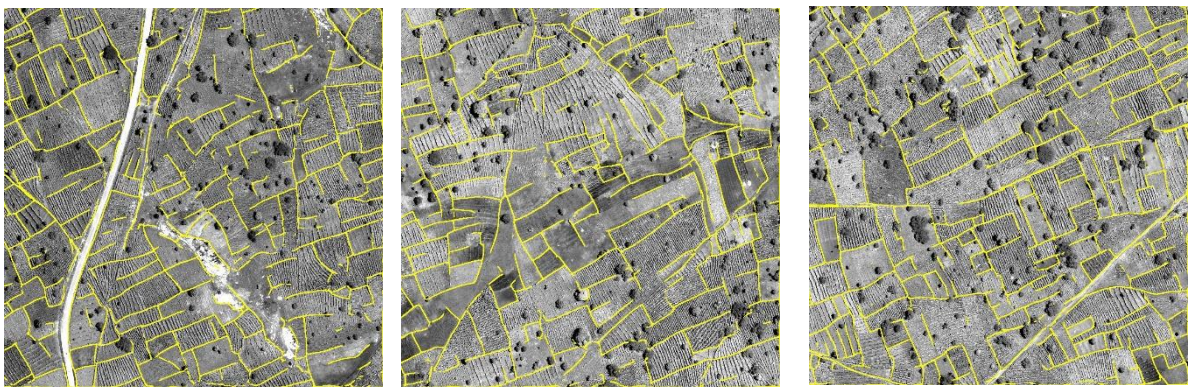
4.2.1.3. SegNet

Table 4-4 shows the numerical results for SegNet including precision, recall and F-score.

Table 4-4 Numerical results of SegNet

Tiles	Precision	Recall	F1 score
TS1	0.689	0.622	0.651
TS2	0.704	0.627	0.661
TS3	0.714	0.617	0.658
Average	0.702	0.622	0.657

Figure 4-5 shows the classification results of SegNet by three test tiles. Compared with these three architectures of FCN, the SegNet performs best in the limited training epochs. Thus, we will adopt the SegNet to conduct the pixel classification in this research.



TS1 TS2 TS3

Figure 4-5. Results of SegNet

4.2.2. Results of GAN

4.2.2.1. ContourGAN

During the experiments of ContourGAN, we fix the discriminator network and focus on the generator network. In the original architecture of the generator, it is an encoder-decoder FCN architecture with some modified layers. For examples, it removed the final pooling layer because it cannot capture meaningful information with its small size. Since the SegNet perform best based on the previous experiments, we decide to instead the original FCN architecture with SegNet to get contour results. Figure 4-6 shows the training curve of this GAN model. We could find that the loss of discriminator doesn't change during the adversarial training and the loss of generator keeps decreasing but it converges to one constant at last. It means the whole GAN model cannot perform adversarial training based on this loss function. The loss curve of generator which is used to extract contour information from the input means that the FCN part could get some contour results but it cannot modify and improve the predictions based on this loss function during the adversarial training. Therefore, the discriminator which is used to distinguish the predictions and ground truth doesn't work during the adversarial training. By tuning the weight of adversarial loss $\propto$, the results of this GAN model still perform not good.

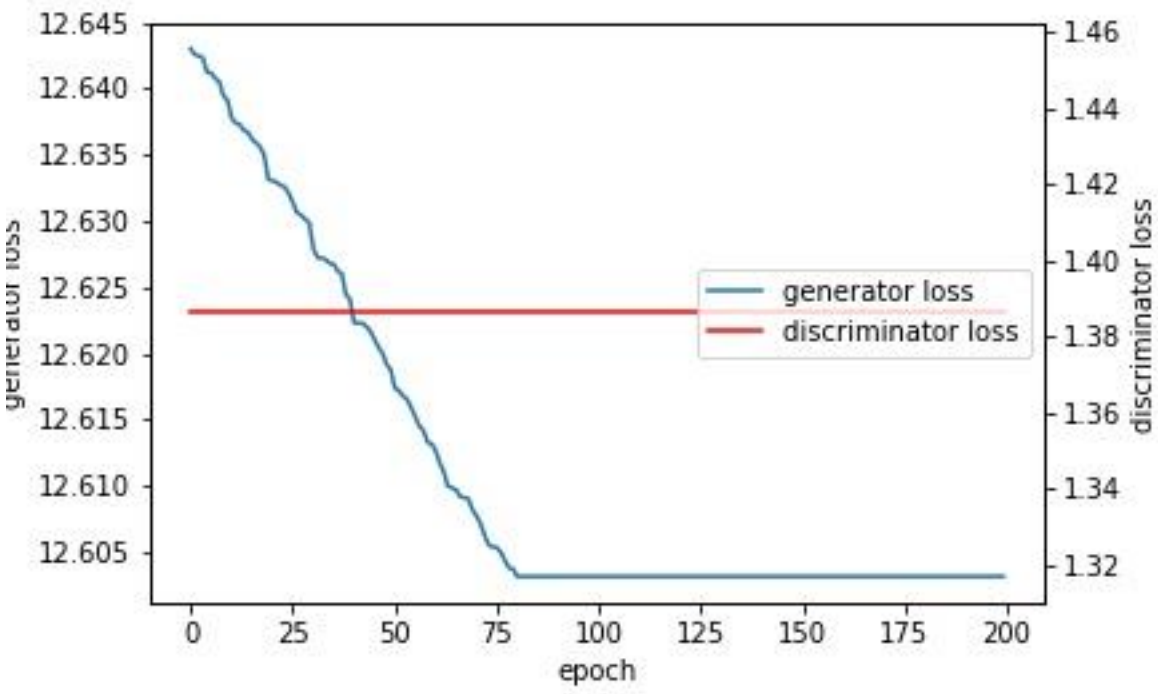


Figure 4-6. Training loss of ContourGAN

The results of the predictions are too sparse, as shown in figure 4-7. The reason for the bad results might be the loss function is not appropriate for the complex satellite images, especially smallholder farms. The original loss function is set for the segmentations of the natural image and we haven't found a better loss function for this research so far. In future studies, some other solutions about the loss functions which could pay more attention to the complex texture and pattern could be investigated to address this issue, especially in complex farm fields.

