remote sensing

MDPI

Article

Unsupervised Domain Adaptation Using Generative Adversarial Networks for Semantic Segmentation of Aerial Images

Bilel Benjdira [1,2,*], Yakoub Bazi [3], Anis Koubaa [4] and Kais Ouni [2]

[1] Robotics and internet of things Laboratory, College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia
[2] Research Laboratory Smart Electricity & ICT, SEICT, LR18ES44, National Engineering School of Carthage, University of Carthage, Carthage 1054, Tunisia; kais.ouni@enicarthage.rnu.tn or kais.ouni@esti.rnu.tn
[3] Computer Engineering Department, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia; ybazi@ksu.edu.sa
[4] Prince Sultan University, Saudi Arabia/Gaitech Robotics, China/CISTER, INESC-TEC, ISEP, Polytechnic Institute of Porto, 4200-465 Porto, Portugal; akoubaa@psu.edu.sa
* Correspondence: bbenjdira@psu.edu.sa

Received: 25 April 2019; Accepted: 1 June 2019; Published: 7 June 2019

Abstract: Segmenting aerial images is of great potential in surveillance and scene understanding of urban areas. It provides a mean for automatic reporting of the different events that happen in inhabited areas. This remarkably promotes public safety and traffic management applications. After the wide adoption of convolutional neural networks methods, the accuracy of semantic segmentation algorithms could easily surpass 80% if a robust dataset is provided. Despite this success, the deployment of a pretrained segmentation model to survey a new city that is not included in the training set significantly decreases accuracy. This is due to the domain shift between the source dataset on which the model is trained and the new target domain of the new city images. In this paper, we address this issue and consider the challenge of domain adaptation in semantic segmentation of aerial images. We designed an algorithm that reduces the domain shift impact using generative adversarial networks (GANs). In the experiments, we tested the proposed methodology on the International Society for Photogrammetry and Remote Sensing (ISPRS) semantic segmentation dataset and found that our method improves overall accuracy from 35% to 52% when passing from the Potsdam domain (considered as source domain) to the Vaihingen domain (considered as target domain). In addition, the method allows efficiently recovering the inverted classes due to sensor variation. In particular, it improves the average segmentation accuracy of the inverted classes due to sensor variation from 14% to 61%.

Keywords: convolutional neural networks; semantic segmentation; aerial imagery; domain adaptation; gener ative adversarial networks

1. Introduction

Semantic segmentation is an image analysis task that assigns for every pixel in an input image a label that describes the class of its enclosing region. Beyond image classification and object detection, semantic segmentation is the highest-level image analysis task that allows a complete scene understanding of the whole input image.

Semantic segmentation was referred in many remote sensing works as pixel-wise classification. Semantic segmentation term is more used in computer vision, and it is being more and more adopted in remote sensing. Semantic segmentation can be used in aerial imagery in a variety of

potential applications, like urban area monitoring and planning, traffic management and analysis, hazard detection and avoidance, and so on. This potential is boosted by the increasing adoption of unmanned aerial vehicles (UAVs). UAVs make the surveillance of inhabited areas easier due to their flexibility, great mobility, and the high resolution images that they can gather and stream in real time. These images can be automatically processed by accurate semantic segmentation algorithms to substantially reinforce the ability to analyze and describe the surveyed scenes automatically.

The progress of semantic segmentation algorithms was delayed years ago by the low accuracy of traditional approaches of image analysis algorithms based on the extraction of hand-crafted features. However, since the emergence of highly descriptive feature extractors like convolutional neural networks, the whole area of image analysis has shown a significant increase in accuracy. In fact, since 2012 [1], convolutional neural networks (CNNs) have shown an outstanding efficiency in computer vision. This advancement enhanced the areas of semantic segmentation algorithms. Recently, several CNN- based architectures have shown their efficiency in this task, such as fully connected networks (FCN) [2], SegNet [3], UNet [4], PSPNets [5], and DeepLab [6]. If a robust dataset is provided and semantically labeled, training one of the state-of-the-art models could lead easily to an accuracy that exceeds 80% [7].

Despite this notable success made in the area of semantic segmentation algorithms, a great challenge is hampering their implementation in real use cases. In fact, if we train a model on a specific dataset, the accuracy will be high when applying this model on images belonging to the same domain of the train set (lighting conditions, sensor type, resolution, object representation). However, if we try to apply this model to segment images acquired under different conditions, the performance falls dramatically due to the domain shift between the images used in the source domain (used during the training) training and the target domain. To illustrate this fact, we conducted an experiment where we chose a state-of-the-art segmentation algorithm (DeepLab v3 plus [8]) which is trained on the International Society for Photogrammetry and Remote Sensing (ISPRS) Potsdam benchmark dataset [9], and we applied it for segmenting a random image from the ISPRS Vaihingen benchmark dataset. A drop in global accuracy from 85% to 35% was observed. Figure 1 shows a typical situation in which we have a trained model on a specific source domain and we want to use this model to segment another domain. The domains have different characteristics (resolution per pixel changed from 5 cm to 9 cm, image information changed from a red-green-blue sensor to a near-infrared-red-green, location changed from Potsdam to Vaihingen).

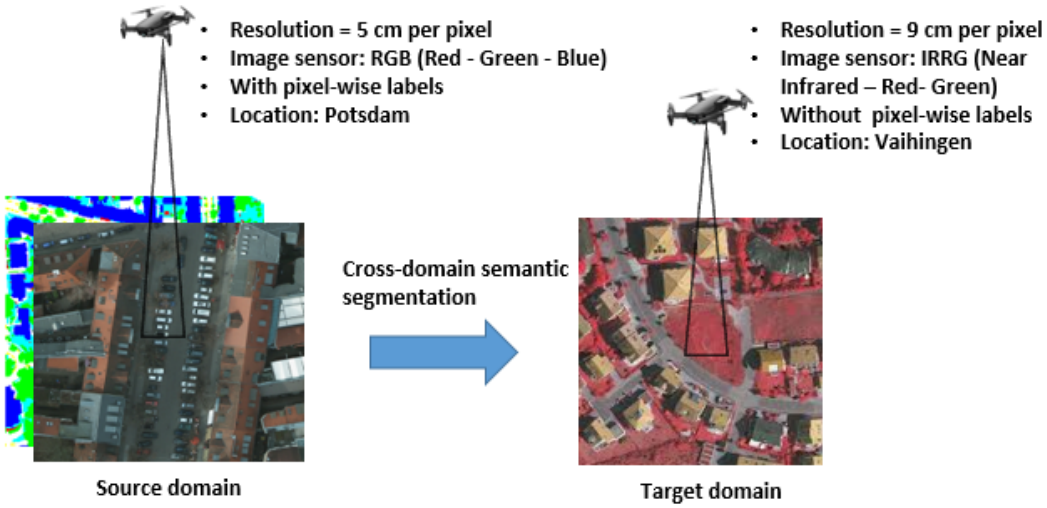

Figure 1. Cross-domain semantic segmentation in aerial imagery.

The ordinary solution to cope with this intriguing limitation is to make a new semantically labeled dataset on the target domain and to train the model on it. This solution is very costly and

impractical. In fact, collecting a large dataset of pixel-labeled images for the targeted city of interest will be time-consuming and expensive. Indeed, pixel-labeling of Cityscapes images (size 2040 by 1016 pixels) takes 90 min on average [10]. Remote sensing is more time-demanding as it contains objects from different sizes (small-sized objects like cars and roads need more attention and effort in the labeling process). To reduce human efforts in manual pixel-wise classification, a number of solutions have been introduced, like synthesizing data from 3D rendered images [11,12] or weakly supervised labeling [13–15]. However, these approaches still have limitations, as they also require significant human efforts. Moreover, they have some drawbacks (like domain shift from 3D rendered images to real images in synthetic data solutions and imprecise boundaries in weakly supervised solutions). This is why it is highly fruitful to invest in an automatic domain adaptation solution.

Domain adaptation is the machine learning field that aims at learning from a source data distribution how to improve the performance of a model on a different target data distribution. It addresses reducing the domain shift problem between the source domain dataset used in training and the target domain dataset. For this purpose, we typically design a mapping function between the source domain data and the target domain data. Recent domain adaptation techniques have used deep learning models to train this mapping function [16–19]. Domain adaptation techniques could also consolidate this mapping function by adding some modifications on the model itself to get a correlated feature level with the target domain dataset.

Inspired by recent advances in generative adversarial networks (GANs) [20,21], we developed an algorithm for domain adaptation for aerial imagery based on GANs. The objective of our method was to handle the scenario presented in Figure 1 and similar cases. We aimed to add the ability for a semantic segmentation model to handle domains that are different from the source domain with minimal cost and maximum accuracy. Our method was divided into two steps. The first step considered the process of converting the images of the dataset from the source domain to the target domain. This was done using a GAN model trained using a cyclic-loss to map between two sets, one taken from the source domain and the other from the target domain. We adopted this approach to eliminate the need for a paired set of images, which may be time-consuming. The second step was to fine-tune the already trained semantic model using the mapped version of the dataset associated with the original labels. After the fine-tuning process, the model will improve its ability to semantically label images taken from the target domain. The major contributions of our work can be presented as follows: (1) To the best of our knowledge, no previous works have addressed the problem of domain adaptation for semantic segmentation in aerial imagery using GANs. (2) We demonstrated that our approach mitigates the domain shift problem for cross-domain semantic segmentation in aerial imagery, which allows the portability of the semantic segmentation model over different image domains. (3) We validated the method on the ISPRS semantic labelling dataset by making cross-domain semantic segmentation between the Potsdam dataset and Vaihingen dataset. (4) We introduced GANs as a promising solution for analysis of aerial imagery.

The rest of the paper is organized as follows: Section 2 gives an overview of the related works in area of domain adaptation in semantic segmentation. Section 3 makes an introduction to GANs. Section 4 describes our proposed method. Section 5 presents the experimental details we used to test our method. Section 6 discusses its efficiency for domain adaptation in aerial imagery. Section 7 concludes our work and deduces the contribution we made in this paper.

2. Related Works

In this section, we discuss the related works on domain adaptation in semantic segmentation. When applying a machine learning algorithm, we generally assume that the training data and the test data belong to the same underlying distribution. In real scenarios, though, we face some discordance between them. This discordance decreases the efficiency of the model outside its training domain. Domain adaptation is a separate field in machine learning that aims to rectify this discordance and help the model to be better generalized to test domains.

The efforts on domain adaptation in image analysis have focused on classification and regression tasks [22], like trying to train models on online photos to classify objects in real world [23]. Recent works are mostly oriented towards improving the adaptability of deep learning algorithms [16,17,24–26].

Concerning the domain adaptation for semantic segmentation, many works on this field focused on simulated data [12,27–31]. In fact, they expected to use domain adaptation to improve the segmentation efficiency on real images by training models on synthetic data. Among the first works that treated domain adaptation on semantic segmentation, we can find FCNs in the wild [32] which employed a pixel-level adversarial loss to guide the model towards learning the domain-invariant features. The goal is to make the adversarial classifier not differentiate between source and target domains to equalize its performance on both domains. Hoffman et al. proposed CyCADA [27] as another method that converts the source images (synthetic data) to the style of the target (real datat) using CycleGAN. The converted images are then fed to the segmentation model to improve its performance on the target images. Zhang et al. [33] proposed a curriculum-style learning approach to minimize the domain shift. They concluded properties of the target data by combining the learning of the local distributions over landmark superpixels with the learning of global label distribution. Then they trained the segmentation network by regularizing it to follow those concluded properties. Chen et al. [34] proposed ROAD (reality-oriented adaptation) by designing two losses to align the source and the target domains. The first is called target-guided distillation loss, and the second is a spatially-aware adaptation loss. The feature map of the image is divided into grids. Then, a maximum mean discrepancy loss is calculated for every grid. Sankaranarayanan et al. [35] proposed an auto-encoder network that takes as input both source and target images and regenerates them before they are fed to the segmentation network. Tsai et al. [36] proposed CGAN to add random noise to the source data before being fed to the segmentation network. They found that this approach improves the performance of the model on target domains. Huang et al. [37] separately trained two models for the source and the target domains. Because the target domain is without labels, the target model is trained by regressing it to the weights of the source model. Further, an adversarial loss is calculated in every layer of the two networks. Zhang et al. [38] used an adversarial loss between the source and the target data on both the first layer and the layers of the network. This method improves the adaptation performance of the network.

These are the main works that treated domain adaptation on semantic segmentation. We can deduce that, to our knowledge, no one has treated domain adaptation on semantic segmentation on aerial imagery. Most of the methods treated images of urban scenes taken from a camera mounted on a car. Aerial imagery has many dissimilarities with the data treated in these works. This is why we targeted this problem in this paper. We used it to test the efficiency of our method on the International Society for Photogrammetry and Remote Sensing (ISPRS) semantic segmentation dataset. We studied the domain adaptation from the Potsdam domain dataset to Vaihingen dataset [9].

3. Generative Adversarial Networks (GANs)

3.1. Generator and Discriminator

GANs are increasingly becoming popular due to the wide area of applications that they address. They were firstly introduced in 2014 by Goodfellow et al. [20]. They are composed of two models, named the generator and discriminator. The generator model is trained to generate data that are similar to the real data considered. The discriminator is trained to differentiate between the real and fake data generated by the generator. During the training, the generator and the discriminator are competing with each other, playing an adversarial zero-sum game. The loss on both models is balanced by the loss of its adverse model. In fact, the generator is trained to generate fake data that fool the discriminator, making it judge the generated fake data as real data. On the other side, the discriminator is trained to differentiate between the fake data and the real data. During the training, this game is solved using game theory theorems. At the end, the generator is well trained to generate data that are

similar to the real data and not previously seen in the training set. The discriminator is well trained to differentiate between the real and fake data. This simultaneous training of the discriminator and generator is shown in Figure 2.

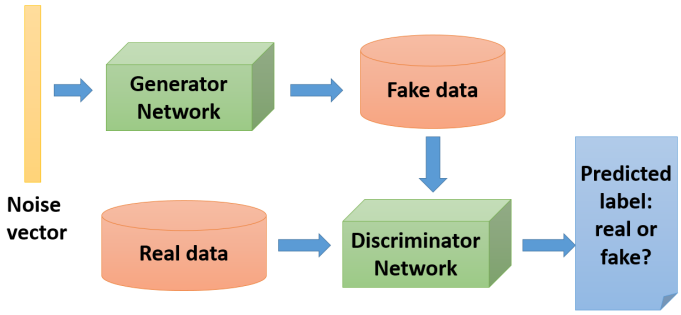

Figure 2. Generative adversarial network.

The two networks compete with each other during the training until reaching the Nash equilibrium. In game theory, Nash equilibrium is a strategy profile in which no player can unilaterally deviate and improve their payoff [39].

GAN's objective function is described by Equation (1):

$$min_G max_D V(D,G) = \mathbb{E}_{X \sim P_{data}(X)}[logD(X)] + \mathbb{E}_{z \sim P_z(z)}[log(1 - D(G(z)))], \quad (1)$$

where G is the cost function of the generator trained by maximizing $D(G(z))$. D is the cost function of the discriminator trained by minimizing $D(G(z))$. X is an image sampled from the real data distribution p_{data}, z is the noise vector sampled from the distribution p_z, $G(z)$ is the fake image generated by the generator. $\mathbb{E}_{X \sim P_{data}(X)}$ is the expectation over X drawn by the distribution described by $P_{data}(X)$. D and G are playing the two-player minimax game with value function $V(G,D)$ [20].

GANs have a plethora of implementations and applications [40]. The most attractive application for domain adaptation is image to image translation. In the next subsection, we focus more on this area and introduce the GAN models designed for this task.

3.2. GAN for Image to Image Translation

Image to image translation is the task of converting one image from a domain to another—for example, translating an image taken in the summer to another one that mimics its appearance as if it were taken in winter. This area may have numerous applications and use cases, and many GAN models were designed in the literature [41–44]. Image translation can be either paired [44] or unpaired [21].