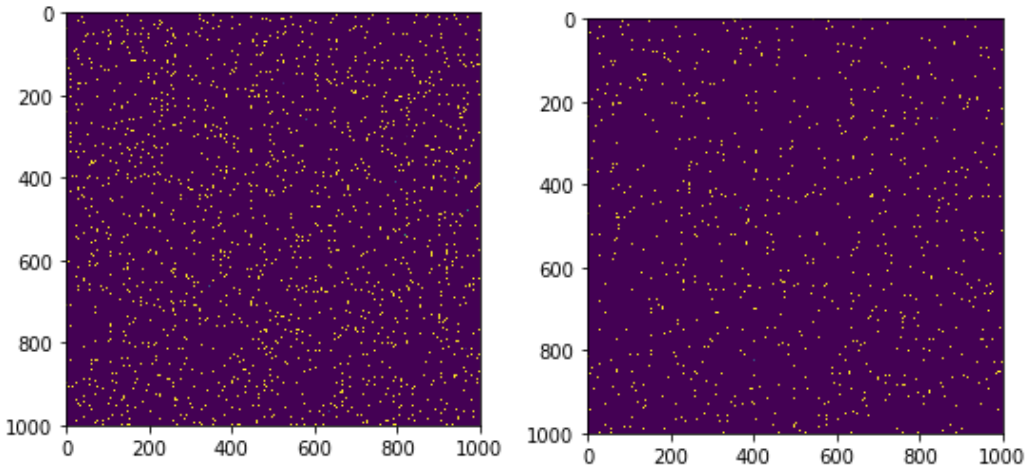


Figure 4-7. Results of ContourGAN

4.2.2.2. Pixel2pixel GAN

Since the ContourGAN performs not good, then we conduct the pixel2pixel GAN to get better training data. In this GAN model, the discriminator defines the relationship between the source images and predictions and it is optimized by binary cross entropy and the weight of updating effect is set to 0.5. The learning rate of this discriminator is set to 0.0002. The generator part of this GAN model is an U-Net architecture mentioned in section 3.1.1. The generator of this GAN model is updated by the adversarial loss functions and we set the weight of the adversarial loss to 0.01 as initial value. With the loss functions, the generator is forced toward generating more realistic images based on the source images during the adversarial training. In this specific research, we take the satellite images of small farmholder fields as real samples and the ground truth as targets. We set the number of training epochs to 1000 to get a better model. As we know, the training process of GAN model is usually not stable because it aims to find an equilibrium between the generator and discriminator models. During the training process, the results could be adjusted based on the original ground truth. Therefore, we need to save the weights and models every 100 epochs in this experiment to get better outputs and optimal model.

Figure 4-8 shows the training curve of pixel2pixel GAN in this research. We could find that the loss curve of generator converges to constant but the loss curve of discriminator is not unstable. Thus, we cannot judge it easily. We could review the results of the saved models and choose one of the best models.

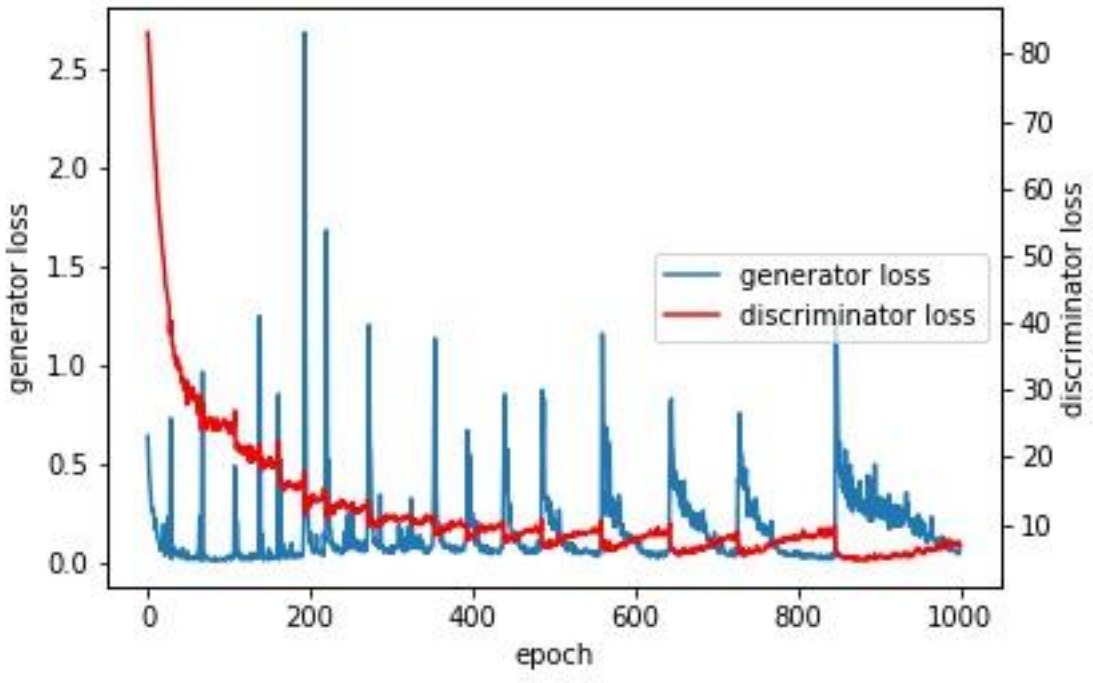


Figure 4-8. Training loss of Pixel2pixel GAN

Figure 4-9 shows one of the results using pixel2pixel GAN including source images, generated images and expected images. The changes between generated images and expected images are based on the pixels so that we cannot clarify it by our eyes. Then, we aim to conduct the optimal FCN architecture (SegNet) to get better contours by this modified ground truth.

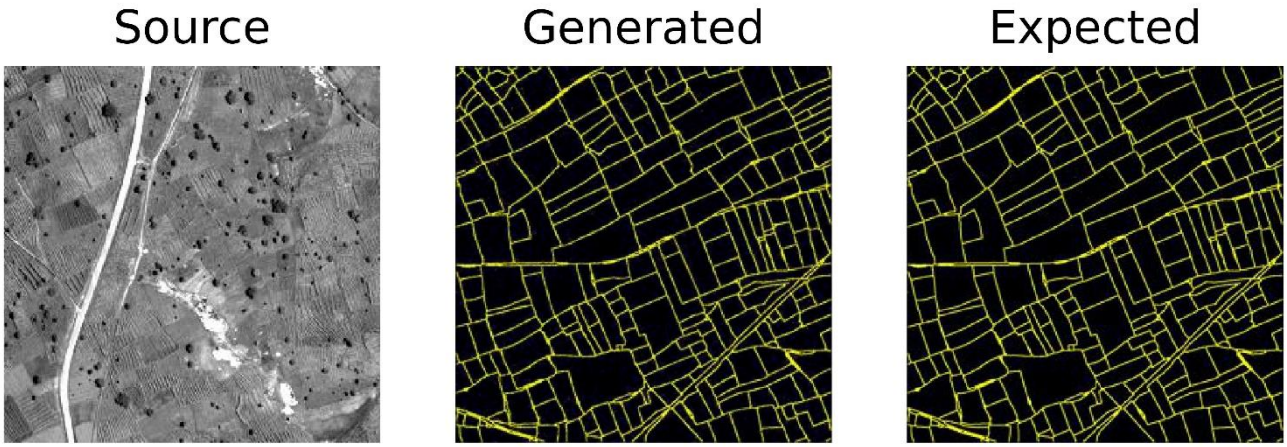


Figure 4-9. Results of Pixel2pixel GAN

To compare with the original training data, we set 9 contrast experiments. Firstly, we conduct the SegNet with one training tile (TR1), two training tiles (TR1, TR2) and three training tiles (TR1, TR2, TR3). Then,

we random selection 25% of the whole data to conduct the data augmentation by random rotation and pixel2pixel GAN. Then, we combine the contour result with the satellite images, as shown in figure 4-10.