3.2.1. Paired Image Translation

In paired image translation, the GAN model should be trained in a supervised way using labeled pairs from source domain to target domain. Considering that X is the source dataset, Y is the target dataset, and N is the number of samples in every dataset, the model will access every pair of corresponding images $\{x_i, y_i\}_{i=0..N}$ and try to learn how to convert between X and Y domains based on these samples. Pix2pix [44] is the major state-of-the-art architecture for paired image to image translation.

3.2.2. Unpaired Image Translation

In unpaired image translation, the GAN model is trained in a unsupervised way between two sets of images. The first set represents the source, while the second represents the target. Considering that X is the source dataset, Y is the target dataset, and N is the number of samples in every dataset, $\{x_i\}_{i=0..N}$ and $\{y_i\}_{i=0..N}$ are not necessarily corresponding and could be taken randomly from the

associated domain set. CycleGAN [21] is the major state-of-the-art architecture for unpaired image to image translation. It makes a bidirectional image to image translation between two sets of images.

4. Proposed Method

4.1. Our Proposed GAN Architecture

The proposed method aims to perform image level translation from the source domain to the target domain using a GAN as shown in Figure 3. We describe in this figure how we implemented an unpaired image to image translation GAN from the source domain to the target domain.

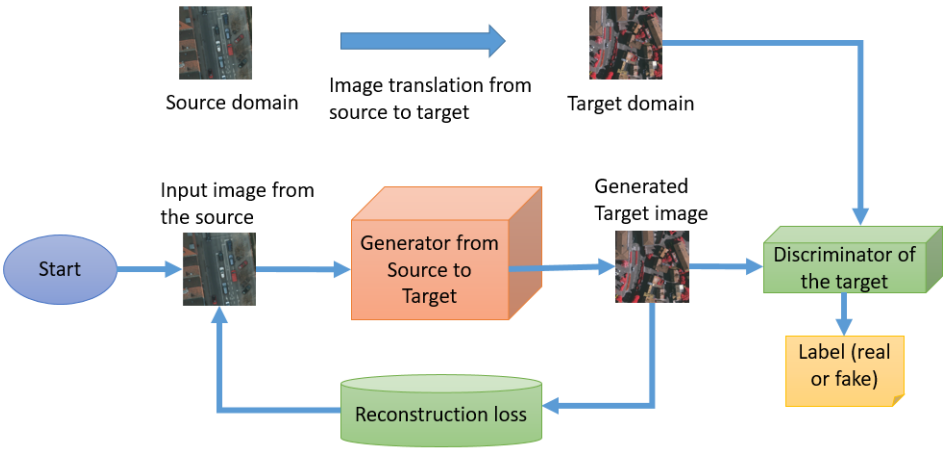

Figure 3. Generative adversarial network (GAN) architecture for unpaired image translation in aerial images.

This procedure was designed to make images of the source domain mimic the characteristics of the target domain (types of sensors, quality of the images, resolution, etc.). This will have the effect of reducing the domain shift related to the quality and characteristics of the images in the training set. To reduce our method cost, we did not adopt the traditional GAN approach. In fact, if we adopt it without modification, a paired dataset should be provided for every class of objects considered in our model. This will be really costly and time-demanding and does not harmonize with our goal to make the domain adaptation straightforward and easy to implement. Hence, we adopted a modified approach inspired from many state-of-the-art architectures [21,45]. We implemented an unpaired image translation adversarial network working in a unidirectional way from the source to the target as shown in Figure 3. The translation of an image from the source domain to the target domain does not need paired images. Images for both domains are collected separately without the need for corresponding pairs to train a mapping function $G : X \rightarrow Y$. This function $G(X)$ learns during the training process to make images from the source X imitate the distribution of images in the target Y, minimizing adversarial loss. However, we have here to take into consideration another condition. If we were only limited to this mapping function, the image translation would not be done as expected. In fact, because this mapping function is not constrained with paired data, the image translation is prone to being done in a meaningless way, leading to a model collapse. Therefore, we considered adding the inverse mapping function $F : Y \rightarrow X$ that makes the image translation on the inverse direction from the target to the source. This function $F(Y)$ learns during the training process to imitate the distribution of images in X, minimizing a second adversarial loss. Then, we added the reconstruction loss to consolidate that $F(G(X)) \approx X$ and $G(F(X)) \approx X$ simultaneously. Then, we trained our model jointly so that the image structure would be conserved during the translation process from the source domain to the target domain.

The architecture of the generator is similar to U-Net [4] architecture. We used an encoder–decoder network as illustrated in Figure 4. Four convolutional layers awee set for downsampling, and four

convolutional layers were used for upsampling. We used Leaky ReLU (rectified linear unit) [46] as the activation function for all the layers of downsampling and standard ReLU for all the layers of upsampling. Leaky ReLU is similar to the standard ReLU (rectified linear unit) but has a small slope α in the negative region. The Leaky ReLU function is defined as $f(x) = x$, if $x >= 0$; and as $f(x) = \alpha x$ if $x < 0$, where α is a very small coefficient. It allows having a small positive gradient when the function is not activated. The output features extracted from the encoder are passed into the decoder that will learn how to rebuild the original feature vector. We used dropout [47] in the decoder architecture to reduce overfitting. We used instance normalization [48] after every layer in the generator, because it was proven in [48] that it works better than batch normalization [49] for generator neural networks. It helps to provide better stylization in the image generation process. Figure 4 shows the architecture of the generator.

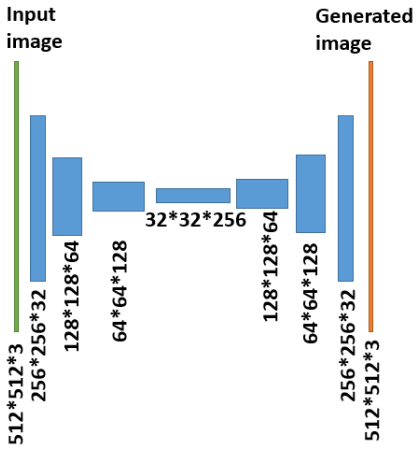

Figure 4. The encoder–decoder architecture of the generator.

Concerning the discriminator architecture, it receives as input the generated image and makes a binary classification output of real or fake image. We used five convolutional layers that encode the generated image into a feature vector of a size of 256. Then, we used an output neuron with Sigmoid activation function in the last layer to convert this feature vector into a binary output. In the same way as the generator, we used the Leaky ReLU [46] as an activation function for all the layers of the network, and we applied instance normalization [48] in every layer of the discriminator except the first and the last layer. We did not add normalization in these layers following the experimental settings given by Xiang et al. [50]. Figure 5 shows the architecture of the discriminator.

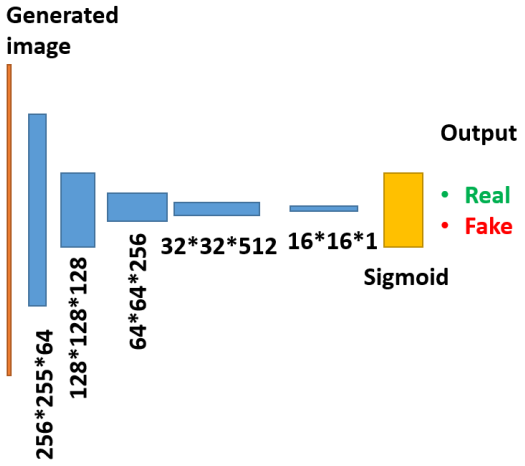

Figure 5. The architecture of the discriminator.

4.2. Algorithm Description

Based on the GAN architecture provided in Figure 3, we designed and implemented our proposed algorithm for domain adaptation in aerial imagery. The flowchart of the algorithm is described in Figure 6. The algorithm is divided into four steps. The first step is to train a segmentation model on the source dataset. In principle, with a good structured dataset, the segmentation accuracy could easily reach a level higher than 80%. The second step considers the training of our proposed GAN architecture to translate image efficiently from the source domain to the target domain. The third step is to convert the source dataset to the target domain using this GAN architecture. The output of the third step is a new dataset that conserves the structures represented in the images of the source dataset but mimics the global characteristics of the target dataset (imaging sensors, global coloring, etc.). The fourth step is to fine-tune the already trained segmentation model with the translated dataset associated with the source labels. This step helps the model parameters to learn the patterns of the target dataset and to converge to a better recognition of image structure on the target dataset. After the fine-tuning process, the semantic segmentation model is adapted to work on the target dataset.

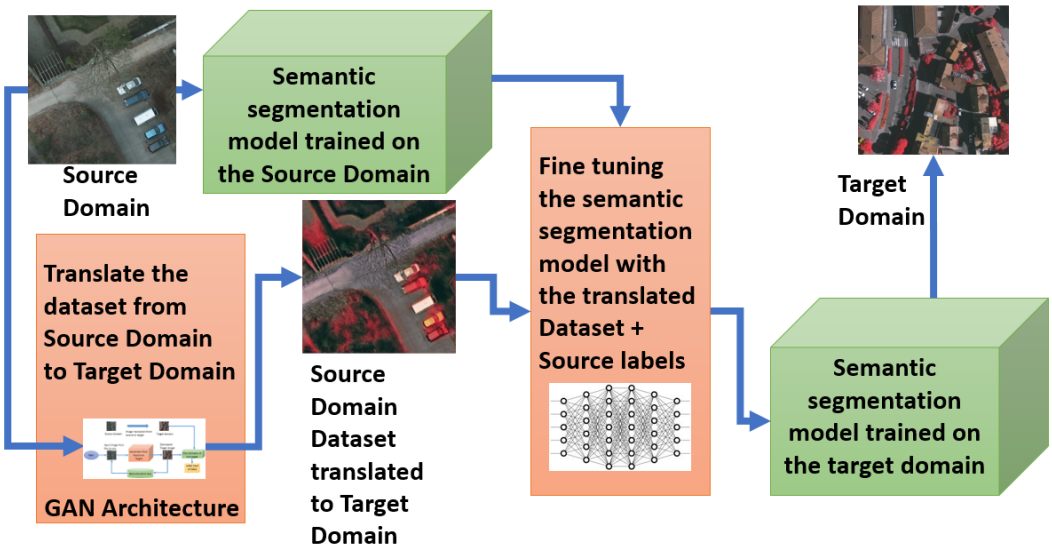

Figure 6. Flowchart of the domain adaptation algorithm.

4.3. Problem Formulation

In this section, we present the formal mathematical model of the algorithm. We considered the problem of domain adaptation from source domain data X_S, which are already mapped to their labels Y_S, to target domain data X_T without labels.

We started by training a source model M_S that performs the semantic segmentation on the source data by mapping the input images and their corresponding labels. The pixel-wise labels have one of C classes. Using the cross-entropy loss function, the source model M_S corresponds to:

$$L_{M_S}(M_S, X_S, Y_S) = -\mathbb{E}_{(x_s,y_s)\sim(X_S,Y_S)} \sum_{c=1}^{C} \mathbb{1}_{[c=y_s]} log(Softmax(M_S^{(c)}(x_s))). \quad (2)$$

$\mathbb{E}_{(x,ys)\sim(X_S,Y_S)}$ is the expectation over x_s, y_s drawn by the distribution described by X_S and Y_S. $\mathbb{1}_{[c=y_s]}$ is the corresponding loss for only the class c separately from other classes. Thanks to the advance in the semantic segmentation algorithms, M_S generally performs well on the source data. However, when applying the source model M_S on the target data, we have lower accuracy due to the domain shift that exists between the source and the target domain. To alleviate this domain shift, we began first by mapping the dataset images of the source domain to the target domain. This was implemented by our proposed GAN architecture that learns how to map the image samples between

domains so that the discriminator will be unable to detect that the mapped image from the source to the target does not really belong to the target. The next step was to fine-tune the source model M_S by running the trained model on the mapped dataset, and this helps to generalize our source model to perform better on the target domain, as proven in the experimental section of this paper.

The mapping model from source to target $G_{S \longrightarrow T}$ was implemented and trained to map from the source domain to the target domain. The goal was to generate image samples that would be classified by the adversarial discriminator D_T as real images from the target domain. On the other side, the adversarial discriminator D_T was trained to not be fooled by the generated images and to detect them successfully as fake. The loss function corresponding to this is:

$$L_{GAN}(G_{S \longrightarrow T}, D_T, X_T, X_S) = \mathbb{E}_{x_t \sim X_T}[log D_T(x_t)] + \mathbb{E}_{x_s \sim X_S}[log(1 - D_T(G_{S \longrightarrow T}(x_s)))]. \quad (3)$$

The training of this loss makes $G_{S \longrightarrow T}$ capable of generating from a sample image taken from the source domain an image that imitates the appearance of an image taken from the target domain. Therefore, from the source segmentation model M_S, we made a new model M_T that minimizes the loss function:

$$L(M_T, G_{S \longrightarrow T}(X_S), Y_S) = \\ - \mathbb{E}_{(G_{S \longrightarrow T}(x_s), y_s) \sim (G_{S \longrightarrow T}(X_S), Y_S)} \sum_{c=1}^{C} \mathbb{1}_{[c=y_s]} log(Softmax(M_T^{(c)}(G_{S \longrightarrow T}(x_s)))). \quad (4)$$

This loss function is trained in a similar manner to the loss defined in Equation (3). Therefore, the target model M_T is a copy from the already trained source model M_S that we trained on the mapped dataset by minimizing the loss defined in Equation (4). This operation makes the model generalize better on the target domain. The GAN loss defined in Equation 4 ensures that for a sample image x_s from the source domain, $G_{S \longrightarrow T}(x_s)$ will resemble the sample images taken from domain X_T. Although general resemblance can be assured through the training, we cannot guarantee that $G_{S \longrightarrow T}(x_s)$ maintains the structural content of x_s.

To preserve the content and the structure of x_s during the mapping operation assured by $G_{S \longrightarrow T}$, we used a GAN network working on the inverse direction from the target to the source as detailed in Section 3. It maps from the target to the source $G_{T \longrightarrow S}$. The loss to train for $G_{T \longrightarrow S}$ is identical to the loss defined for $G_{S \longrightarrow T}$ in Equation (3); just the parameters of the loss are changed to be:

$$L_{GAN}(G_{T \longrightarrow S}, D_S, X_S, X_T). \quad (5)$$

Then, we ensured that mapping a sample image x_s from the source to the target using $G_{S \longrightarrow T}$, followed by another mapping of this generated image $G_{S \longrightarrow T}(x_s)$ back to the source using the mapping function $G_{T \longrightarrow S}$, will generate an identical image of the source x_s. This is the reconstruction loss constraint that we added, as we explained in Section 3, to keep the structural content of the images during the mapping process. This loss constraint is formulated by the following equations:

$$G_{T \longrightarrow S}(G_{S \longrightarrow T}(x_s)) \approx x_s, \quad (6)$$

$$G_{S \longrightarrow T}(G_{T \longrightarrow S}(x_t)) \approx x_t. \quad (7)$$

To ensure that Equations (6) and (7) are satisfied, we imposed the reconstruction loss constraint defined in the following equation:

$$L_{reconstruction}(G_{S \longrightarrow T}, G_{T \longrightarrow S}, X_S, X_T) = \\ \mathbb{E}_{x_s \sim X_S}[\|G_{T \longrightarrow S}(G_{S \longrightarrow T}(x_s)) - x_s\|_1] + \mathbb{E}_{x_t \sim X_T}[\|G_{S \longrightarrow T}(G_{T \longrightarrow S}(x_t)) - x_t\|_1]. \quad (8)$$

After finishing the training of our proposed GAN architecture, we used it to translate the source data X_S to X_{S_tr}.Then, we profited from the labels provided with the source data by reusing them exactly the same in the training with the new translated dataset. We took the segmentation model M_S which is already trained on source data before translation, we fixed the weight values and used it as a start point for the training of our target model M_T. This model performs the semantic segmentation on the translated image data by mapping X_{S_tr} with their corresponding pixel-wise labels Y_S. Using cross-entropy as loss function, the target model corresponds to:

$$L_{M_T}(M_T, X_{S\_tr}, Y_S) = -\mathbb{E}_{(x_{s\_tr}, y_s)\sim(X_{S\_tr}, Y_S)} \sum_{c=1}^{C} \mathbb{1}_{[c=y_s]} log(Softmax(M_T^{(c)}(x_{s\_tr}))). \quad (9)$$

Finally, we obtained a target model M_T that is more adapted to work on the target domain, as described in the Experimental Section.