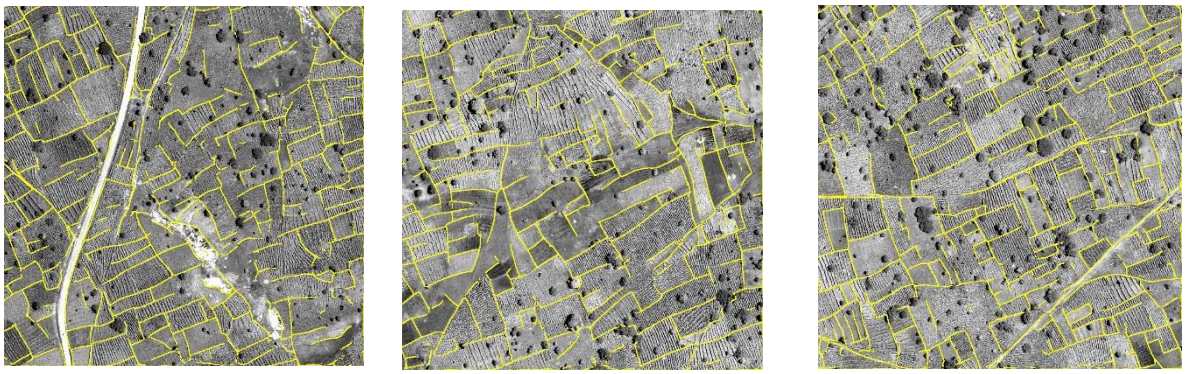


Figure 4-10. Result images of SegNet with GAN

4.2.3. Comparison of different methods

From the above, we conduct 11 experiments, and then we increase the proportion of selected data to perform data augmentation. In this section, we compare and analyze all the results of different methods in this research. As shown in the table, we could get the following conclusions:

1. Among the three FCN architecture of this research, PSPNet performs worst and SegNet performs best.
2. The more original training data, the better results we will get.
3. Random rotation and GAN could get higher the accuracy of results than original training data.
4. The higher proportion of augmented data with GAN doesn't mean higher accuracy because it depends on the performance of GAN training. If the GAN is sufficiently well trained and get good results, we will get better training data. Otherwise, the results of GAN cannot increase the accuracy as we expect.

Table 4-5 Different methods and accuracy assessment

Method	TS1	TS2	TS3
PSPNet	0.602	0.601	0.611
U-Net	0.623	0.613	0.624
SegNet – TR1	0.641	0.653	0.648
SegNet – TR1 with random rotation	0.643	0.654	0.650
SegNet – TR1 with pixel2piexlGAN	0.645	0.655	0.652
SegNet – TR1&2	0.648	0.656	0.657
SegNet – TR1&2 with random rotation	0.649	0.658	0.657
SegNet – TR1&2 with pixel2piexlGAN	0.650	0.659	0.658
SegNet – TR1&2&3	0.651	0.661	0.658
SegNet – TR1&2&3 with random rotation (25%)	0.653	0.665	0.659
SegNet – TR1&2&3 with pixel2piexlGAN (25%)	0.654	0.666	0.661
SegNet – TR1&2&3 with random rotation (50%)	0.654	0.663	0.656
SegNet – TR1&2&3 with pixel2piexlGAN (50%)	0.652	0.661	0.654

5. DISCUSSION

Contour detection task could be divided into three parts which are sparse contour detection, closed segment extraction and final contour generation. This research will pay more attention to propose an improved algorithm in the first part which is sparse contour detection by GAN-based techniques. Deep convolutional neural networks often have a good performance in this part according to the previous researches, especially the FCN. FCN significantly improves the segmentation accuracy and speed with multi-resolution layer combinations but the appearance of the GANs also gives people more reflection on whether we could improve the segmentation results of the FCN. So far, few researchers have introduced GAN in VHR satellite images, especially in contour detection. Smallholder farm fields are worthy of study because of their specific geographic features and it is helpful for agricultural management. Besides, GAN-based techniques are usually applied in small natural images and the introduction of GAN-based technology for VHR satellite images in contour detection tasks is also one of the main contributions. However, how to combine FCN with GAN for automatic delineation of smallholder farms from VHR satellite images is still a difficult problem that needs to be solved. This research could give a new perspective on how to improve the existing models with nascent concepts. Therefore, this research will focus on FCN in combination with the GAN to improve the delineation accuracy of smallholder farm fields. This section discusses the strengths and weakness of the methods in this research. Section 5.1 analyses the results of FCN methods. Section 5.2 analyses the results of GAN methods.

5.1. FCN methods

In this research, we aim to adopt different architectures of FCN to acquire the optimal network for contour detection task in the smallholder farm fields. Based on the primary goal of this research, we propose three state-of-the-art architectures of FCN (U-Net, PSPNet, SegNet) to find the optimal architecture in the contour detection task of smallholder farm fields. After the relevant experiments with three different architectures, we conclude that the SegNet architecture acquires the optimal results with the same hyperparameters and training epochs. During the training process, the U-Net acquires the second-best results and PSPNet acquires the worst results because of the complex geographic information and mixed-cropping systems in smallholder farm fields. SegNet obtains the best contour results because it uses pooling slices of max-pooling layers to conduct un-linear upsampling rather than deconvolution or transposed convolutions. Therefore, the SegNet is the optimal architecture among the three state-of-the-art FCN architectures for the contour detection task in smallholder farm fields.

5.2. GAN methods

In this research, we aim to introduce GAN technology in combination with the optimal FCN architecture to increase the accuracy of pixel-wise classification. We try to adopt GAN as a training method and introduce the ContourGAN. The generator aims to conduct the pixel-wise classification and the discriminator aims to assess the difference between the prediction result and ground truth. Then, we could adjust the generator by the loss function. However, the results of ContourGAN are not good as mentioned in section 4.2.2.1 because the loss function of the ContourGAN is suited for the natural images rather than complex smallholder farms. It cannot adjust the subtle delineation in the complex fields. In this research, we have tried many other loss functions and still cannot adjust a suitable loss function for the study area. In theory, if we could adjust a suitable loss function of smallholder farm fields, the results of ContourGAN will be a better one. In future studies, some other solutions about the loss functions which could pay more attention to the complex texture and pattern could be investigated to address this issue, especially in complex farm fields. On the other hand, we try to adopt GAN as a data

augmentation method and introduce the pixel2pixelGAN. The generator of it aims to generate the fake images based on the original data and the discriminator aims to identify these fake images. During the training process, the results could be adjusted based on the original ground truth. Based on this method, we could adjust some complex textual fields of smallholder farm fields and input them as training data. However, the training process of pixel2pixel GAN model is not stable and converge because it aims to find an equilibrium between the generator and discriminator. Thus, we should save the model and weights by some epochs to get better results. Comparing with original and random-rotated training data, the results of pixel2pixelGAN just get a bit higher accuracy. We consider the reason is that the GAN method generates the potential features from the original data rather than new data, so the GAN methods cannot produce new information from the original data space. Thus, if the GAN methods are sufficiently well trained, the results could be a bit better. Otherwise, the results will be worse if the GAN training is not performed well.

6. CONCLUSION

In this research, we aim to propose an automated algorithm by GAN-based techniques to improve the delineation accuracy of smallholder farms from VHR images. Based on the improved results of sparse contours, this research will then extract the closed segments and generate the final delineation by some specific methods. The VHR image data of Kofa are six tiles of 1000×1000 pixels from WorldView-3 data (acquired on September 25th 2015). The research consists of two parts which are to find the optimal FCN architecture and the potential GAN methods.

For FCN architectures, we select three state-of-the-art FCN architectures which are PSPNet, U-Net and SegNet from the computer vision field. For this part, firstly we adopt the U-Net for pixel-wise classification with original training data. With the same training data and hyperparameters, then we conduct contour detection with PSPNet and SegNet. Comparing three experiments with three different SOTA FCN architectures, the SegNet performs best. Therefore, we could adopt SegNet as the optimal FCN architecture in this research. Based on the SegNet architecture, we could try to conduct potential GAN methods in combination with SegNet to increase the accuracy of contour detection.

For GAN methods, we select two potential GAN methods for this specific task. The first one is ContourGAN. Its generator aims to get contour results and the discriminator aims to assess the difference between the prediction result and ground truth. However, the results of ContourGAN are not well because of the loss function is not suitable for complex smallholder farm fields and we cannot find an appropriate one. In future studies, some other solutions about the loss functions which could pay more attention to the complex texture and pattern could be investigated to address this issue, especially in complex farm fields. The second potential GAN method is pixel2pixelGAN. Its generator aims to generate the fake images based on the original data and the discriminator aims to identify these fake images. The pixel2pixelGAN could adjust the images based on the original ground truth. Therefore, the results of pixel2pixelGAN are more suitable for the complex textual patterns. Then, we compare the contour detection results with different training data as mentioned in section 4.2.3. Although the results are just improved a little, we could consider it as an optional method for other researches.