5. Experimental Results

In this section, our objective is to prove the efficiency of the proposed algorithm by providing the description of the implemented experiments and discussing the obtained results.

5.1. Datasets and Evaluation Metrics

5.1.1. Datasets

To validate our methodology, we used the ISPRS (WGII/4) 2D semantic segmentation benchmark dataset [9]. It is afforded by the ISPRS 2D semantic labeling challenge that currently provides the best platform to evaluate semantic segmentation algorithms for aerial images. We used the Vaihingen and Potsdam datasets, which are publicly available to the community. Although digital surface model (DSM) data are provided for every image, we only used the image data as we were targeting domain adaptation using only image data. Both datasets contain very-high resolution images with a resolution of 9 cm for Vaihingen images and 5 cm for Potsdam images. Note that the resolutions are different in both datasets, and this represents one of the factors that require domain adaptation. These resolutions are categorized in aerial imagery as very high resolution (VHR) and are helpful in recognizing objects clearly. In addition, this helps to maximize the intraclass variance and minimize the interclass variance by providing more details about objects. All images in both datasets are provided with their semantic segmentation labels, which comprise six classes of ground objects: building, tree, car, impervious surfaces, low vegetation, and clutter/background. Impervious surfaces indicate a paved area with no building on it. The clutter/background category refers to all the ground objects that are not included in the other five categories. The Vaihingen dataset includes 33 TOP images with sizes near to 2000 $\times$ 2000 pixels. All these 33 TOP images are released with the ground truth. The TOP file contains three channels: Infrared, red, and green bands. Among the 33 TOP images, 27 TOP images were used for training, and 6 images were used for the test. The Potsdam dataset is a larger dataset that contains 38 TOP images with a fixed size of 6000 $\times$ 6000 pixels. All these images are released with their ground truth. The TOP files for Potsdam contain 3 different spectral channels: red, green, and nlue. Among the 38 TOP images, 32 images were used for the training, and 6 images were used for the test. To train the segmentation model, we divided the images and their labels into squares of a size of 512 $\times$ 512 and fed the network with uniform patches of a size of 512 $\times$ 512. Figure 7 shows samples from Potsdam and Vaihingen ISPRS datasets.

Figure 7. Samples of images from Potsdam and Vaihingen International Society for Photogrammetry and Remote Sensing (ISPRS) datasets.

The distribution of pixels over the six classes is not proportionally balanced. Categories like impervious surface or buildings are much more represented as compared to other classes, like cars or clutter. Table 1 represents the percentage of each class proportionally to the total number of pixels. The percentage of a class is calculated by summing the number of pixels belonging to this class divided by the total number of pixels in the dataset.

Table 1. Percentage of each category in the datasets.

Category	Potsdam	Vaihingen
Impervious Surfaces	29.9%	29.3%
Buildings	28.2%	26.9%
Low vegetation	20.9%	19.4%
Trees	14.4%	22.4%
Cars	1.7%	1.3%
Clutter	4.8%	0.7%

5.1.2. Domain Shift Analysis

The domain shift from the source domain (Potsdam) to the target domain (Vaihingen) resulted from 3 essential factors. The first factor is the imaging sensor factor. Images of Vaihingen are captured using a 3-band sensor, IRRG (infrared, red, green). The images of Potsdam are also captured using a

3-band sensor, RGB (red, green, blue). For example, the class vegetation and trees are characterized by the green color due to the RGB sensor used for the Potsdam dataset. The segmentation model will be trained to recognize the varieties of green color that help to identify these classes accurately. In the Postdam dataset, the green color is well represented. In the Vaihingen dataset, it is totally transformed to a red color due to the change of the sensor. This change will affect the accuracy of the segmentation model and lead to a significant domain shift. The second factor is the resolution factor. Images of Vaihingen are captured using a resolution of 9 cm per pixel. Images of Potsdam are captured using a resolution of 5 cm per pixel. Going from one resolution to another could affect the ability of the segmentation model to accurately identify the classes and therefore generate a domain shift. The third factor of domain shift is the structural representation of the classes. Many classes show a difference of representation passing from the Potsdam dataset to Vaihingen dataset. For example, buildings in Postdam and Vaihingen are very comparable as they correspond to the building style of modern German towns. There is not much difference in the representation of the class building when going from Potsdam to Vaihingen. However, for other classes like ;ow vegetation and trees, there is a clear difference. In fact, Vaihingen contains agricultural areas, while Potsdam does not contain this kind of areas. Types of trees and vegetations differ when switching between the two datasets. The difference is clearer in the low vegetation class than in the trees class. In fact, there are similarities between most tree types of Vaihingen and Potsdam.

The domain shift between Potsdam and Vaihingen is generated from a combination of the three factors (imaging sensors, resolution, class representation). This allows us to study the effect of our proposed algorithm on reducing the domain shift related to every factor. Table 2 summarizes the effect of these factors on the domain shift of every class. The estimation of the factor impact is made after a careful analysis of every class on both domains. We can note that the effect of the resolution on the domain shift is low on all classes. In fact, passing from 5 cm per pixel to 9 cm per pixel does not affect the accuracy of the segmentation model very much. The feature extraction layers of the model have the ability to manage this scale of resizing. We note that the class building is mostly affected by the sensor factor; thus, it will be a study case for the effect of our algorithm on reducing domain shift made by the sensor factor only. The trees class will similarly be a study case, as it is mostly affected by the sensor factory and moderately affected by the class representation factor. The impervious surfaces and cars classes are not really affected by the three factors, so they will be a study case for the effect of our algorithm on classes that are not subjected to a domain shift when passing from one dataset to another. Classes low vegetation and clutter are highly affected by the sensor factor and the class representation factor. They will be a study case to study the effect of our algorithm on reducing the domain shift related to these factors combined.

Table 2. Effect of the domain shift factors on every class when passing from the Potsdam dataset to the Vaihingen dataset.

Factor of Domain Shift	Resolution	Sensor	Class Representation
Impervious Surfaces	low	low	low
Buildings	low	high	low
Low vegetation	low	high	high
Trees	low	high	medium
Cars	low	low	low
Clutter	low	high	high

5.1.3. Evaluation Metrics

To measure the efficiency of the semantic segmentation algorithms, we used four measures: the accuracy, the precision, the recall, and the F1 score. They are expressed using TP (true positives), TN (true negatives), FP (false positives), and FN (false negatives). If we consider a class C, TP corresponds to the number of pixels classified as C. TN is the number of pixels that do not belong to the class C, and the segmentation model did not classified them as C. FP is the number of pixels that are

classified falsely as C while they belong to other classes. FN is the number of pixels that belong to the class C, but the segmentation model associated them falsely to other classes. These measures are defined below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{10}$$

$$Precision = \frac{TP}{TP + FP}, \tag{11}$$

$$Recall = Sensitivity = \frac{TP}{TP + FN}, \tag{12}$$

$$F1\ Score = 2 * \frac{Pecision * Recall}{(Precision + Recall)}. \tag{13}$$

Moreover, we also used the intersection over union (IoU) to measure the efficiency of the segmentation. Since we have six different classes, IoU is calculated for every class separately. Then, the mean IoU of all classes is caculated. Equation (14) represents how to calculate the IoU for two different data samples, A and B.

$$IoU(A, B) = \frac{size(A \cap B)}{size(A \cup B)} \tag{14}$$

5.2. Experimental Settings

5.2.1. Step 1: Training the Segmentation Model

We first started with training a segmentation model on the source dataset. We chose Potsdam as the source dataset because it is far greater than the Vaihingen dataset. In fact, in real scenarios, target datasets are smaller and less structured than the source datasets. Then, we performed the segmentation using a state-of-the-art segmentation model, which is BiSeNet (bilateral segmentation network) [51]. It is currently the fastest segmentation model tested on the Cityscapes dataset [10] without affecting the accuracy. It achieves a 74.7% mean IoU on the CityScape dataset, with a speed of 65.5 frames per second [52]. The state-of-the-art on the CityScape dataset is PSPNet [5] that achieves a mean IoU of 81.2% but at a very low speed: 0.78 frames per second [52]. The factor of speed is significantly important in aerial image processing, as we need to process the video streams captured from aerial vehicles in real time. Figure 8 represents the architecture of BiSeNet.

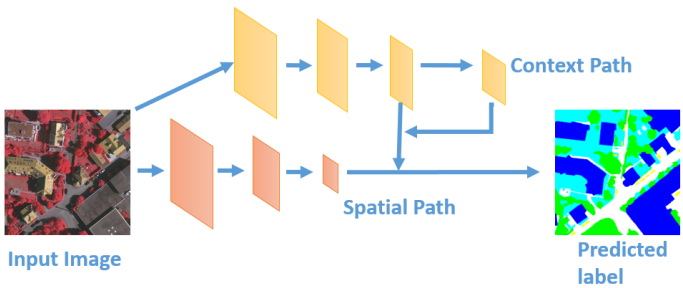

Figure 8. Architecture of the bilateral segmentation network (BiSeNet).

The experiments related to this research work were conducted on a GPU machine with the following characteristics:

- CPU: Intel Core i9-8950HK (six cores, Coffee Lake architecture);
- Graphic card: Nvidia GTX 1080, 8GB GDDR5;
- RAM: 32 GB RAM;
- Operating system: Linux (Ubuntu 16.04).

To train BiseNet on Potsdam, we used the Semantic Segmentation Suite [53], which is an open source framework that provides the implementation of many segmentation models in Tensorflow [54]. We used as the feature extractor for BiSeNet a state-of-the-art network, which is ResNet101 [55]. We ran the training for the Postdam dataset for 80 epochs, and the batch size was 1 image per batch. We did not use image augmentation techniques. As an optimizer for the training, we used ADAM optimizer [56], with the learning rate set to 0.0001. The training converges fast in less than 15 epochs, and the average segmentation accuracy exceeds 86%. Figure 9 shows the evolution of the training loss of BiSeNet on the Potsdam dataset over epochs.

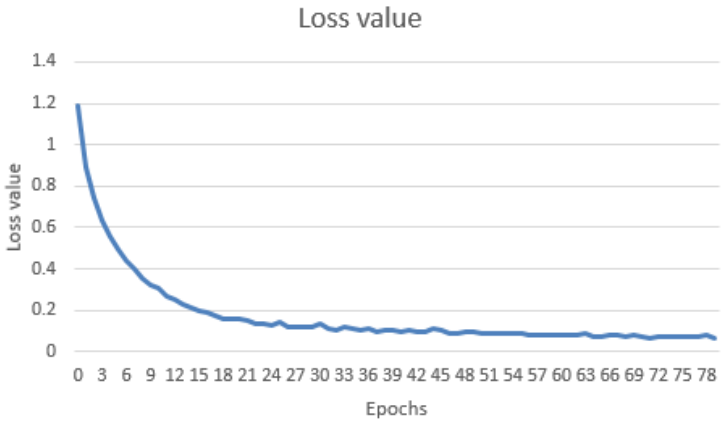

Figure 9. Loss of training BiseNet on the Potsdam dataset.

Figure 10 shows the evolution of the segmentation accuracy of BiseNet on the Potsdam validation dataset over epochs. We can see that segmentation accuracy exceeds rapidly 86% in a few epochs.

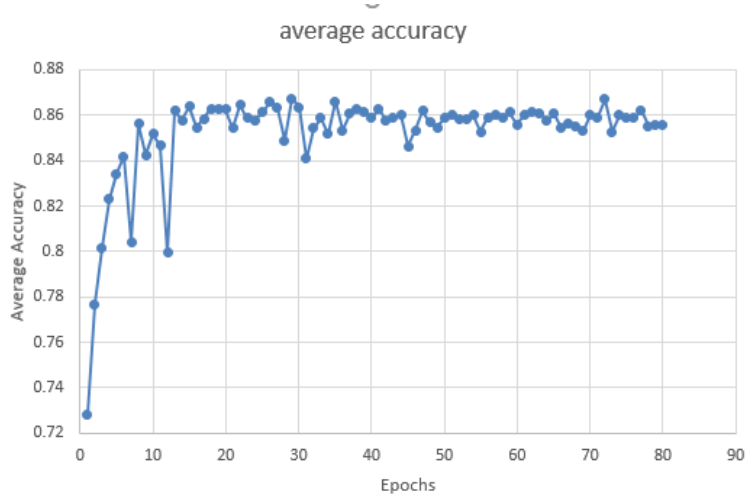

Figure 10. Evolution of average accuracy of BiseNet trained on Potsdam.

Figure 11 shows the evolution of the segmentation accuracy of every class on Potsdam validation dataset over epochs.

After finishing the training, we saved the weights of the BiSeNet model to be used later in Step 4 of the algorithm.

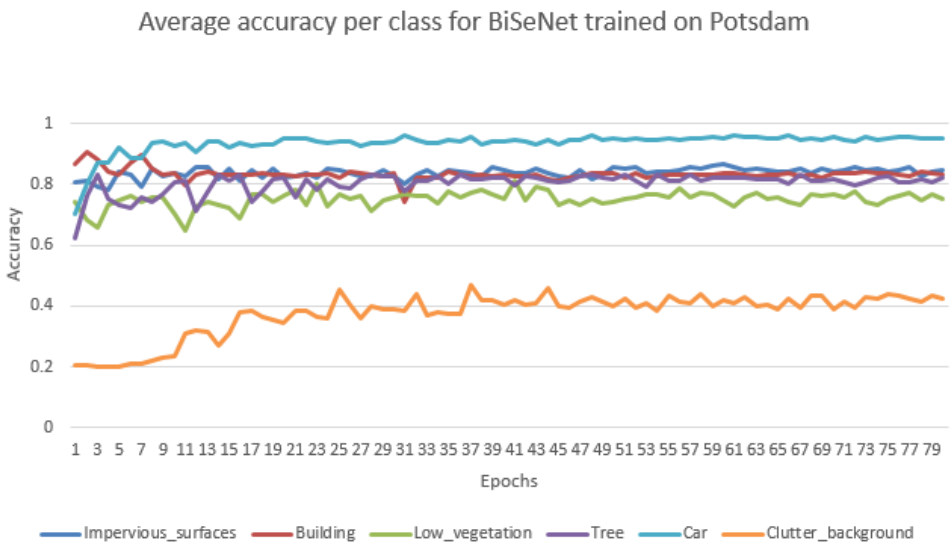

Figure 11. Evolution of per class accuracy of BiseNet trained on Potsdam.

5.2.2. Step 2: Training Our Proposed GAN Architecture

To train our proposed GAN architecture described in Section 3, we constructed two datasets: one for Potsdam and the other for Vaihingen. For each dataset, we collected randomly 400 images of a size of 512×512 from the original TOP images and divided these images into a training subset of 300 images and a test subset of 100 images. The proposed GAN architecture trains to translate images from the Potsdam domain (source domain) to the Vaihingen domain (target domain). The GAN architecture was implemented using Keras [57], which is a high-level deep learning framework developed in Python. We used Tensorflow [54] as a backend for the training. We set the slope α for Leaky ReLU as 0.2. We used as an optimizer for the training the ADAM optimizer [56], with the learning rate set to 0.0002. We trained the model until we got the discriminator accuracy superior to 92% and the generator loss inferior to 3. The convergence of the discriminator and the generator just needed a few epochs of joint training.

5.2.3. Step 3: Translating the Source Dataset to the Target Domain

Once the training of the proposed GAN architecture was done, we used it to translate the full dataset of the source domain (Potsdam) to the target domain (Vaihingen). Figure 12 shows samples of the Potsdam dataset translated to the Vaihingen domain. We note that the global style of the translated image is imitating the style of the target domain. The images generated are similar to what we can get as new images of the Potsdam town using the IRRG sensor used for Vaihingen images. We kept this translated dataset to be used in the fourth step of our algorithm.