7. REFERENCES

Adams, R., & Bischof, L. (1994). Seeded Region Growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*(6), 641–647. https://doi.org/10.1109/34.295913

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(12), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

Bergado, J. R., Persello, C., & Gevaert, C. (2016). A deep learning approach to the classification of sub-decimetre resolution aerial images. *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 1516–1519. https://doi.org/10.1109/IGARSS.2016.7729387

Chen, P. C., & Pavlidis, T. (1979). Segmentation by texture using a co-occurrence matrix and a split-and-merge algorithm. *Computer Graphics and Image Processing*, *10*(2), 172–182. https://doi.org/10.1016/0146-664X(79)90049-2

Christ, P. F., Elshaer, M. E. A., Ettlinger, F., Tatavarty, S., Bickel, M., Bilic, P., Rempfler, M., Armbruster, M., Hofmann, F., D'Anastasi, M., Sommer, W. H., Ahmadi, S.-A., & Menze, B. H. (2016). *Automatic Liver and Lesion Segmentation in CT Using Cascaded Fully Convolutional Neural Networks and 3D Conditional Random Fields* (pp. 415–423). Springer, Cham. https://doi.org/10.1007/978-3-319-46723-8_48

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(5), 603–619. https://doi.org/10.1109/34.1000236

Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., & Pal, C. (2016). *The Importance of Skip Connections in Biomedical Image Segmentation* (pp. 179–187). Springer, Cham. https://doi.org/10.1007/978-3-319-46976-8_19

García-Pedrero, A., Gonzalo-Martín, C., & Lillo-Saavedra, M. (2017). *International Journal of Remote Sensing A machine learning approach for agricultural parcel delineation through agglomerative segmentation A machine learning approach for agricultural parcel delineation through agglomerative segmentation*. https://doi.org/10.1080/01431161.2016.1278312

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative Adversarial Nets* (pp. 2672–2680). http://papers.nips.cc/paper/5423-generative-adversarial-nets

Hong, Y., Hwang, U., Yoo, J., & Yoon, S. (2017). *How Generative Adversarial Networks and Their Variants Work: An Overview*. https://doi.org/10.1145/3301282

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2016). Image-to-Image Translation with Conditional Adversarial Networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-January*, 5967–5976. http://arxiv.org/abs/1611.07004

Li, J., Liang, X., Wei, Y., Xu, T., Feng, J., & Yan, S. (2017). *Perceptual Generative Adversarial Networks for Small Object Detection*. http://arxiv.org/abs/1706.05274

Lin, D., Fu, K., Wang, Y., Xu, G., & Sun, X. (2017). MARTA GANs: Unsupervised Representation Learning for Remote Sensing Image Classification. *IEEE Geoscience and Remote Sensing Letters*, *14*(11), 2092–2096. https://doi.org/10.1109/LGRS.2017.2752750

Long, J., Shelhamer, E., & Darrell, T. (2014). *Fully Convolutional Networks for Semantic Segmentation*. http://arxiv.org/abs/1411.4038

Lowder, S. K., Skoet, J., & Raney, T. (2016). The Number, Size, and Distribution of Farms, Smallholder Farms, and Family Farms Worldwide. *World Development*, *87*, 16–29. https://doi.org/10.1016/J.WORLDDEV.2015.10.041

Luc, P., Couprie, C., Chintala, S., & Verbeek, J. (2016). *Semantic Segmentation using Adversarial Networks*. http://arxiv.org/abs/1611.08408

Lucio, D. R., Laroca, R., Severo, E., Britto, A. S., & Menotti, D. (2018). *Fully Convolutional Networks and Generative Adversarial Networks Applied to Sclera Segmentation*. http://arxiv.org/abs/1806.08722

Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2016). Fully convolutional neural networks for remote sensing image classification. *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 5071–5074. https://doi.org/10.1109/IGARSS.2016.7730322

Martin, D. R., Fowlkes, C. C., & Malik, J. (2004a). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(5), 530–549. https://doi.org/10.1109/TPAMI.2004.1273918

Martin, D. R., Fowlkes, C. C., & Malik, J. (2004b). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(5), 530–549. https://doi.org/10.1109/TPAMI.2004.1273918

Mueller, M., Segl, K., & Kaufmann, H. (2004). Edge- and region-based segmentation technique for the extraction of large, man-made objects in high-resolution satellite imagery. *Pattern Recognition*, *37*(8), 1619–1628. https://doi.org/10.1016/J.PATCOG.2004.03.001

Persello, C., Tolpekin, V. A., Bergado, J. R., & de By, R. A. (2019). Delineation of agricultural fields in smallholder farms from satellite images using fully convolutional networks and combinatorial grouping. *Remote Sensing of Environment*, *231*, 111253. https://doi.org/10.1016/J.RSE.2019.111253

Radford, A., Metz, L., & Chintala, S. (2015). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. http://arxiv.org/abs/1511.06434

Roberts, L. (1963). *Machine perception of three-dimensional solids*. https://dspace.mit.edu/bitstream/handle/1721.1/11589/33959125-MIT.pdf

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9351*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Shelhamer, E., Long, J., & Darrell, T. (2017). Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 640–651. https://doi.org/10.1109/TPAMI.2016.2572683

Sobel, I. (1972). *Camera models and machine perception*. http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1972/CS/CS0016.pdf

Sun, Q., & Bourennane, S. (2019). Unsupervised feature extraction based on improved Wasserstein generative adversarial network for hyperspectral classification. In S. Negahdaripour, E. Stella, D. Ceglarek, & C. Möller (Eds.), *Multimodal Sensing: Technologies and Applications* (Vol. 11059, p. 32). SPIE. https://doi.org/10.1117/12.2527466

Tiwari, P. S., Pande, H., Kumar, M., & Dadhwal, V. K. (2009). Potential of IRS P-6 LISS IV for agriculture field boundary delineation. *Journal of Applied Remote Sensing*, *3*(1), 033528. https://doi.org/10.1117/1.3133306

Turker, M., & Kok, E. H. (2013). Field-based sub-boundary extraction from remote sensing imagery using perceptual grouping. *ISPRS Journal of Photogrammetry and Remote Sensing*, *79*, 106–121. https://doi.org/10.1016/J.ISPRSJPRS.2013.02.009

Xie, S., & Tu, Z. (2015). *Holistically-Nested Edge Detection* (pp. 1395–1403). http://openaccess.thecvf.com/content_iccv_2015/html/Xie_Holistically-Nested_Edge_Detection_ICCV_2015_paper.html

Yang, H., Li, Y., Yan, X., & Cao, F. (2019). ContourGAN: Image contour detection with generative adversarial network. *Knowledge-Based Systems*, *164*, 21–28. https://doi.org/10.1016/J.KNOSYS.2018.09.033

Yang, J., Price, B., Cohen, S., Lee, H., & Yang, M.-H. (2016). *Object Contour Detection With a Fully Convolutional Encoder-Decoder Network* (pp. 193–202). http://openaccess.thecvf.com/content_cvpr_2016/html/Yang_Object_Contour_Detection_CVPR_2016_paper.html

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). PSPNet. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 6230–6239. https://doi.org/10.1109/CVPR.2017.660

Zhu, L., Chen, Y., Ghamisi, P., & Benediktsson, J. A. (2018). Generative Adversarial Networks for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, *56*(9), 5046–5063. https://doi.org/10.1109/TGRS.2018.2805286

8. APPENDIX

8.1. Appendix 1

Table 8-1 Architecture of U-Net

Layer (type)	Output shape	Param #	Connected to
Img (Inputlayer)	96×96×3	--	
Conv2d_1 (Conv2d)	96×96×16	448	Img
batch_normalization_1 (BatchNo)	96×96×16	64	Conv2d_1
Activation_1 (activation)	96×96×16	--	batch_normalization_1
Conv2d_2 (Conv2d)	96×96×16	2320	Activation_1
batch_normalization_2 (BatchNo)	96×96×16	64	Conv2d_2
Activation_2 (activation)	96×96×16	--	batch_normalization_2
max_pooling2d_1 (MaxPooling2D)	48×48×16	--	Activation_2
dropout_1 (Dropout)	48×48×16	--	max_pooling2d_1
Conv2d_3 (Conv2d)	48×48×32	4640	dropout_1
batch_normalization_3 (BatchNo)	48×48×32	128	Conv2d_3
Activation_3 (activation)	48×48×32	--	batch_normalization_3
Conv2d_4 (Conv2d)	48×48×32	9248	Activation_3
batch_normalization_4 (BatchNo)	48×48×32	128	Conv2d_4
Activation_4 (activation)	48×48×32	--	batch_normalization_4
max_pooling2d_2 (MaxPooling2D)	24×24×32	--	Activation_4
dropout_2 (Dropout)	24×24×32	--	max_pooling2d_2
Conv2d_5 (Conv2d)	24×24×64	18496	dropout_2
batch_normalization_5 (BatchNo)	24×24×64	256	Conv2d_5
Activation_5 (activation)	24×24×64	--	batch_normalization_5
Conv2d_6 (Conv2d)	24×24×64	36928	Activation_5
batch_normalization_6 (BatchNo)	24×24×64	256	Conv2d_6
Activation_6 (activation)	24×24×64	--	batch_normalization_6
max_pooling2d_3 (MaxPooling2D)	12×12×64	--	Activation_6
dropout_3 (Dropout)	12×12×64	--	max_pooling2d_3
Conv2d_7 (Conv2d)	12×12×128	73856	dropout_3
batch_normalization_7 (BatchNo)	12×12×128	512	Conv2d_7
Activation_7 (activation)	12×12×128	--	batch_normalization_7
Conv2d_8 (Conv2d)	12×12×128	147584	Activation_7
batch_normalization_8 (BatchNo)	12×12×128	512	Conv2d_8
Activation_8 (activation)	12×12×128	--	batch_normalization_8
max_pooling2d_4 (MaxPooling2D)	6×6×128	--	Activation_8
dropout_4 (Dropout)	6×6×128	--	max_pooling2d_4
Conv2d_9 (Conv2d)	6×6×256	295168	dropout_4
batch_normalization_9 (BatchNo)	6×6×256	1024	Conv2d_9
Activation_9 (activation)	6×6×256	--	batch_normalization_9
Conv2d_10 (Conv2d)	6×6×256	590080	Activation_9
batch_normalization_10 (BatchNo)	6×6×256	1024	Conv2d_10
Activation_10 (activation)	6×6×256	--	batch_normalization_10
conv2d_transpose_1 (Conv2DTrans)	12×12×128	295040	Activation_10

concatenate_1 (Concatenate)	12×12×256	--	conv2d_transpose_1 Activation_8
dropout_5 (Dropout)	12×12×256	--	concatenate_1
Conv2d_11 (Conv2d)	12×12×128	295040	dropout_5
batch_normalization_11 (BatchNo)	12×12×128	512	Conv2d_11
Activation_11 (activation)	12×12×128	--	batch_normalization_11
Conv2d_12 (Conv2d)	12×12×128	147584	Activation_11
batch_normalization_12 (BatchNo)	12×12×128	512	Conv2d_12
Activation_12 (activation)	12×12×128	--	batch_normalization_12
conv2d_transpose_2 (Conv2DTrans)	24×24×64	73792	Activation_12
concatenate_2 (Concatenate)	24×24×128	--	conv2d_transpose_2 Activation_6
dropout_6 (Dropout)	24×24×128	--	concatenate_2
Conv2d_13 (Conv2d)	24×24×64	73792	dropout_6
batch_normalization_13 (BatchNo)	24×24×64	256	Conv2d_13
Activation_13 (activation)	24×24×64	--	batch_normalization_13
Conv2d_14 (Conv2d)	24×24×64	36928	Activation_13
batch_normalization_14 (BatchNo)	24×24×64	256	Conv2d_14
Activation_14 (activation)	24×24×64	--	batch_normalization_14
conv2d_transpose_3 (Conv2DTrans)	48×48×32	18464	Activation_14
concatenate_3 (Concatenate)	48×48×64	--	conv2d_transpose_3 Activation_4
dropout_7 (Dropout)	48×48×64	--	concatenate_3
Conv2d_15 (Conv2d)	48×48×32	18464	dropout_7
batch_normalization_15 (BatchNo)	48×48×32	128	Conv2d_15
Activation_15 (activation)	48×48×32	--	batch_normalization_15
Conv2d_16 (Conv2d)	48×48×32	9248	Activation_15
batch_normalization_16 (BatchNo)	48×48×32	128	Conv2d_16
Activation_16 (activation)	48×48×32	--	batch_normalization_16
conv2d_transpose_4 (Conv2DTrans)	96×96×16	4624	Activation_16
concatenate_4 (Concatenate)	96×96×32	--	conv2d_transpose_4 Activation_2
dropout_8 (Dropout)	96×96×32	--	concatenate_4
Conv2d_17 (Conv2d)	96×96×16	4624	dropout_8
batch_normalization_17 (BatchNo)	96×96×16	64	Conv2d_17
Activation_17 (activation)	96×96×16	--	batch_normalization_17
Conv2d_18 (Conv2d)	96×96×16	2320	Activation_17
batch_normalization_18 (BatchNo)	96×96×16	64	Conv2d_18
Activation_18 (activation)	96×96×16	--	batch_normalization_18
conv2d_19 (Conv2D)	96×96×2	34	Activation_18