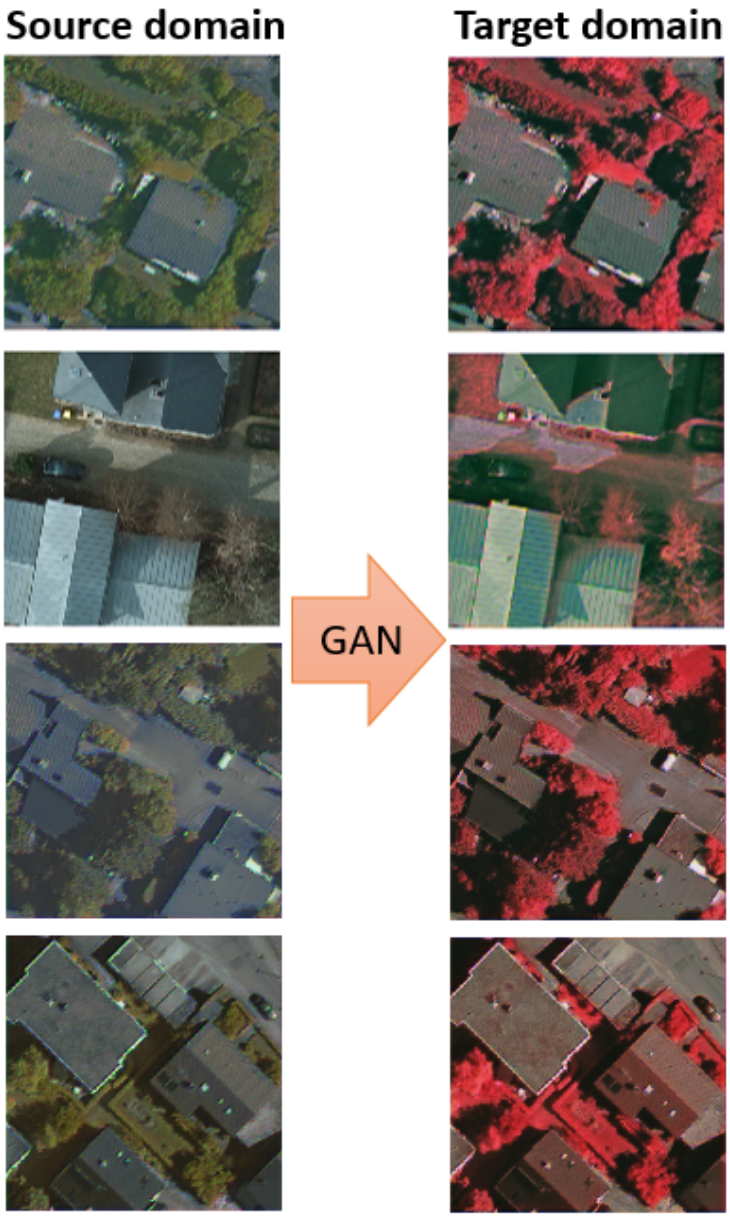

Figure 12. Mapping images from the source domain to the target domain using our proposed GAN.

5.2.4. Step 4: Fine-Tuning the Segmentation Model with the Translated Dataset

Once the translated dataset was ready, we used it to fine-tune the trained model prepared at Step 1. We did the fine-tuning process epoch by epoch, and we tested the model on the target dataset after every epoch to measure the improvement of average accuracy on the target dataset. We noted an increase in average accuracy between 5%and 17%. The average accuracy value was improved from 34% to values between 39% and 52%. We got an increase of 17% after 8 epochs only. In Figure 13, we show the improvement in average accuracy on the target dataset (Vaihingen) after every epoch of the fine-tuning process.

In Figure 14, we show the improvement of per class accuracy on the target dataset (Vaihingen) after every epoch of the fine-tuning process. We can see in the figure that the accuracy of two classes (trees and building) increased highly over epochs, although the other remained practically the same. In Section 6, we discuss the obtained results and we demonstrate the utility of our proposed approach in the domain adaptation of aerial imagery.

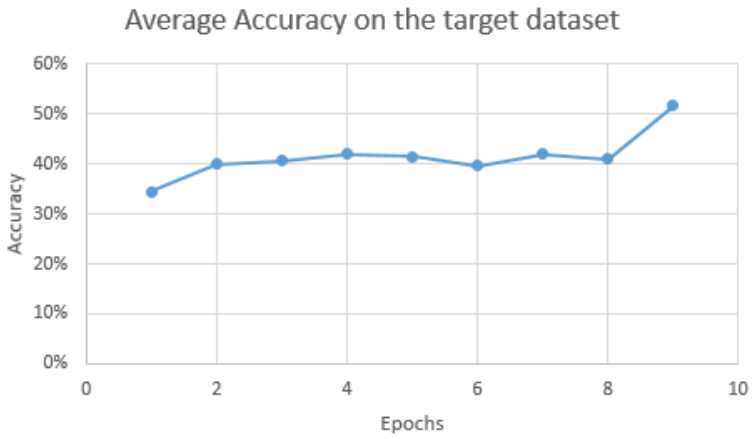

Figure 13. Improvement of average accuracy on the target dataset after each epoch.

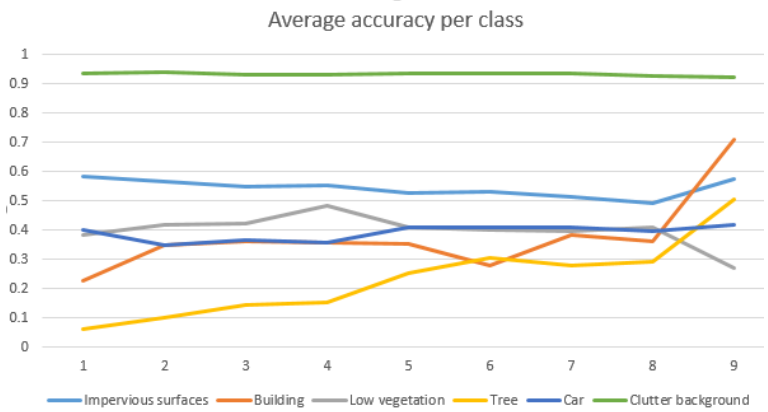

Figure 14. Improvement of accuracy per class on the target dataset after each epoch.

6. Discussion

The implementation of our algorithm increased the average accuracy of the segmentation model on the target dataset by a significant margin that reached 17%. Further, as presented in Table 3, similar improvements were also seen in the precision, recall, F1, and IoU (intersection over union) measures. These improvements made a visible amelioration on the predicted segmentation mask, as presented in Figure 15.

Going deeper, we made a study of the effect of our algorithm on every class apart. As described in Table 4, we have two types of effects. First, we have classes where our algorithms increased model accuracy by a high margin (classes building and tree). Comparing these results with Table 2, we note that these classes are characterized by a domain shift related highly to the sensor factor. If the domain shift is related only to the sensor factor, our algorithm is very efficient in increasing the accuracy of the model. For example, the class building, as explained in IV-A-2, is only affected by the sensor factor. We can see its average accuracy increasing from 0.23 to 0.71. If the domain shift is related mostly to the sensor factor, like class tree, our algorithm will be very efficient in increasing the accuracy but with some limitations due to the other domain shift factors. Concerning other classes (impervious surfaces, car, clutter background, and low vegetation), we can note that our algorithm has no practical effect in increasing or decreasing the accuracy. Accuracy will be conserved by our algorithm. These classes are, as described in Table 2, either not affected by any domain shift factor (like classes cars and impervious surfaces) or highly affected by a factor other than the sensor factor (like clutter Background or low vegetation).