8.2. Appendix 2

Table 8-2. Architecture of PSPNet

Layer (type)	Output shape	Param #	Connected to
Img (Inputlayer)	96×96×3	--	
Conv2d_1 (Conv2d)	48×48×64	1792	Img
batch_normalization_1 (BatchNo)	48×48×64	256	Conv2d_1
Activation_1 (activation)	48×48×64	--	batch_normalization_1
Conv2d_2 (Conv2d)	48×48×64	36928	Activation_1
batch_normalization_2 (BatchNo)	48×48×64	256	Conv2d_2
Activation_2 (activation)	48×48×64	--	batch_normalization_2
Conv2d_3 (Conv2d)	48×48×128	73856	Activation_2
batch_normalization_3 (BatchNo)	48×48×128	512	Conv2d_3
Activation_3 (activation)	48×48×128	--	batch_normalization_3
max_pooling2d_1 (MaxPooling2D)	24×24×128	--	Activation_3
Conv2d_4 (Conv2d)	24×24×64	8256	max_pooling2d_1
batch_normalization_4 (BatchNo)	24×24×64	256	Conv2d_4
Activation_4 (activation)	24×24×64	--	batch_normalization_4
Conv2d_5 (Conv2d)	24×24×64	36928	Activation_4
batch_normalization_5 (BatchNo)	24×24×64	256	Conv2d_5
Activation_5 (activation)	24×24×64	--	batch_normalization_5
Conv2d_6 (Conv2d)	24×24×256	16640	Activation_5
Conv2d_7 (Conv2d)	24×24×256	33024	max_pooling2d_1
batch_normalization_6 (BatchNo)	24×24×256	1024	Conv2d_6
batch_normalization_7 (BatchNo)	24×24×256	1024	Conv2d_7
add_1 (Add)	24×24×256	--	batch_normalization_6 batch_normalization_7
activation_6 (Activation)	24×24×256	--	Add_1
Conv2d_8 (Conv2d)	24×24×64	16448	activation_6
batch_normalization_8 (BatchNo)	24×24×64	256	Conv2d_8
Activation_7 (activation)	24×24×64	--	batch_normalization_8
Conv2d_9 (Conv2d)	24×24×64	36928	Activation_7
batch_normalization_9 (BatchNo)	24×24×64	256	Conv2d_9
Activation_8 (activation)	24×24×64	--	batch_normalization_9
Conv2d_10 (Conv2d)	24×24×256	16640	Activation_8
batch_normalization_10 (BatchNo)	24×24×256	1024	Conv2d_10
add_2 (Add)	24×24×256	--	batch_normalization_10 activation_6
Activation_9 (activation)	24×24×256	--	Add_2
Conv2d_11 (Conv2d)	24×24×64	16448	activation_9
batch_normalization_11 (BatchNo)	24×24×64	256	Conv2d_11
Activation_10 (activation)	24×24×64	--	batch_normalization11
Conv2d_12 (Conv2d)	24×24×64	36928	Activation_10
batch_normalization_12 (BatchNo)	24×24×64	256	Conv2d_12
Activation_11 (activation)	24×24×64	--	batch_normalization_12
Conv2d_13 (Conv2d)	24×24×256	16640	Activation_11
batch_normalization_13 (BatchNo)	24×24×256	1024	Conv2d_13

add_3 (Add)	24×24×256	--	batch_normalization_13 activation_9
Activation_12 (activation)	24×24×256	--	Add_3
Conv2d_14 (Conv2d)	24×24×128	32896	activation_12
batch_normalization_14 (BatchNo)	24×24×128	512	Conv2d_14
Activation_13 (activation)	24×24×128	--	batch_normalization_14
Conv2d_15 (Conv2d)	12×12×128	147584	Activation_13
batch_normalization_15 (BatchNo)	12×12×128	512	Conv2d_15
Activation_14 (activation)	12×12×128	--	batch_normalization_15
Conv2d_16 (Conv2d)	12×12×512	66048	Activation_14
Conv2d_17 (Conv2d)	12×12×512	131584	Activation_12
batch_normalization_16 (BatchNo)	12×12×512	2048	Conv2d_16
batch_normalization_17 (BatchNo)	12×12×512	2048	Conv2d_17
add_4 (Add)	12×12×512	--	batch_normalization_16 batch_normalization_17
Activation_15 (activation)	12×12×512	--	Add_4
Conv2d_18 (Conv2d)	12×12×128	65664	activation_15
batch_normalization_18 (BatchNo)	12×12×128	512	Conv2d_18
Activation_16 (activation)	12×12×128	--	batch_normalization_18
Conv2d_19 (Conv2d)	12×12×128	147584	Activation_16
batch_normalization_19 (BatchNo)	12×12×128	512	Conv2d_19
Activation_17 (activation)	12×12×128	--	batch_normalization_19
Conv2d_20 (Conv2d)	12×12×512	66048	Activation_17
batch_normalization_20 (BatchNo)	12×12×512	2048	Conv2d_20
add_5 (Add)	12×12×512	--	batch_normalization_20 activation_15
Activation_18 (activation)	12×12×512	--	Add_5
Conv2d_21 (Conv2d)	12×12×128	65664	activation_18
batch_normalization_21 (BatchNo)	12×12×128	512	Conv2d_21
Activation_19 (activation)	12×12×128	--	batch_normalization_21
Conv2d_22 (Conv2d)	12×12×128	147584	Activation_19
batch_normalization_22 (BatchNo)	12×12×128	512	Conv2d_22
Activation_20 (activation)	12×12×128	--	batch_normalization_22
Conv2d_23 (Conv2d)	12×12×512	66048	Activation_20
batch_normalization_23 (BatchNo)	12×12×512	2048	Conv2d_23
add_6 (Add)	12×12×512	--	batch_normalization_23 activation_18
Activation_21 (activation)	12×12×512	--	Add_6
Conv2d_24 (Conv2d)	12×12×128	65664	activation_21
batch_normalization_24 (BatchNo)	12×12×128	512	Conv2d_24
Activation_22 (activation)	12×12×128	--	batch_normalization_24
Conv2d_25 (Conv2d)	12×12×128	147584	Activation_22
batch_normalization_25 (BatchNo)	12×12×128	512	Conv2d_25
Activation_23 (activation)	12×12×128	--	batch_normalization_25
Conv2d_26 (Conv2d)	12×12×512	66048	Activation_23
batch_normalization_26 (BatchNo)	12×12×512	2048	Conv2d_26
add_7 (Add)	12×12×512	--	batch_normalization_26

			activation_21
Activation_24 (activation)	12×12×512	--	Add_7
Conv2d_27 (Conv2d)	12×12×256	131328	activation_24
batch_normalization_27 (BatchNo)	12×12×256	1024	Conv2d_27
Activation_25 (activation)	12×12×256	--	batch_normalization_27
Conv2d_28 (Conv2d)	12×12×256	590080	Activation_25
batch_normalization_28 (BatchNo)	12×12×256	1024	Conv2d_28
Activation_26 (activation)	12×12×256	--	batch_normalization_28
Conv2d_29 (Conv2d)	12×12×1024	263168	Activation_26
Conv2d_30 (Conv2d)	12×12×1024	525312	Activation_24
batch_normalization_29 (BatchNo)	12×12×1024	4096	Conv2d_29
batch_normalization_30 (BatchNo)	12×12×1024	4096	Conv2d_30
add_8 (Add)	12×12×1024	--	batch_normalization_29 batch_normalization_30
Activation_27 (activation)	12×12×1024	--	Add_8
Conv2d_31 (Conv2d)	12×12×256	262400	activation_27
batch_normalization_31 (BatchNo)	12×12×256	1024	Conv2d_31
Activation_28 (activation)	12×12×256	--	batch_normalization_31
Conv2d_32 (Conv2d)	12×12×256	590080	Activation_28
batch_normalization_32 (BatchNo)	12×12×256	1024	Conv2d_32
Activation_29 (activation)	12×12×256	--	batch_normalization_32
Conv2d_33 (Conv2d)	12×12×1024	263168	Activation_29
batch_normalization_33 (BatchNo)	12×12×1024	4096	Conv2d_33
add_9 (Add)	12×12×1024	--	batch_normalization_33 activation_27
Activation_30 (activation)	12×12×1024	--	Add_9
Conv2d_34 (Conv2d)	12×12×256	262400	activation_30
batch_normalization_44 (BatchNo)	12×12×256	1024	Conv2d_34
Activation_31 (activation)	12×12×256	--	batch_normalization_34
Conv2d_35 (Conv2d)	12×12×256	590080	Activation_31
batch_normalization_35 (BatchNo)	12×12×256	1024	Conv2d_35
Activation_32 (activation)	12×12×256	--	batch_normalization_35
Conv2d_36 (Conv2d)	12×12×1024	263168	Activation_32
batch_normalization_36 (BatchNo)	12×12×1024	4096	Conv2d_36
add_10 (Add)	12×12×1024	--	batch_normalization_36 activation_30
Activation_33 (activation)	12×12×1024	--	Add_10
Conv2d_37 (Conv2d)	12×12×256	262400	activation_33
batch_normalization_37 (BatchNo)	12×12×256	1024	Conv2d_37
Activation_34 (activation)	12×12×256	--	batch_normalization_37
Conv2d_38 (Conv2d)	12×12×256	590080	Activation_34
batch_normalization_38 (BatchNo)	12×12×256	1024	Conv2d_38
Activation_35 (activation)	12×12×256	--	batch_normalization_38
Conv2d_39 (Conv2d)	12×12×1024	263168	Activation_35
batch_normalization_39 (BatchNo)	12×12×1024	4096	Conv2d_39
add_11 (Add)	12×12×1024	--	batch_normalization_39 activation_33

Activation_36 (activation)	12×12×1024	--	Add_11
Conv2d_40 (Conv2d)	12×12×256	262400	activation_36
batch_normalization_40 (BatchNo)	12×12×256	1024	Conv2d_40
Activation_37 (activation)	12×12×256	--	batch_normalization_40
Conv2d_41 (Conv2d)	12×12×256	590080	Activation_37
batch_normalization_41 (BatchNo)	12×12×256	1024	Conv2d_41
Activation_38 (activation)	12×12×256	--	batch_normalization_41
Conv2d_42 (Conv2d)	12×12×1024	263168	Activation_38
batch_normalization_42 (BatchNo)	12×12×1024	4096	Conv2d_42
add_12 (Add)	12×12×1024	--	batch_normalization_42 activation_36
Activation_39 (activation)	12×12×1024	--	Add_12
Conv2d_43 (Conv2d)	12×12×256	262400	activation_39
batch_normalization_43 (BatchNo)	12×12×256	1024	Conv2d_43
Activation_40 (activation)	12×12×256	--	batch_normalization_43
Conv2d_44 (Conv2d)	12×12×256	590080	Activation_40
batch_normalization_44 (BatchNo)	12×12×256	1024	Conv2d_44
Activation_41 (activation)	12×12×256	--	batch_normalization_44
Conv2d_45 (Conv2d)	12×12×1024	263168	Activation_41
batch_normalization_45 (BatchNo)	12×12×1024	4096	Conv2d_45
add_13 (Add)	12×12×1024	--	batch_normalization_45 activation_39
Activation_42 (activation)	12×12×1024	--	Add_13
Conv2d_46 (Conv2d)	12×12×512	524800	activation_42
batch_normalization_46 (BatchNo)	12×12×512	2048	Conv2d_46
Activation_43 (activation)	12×12×512	--	batch_normalization_46
Conv2d_47 (Conv2d)	12×12×512	2359808	Activation_43
batch_normalization_47 (BatchNo)	12×12×512	2048	Conv2d_47
Activation_44 (activation)	12×12×512	--	batch_normalization_47
Conv2d_48 (Conv2d)	12×12×2048	1050624	Activation_44
Conv2d_49 (Conv2d)	12×12×2048	2099200	Activation_45
batch_normalization_48 (BatchNo)	12×12×2048	8192	Conv2d_48
batch_normalization_49 (BatchNo)	12×12×2048	8192	Conv2d_49
add_14 (Add)	12×12×2048	--	batch_normalization_48 batch_normalization_49
Activation_45 (activation)	12×12×2048	--	Add_14
Conv2d_50 (Conv2d)	12×12×512	1049088	activation_45
batch_normalization_50 (BatchNo)	12×12×512	2048	Conv2d_50
Activation_46 (activation)	12×12×512	--	batch_normalization_50
Conv2d_51 (Conv2d)	12×12×512	2359808	Activation_46
batch_normalization_51 (BatchNo)	12×12×512	2048	Conv2d_51
Activation_47 (activation)	12×12×512	--	batch_normalization_51
Conv2d_52 (Conv2d)	12×12×2048	1050624	Activation_47
batch_normalization_52 (BatchNo)	12×12×2048	8192	Conv2d_52
add_15 (Add)	12×12×2048	--	batch_normalization_52 activation_45
Activation_48 (activation)	12×12×2048	--	Add_15

Conv2d_53 (Conv2d)	12×12×512	1049088	activation_48
batch_normalization_53 (BatchNo)	12×12×512	2048	Conv2d_53
Activation_49 (activation)	12×12×512	--	batch_normalization_53
Conv2d_54 (Conv2d)	12×12×512	2359808	Activation_49
batch_normalization_54 (BatchNo)	12×12×512	2048	Conv2d_54
Activation_50 (activation)	12×12×512	--	batch_normalization_54
Conv2d_55 (Conv2d)	12×12×2048	1050624	Activation_50
batch_normalization_55 (BatchNo)	12×12×2048	8192	Conv2d_55
add_16 (Add)	12×12×2048	--	batch_normalization_55 activation_48
Activation_51 (activation)	12×12×2048	--	Add_16
average_pooling2d_1 (AveragePool)	1×1×2048	--	activation_51
average_pooling2d_2 (AveragePool)	2×2×2048	--	activation_51
average_pooling2d_3 (AveragePool)	3×3×2048	--	activation_51
average_pooling2d_4 (AveragePool)	6×6×2048	--	activation_51
Conv2d_56 (Conv2d)	1×1×512	1049088	average_pooling2d_1
Conv2d_57 (Conv2d)	2×2×512	1049088	average_pooling2d_2
Conv2d_58 (Conv2d)	3×3×512	1049088	average_pooling2d_3
Conv2d_59 (Conv2d)	6×6×512	1049088	average_pooling2d_4
lambda_1 (Lambda)	12×12×512	--	conv2d_56
lambda_2 (Lambda)	12×12×512	--	conv2d_57
lambda_3 (Lambda)	12×12×512	--	conv2d_58
lambda_4 (Lambda)	12×12×512	--	conv2d_59
concatenate_1 (Concatenate)	12×12×4096	--	activation_51 lambda_1 lambda_2 lambda_3 lambda_4
Conv2d_60 (Conv2d)	12×12×512	18874880	concatenate_1
batch_normalization_56 (BatchNo)	12×12×512	2048	Conv2d_60
Activation_52 (activation)	12×12×512	--	batch_normalization_56
dropout_1 (Dropout)	12×12×512	--	Activation_52
Conv2d_61 (Conv2d)	12×12×2	1026	dropout_1
conv2d_transpose_1 (Conv2DTrans)	96×96×2	1026	Conv2d_61
Activation_53 (activation)	96×96×2	--	conv2d_transpose_1

8.3. Appendix 3

Table 8-3. Architecture of SegNet

Layer (type)	Output shape	Param #
Img (Inputlayer)	96×96×3	--
Block1_conv1 (Conv2D)	96×96×64	1792
Block1_conv2 (Conv2D)	96×96×64	36928
Block1_pool (MaxPooling2D)	48×48×64	--
Block2_conv1 (Conv2D)	48×48×128	73856
Block2_conv2 (Conv2D)	48×48×128	147584
Block2_pool (MaxPooling2D)	24×24×128	--
Block3_conv1 (Conv2D)	24×24×256	295168
Block3_conv2 (Conv2D)	24×24×256	590080
Block3_conv3 (Conv2D)	24×24×256	590080
Block3_pool (MaxPooling2D)	12×12×256	--
Block4_conv1 (Conv2D)	12×12×512	1180160
Block4_conv2 (Conv2D)	12×12×512	2359808
Block4_conv3 (Conv2D)	12×12×512	2359808
Block4_pool (MaxPooling2D)	6×6×512	--
Block5_conv1 (Conv2D)	6×6×512	2359808
Block5_conv2 (Conv2D)	6×6×512	2359808
Block5_conv3 (Conv2D)	6×6×512	2359808
Block5_pool (MaxPooling2D)	3×3×512	--
dropout_1 (Dropout)	3×3×512	--
de_pool2d_1 (DePool2D)	6×6×512	--
Conv2d_1 (Conv2d)	6×6×512	2359808
batch_normalization_1 (BatchNo)	6×6×512	2048
Activation_1 (activation)	6×6×512	--
Conv2d_2 (Conv2d)	6×6×512	2359808
batch_normalization_2 (BatchNo)	6×6×512	2048
Activation_2 (activation)	6×6×512	--
Conv2d_3 (Conv2d)	6×6×512	2359808
batch_normalization_3 (BatchNo)	6×6×512	2048
Activation_3 (activation)	6×6×512	--
dropout_2 (Dropout)	6×6×512	--
de_pool2d_2 (DePool2D)	12×12×512	--
Conv2d_4 (Conv2d)	12×12×512	2359808
batch_normalization_4 (BatchNo)	12×12×512	2048
Activation_4 (activation)	12×12×512	--
Conv2d_5 (Conv2d)	12×12×512	2359808
batch_normalization_5 (BatchNo)	12×12×512	2048
Activation_5 (activation)	12×12×512	--
Conv2d_6 (Conv2d)	12×12×512	2359808
batch_normalization_6 (BatchNo)	12×12×512	2048
Activation_6 (activation)	12×12×512	--
dropout_3 (Dropout)	12×12×512	--

de_pool2d_3 (DePool2D)	24×24×512	--
Conv2d_7 (Conv2d)	24×24×256	1179904
batch_normalization_7 (BatchNo)	24×24×256	1024
Activation_7 (activation)	24×24×256	--
Conv2d_8 (Conv2d)	24×24×256	590080
batch_normalization_8 (BatchNo)	24×24×256	1024
Activation_8 (activation)	24×24×256	--
dropout_4 (Dropout)	24×24×256	--
de_pool2d_4 (DePool2D)	48×48×256	--
Conv2d_9 (Conv2d)	48×48×128	295040
batch_normalization_9 (BatchNo)	48×48×128	512
Activation_9 (activation)	48×48×128	--
Conv2d_10 (Conv2d)	48×48×128	147584
batch_normalization_10 (BatchNo)	48×48×128	512
Activation_10 (activation)	48×48×128	--
de_pool2d_5 (DePool2D)	96×96×128	--
Conv2d_11 (Conv2d)	96×96×64	73792
batch_normalization_11 (BatchNo)	96×96×64	256
Activation_11 (activation)	96×96×64	--
Conv2d_12 (Conv2d)	96×96×64	36928
batch_normalization_12 (BatchNo)	96×96×64	256
Activation_12 (activation)	96×96×64	--
Conv2d_13 (Conv2d)	96×96×2	130
Activation_13 (activation)	96×96×2	--

8.4. Appendix 4

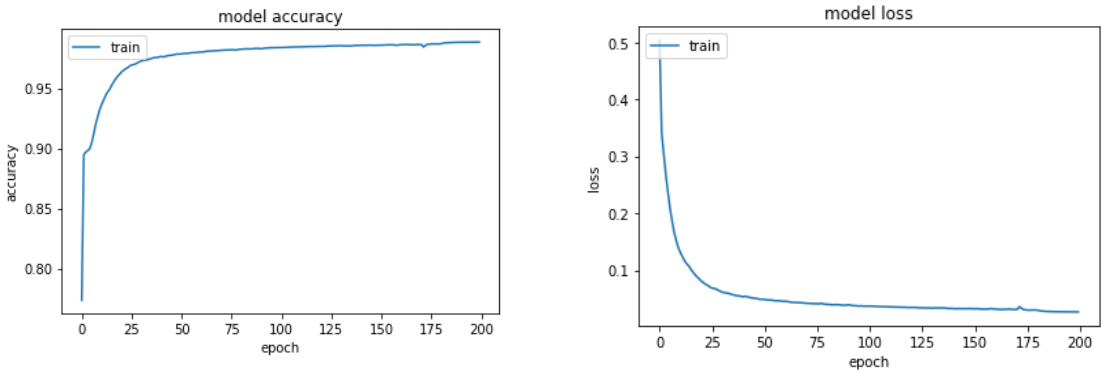


Figure 8-1 PSPNet training curve

8.5. Appendix 5

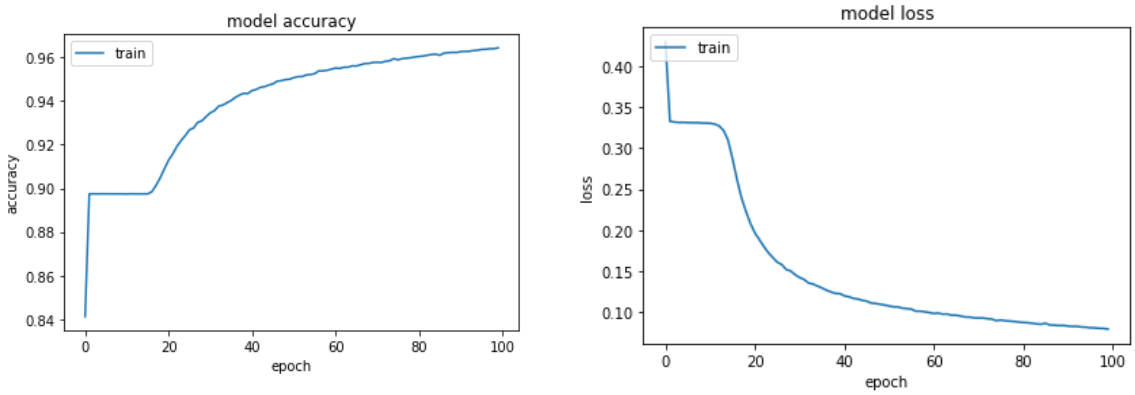


Figure 8-2 U-Net training curve

8.6. Appendix 6

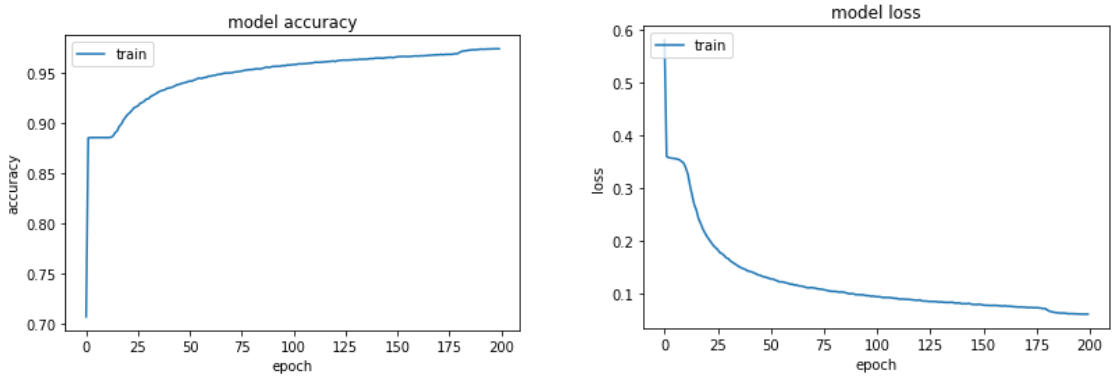


Figure 8-3 SegNet training curve