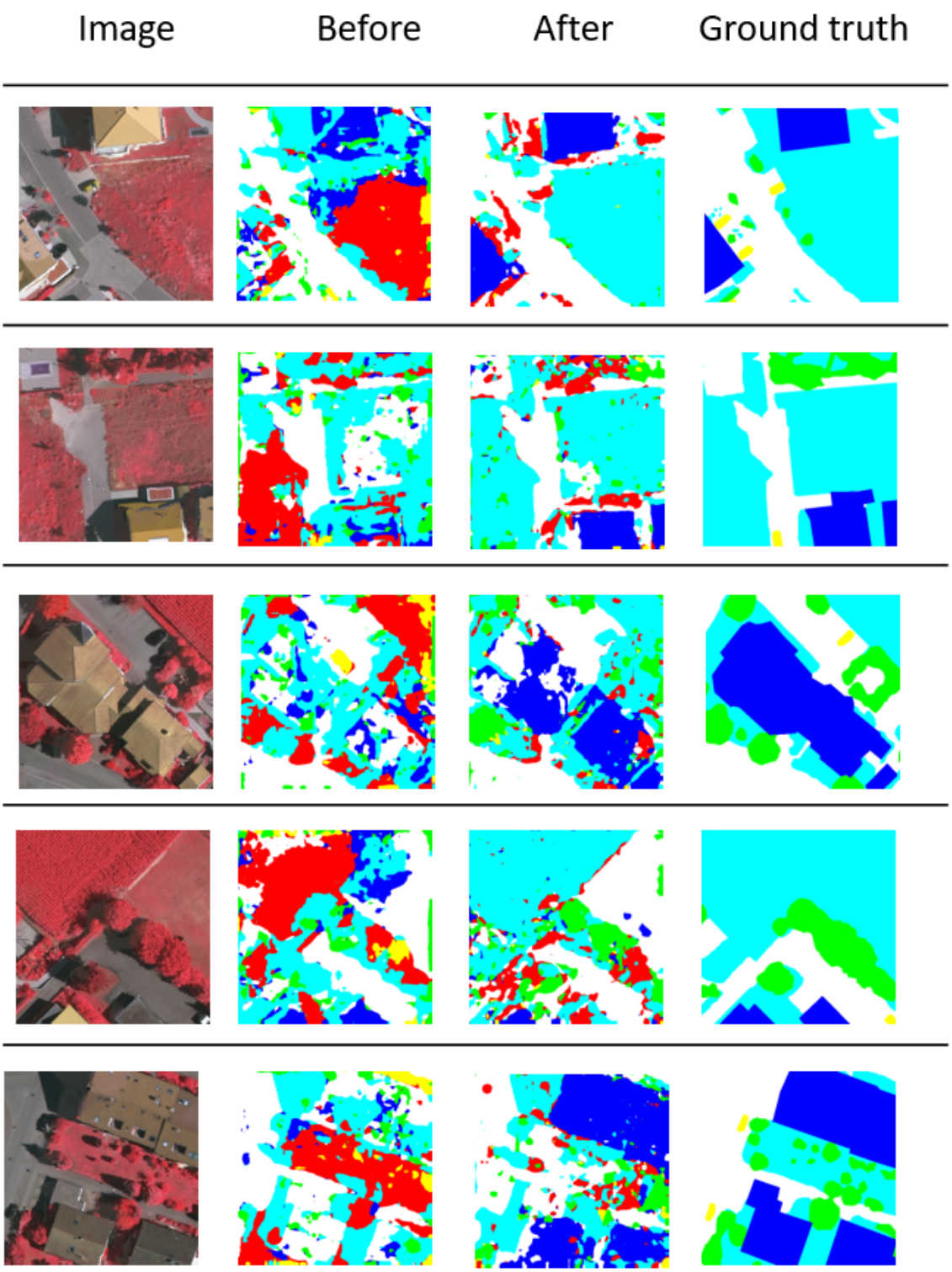

Figure 15. Samples of segmentation before and after implementation of our algorithm.

Table 3. Segmentation metrics on the target dataset before and after the implementation of our algorithm.

	Before	After
Average accuracy	0.35	0.52
Precision	0.35	0.54
Recall	0.35	0.52
F1 measure	0.32	0.49
IoU score	0.17	0.30

Table 4. Accuracy of the segmentation on every class before and after implementation of our algorithm.

	Before	After
Building	0.23	0.71
Tree	0.06	0.51
Impervious surfaces	0.58	0.57
Car	0.40	0.42
Clutter background	0.94	0.93
Low vegetation	0.38	0.27

We can estimate that our algorithm conserves the accuracy of the model if there is no domain shift or if the domain shift is related highly to a factor other than the sensor factor. This is a highly appreciated feature, as it allows combining it with other techniques that may reduce other domain shift factors. Our algorithm targets successfully the elimination of the sensor factor without affecting other factors. If the domain shift between the source dataset and the target dataset is only related to it, our algorithm is capable of improving the accuracy to a level similar to training the model on a full labeled dataset of the target, as seen in the class building. This fact is very helpful for aerial imagery processing, as it will relieve us from making new labeling dataset. Table 5 resumes the efficiency of our algorithm per case.

Table 5. Efficiency of our algorithm per case.

Domain Shift Factor	Efficiency	Examples of Classes
Sensor	High	Building, Tree
Other factors	Conserves efficiency	Low vegetation
No Domain shift	Conserves efficiency	Cars

Concerning the execution time of our algorithm, we needed around 7 to 10 h to train the system. Then, we needed 15 milliseconds to segment an image from the target domain with size 2048×1024. The training of the system can be automated to work with any targeted dataset and does not need labelling data, as it works in an unsupervised way. Step 1 of the algorithm that makes the training of the segmentation model on the source dataset needs around 3 to 4 h using the PC configuration detailed in the experimental settings part. Then, the training of the GAN architecture needs 2 to 3 h. The training works in an unpredictable way but generally converges to an acceptable stage within this period of time. The translation of the dataset needs only a few minutes. The last step of fine-tuning the model with the translated dataset needs 2 to 3 h. This range of timing makes our algorithm practical for adoption in real case scenarios of aerial imagery analysis, especially where the domain shift results mostly from sensor variation.

7. Conclusions

In this work, we have proposed a new method for domain adaptation in semantic segmentation of aerial imagery based on GANs. This method was confirmed to be efficient in targeting domain shift that results from sensor variation between the source and the target. In this case, this method is capable of substantially improving the accuracy of the segmentation model. In addition, it does

not affect the ability of the segmentation model to classify classes that do not have domain shift or classes that are subject to other domain shift factors, like variation of resolution or variation of class representation. Moreover, it has a very minimal cost, as it does not need labeling data or other manual work. The whole process can be trained for any new dataset and finished within 7 to 10 h. Our work is showing the promising potential of GANs in aerial image analysis and it is, to our knowledge, the first to treat the problem of domain adaptation in semantic segmentation of aerial imagery using GANs. Nevertheless, our algorithm should be coupled with a supervised approach to cope with domain shift factors other than sensor variation. In fact, these factors could only be mitigated in a supervised way to guide the image generation process during the translation between domains.

Author Contributions: B.B. and Y.B. designed the method. B.B. implemented the method and wrote the paper. Y.B., A.K., and K.O. contributed to the supervision of the work, analysis of the method, and paper writing.

Funding: This research was funded by Prince Sultan University.

Acknowledgments: This work is supported by the Robotics and Internet of Things Lab (RIOTU), Prince Sultan University under the research grant entitled: Smart Vehicle Surveillance System using Deep Learning on Aerial Images.

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

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
2. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, Boston, MA, USA, 7–12 Jun 2015; pp. 3431–3440.
3. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef] [PubMed]
4. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; Springer: Cham, Switzerland, 2015; pp. 234–241.
5. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid Scene Parsing Network. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 22–25 July 2017. [CrossRef]
6. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 834–848. [CrossRef] [PubMed]
7. Lateef, F.; Ruichek, Y. Survey on semantic segmentation using deep learning techniques. *Neurocomputing* **2019**, *338*, 321–348 [CrossRef]
8. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *Lecture Notes in Computer Science*; Springer International Publishing, 2018; pp. 833–851. [CrossRef]
9. Gerke, M. *Use of the Stair Vision Library within the ISPRS 2D Semantic Labeling Benchmark (Vaihingen)*; ResearcheGate: Berlin, Germany, 2014.
10. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 3213–3223.
11. Richter, S.R.; Vineet, V.; Roth, S.; Koltun, V. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 102–118.
12. Ros, G.; Sellart, L.; Materzynska, J.; Vazquez, D.; Lopez, A.M. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 3234–3243.

13. Saleh, F.; Aliakbarian, M.S.; Salzmann, M.; Petersson, L.; Gould, S.; Alvarez, J.M. Built-in foreground/background prior for weakly-supervised semantic segmentation. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 413–432.
14. Bearman, A.; Russakovsky, O.; Ferrari, V.; Fei-Fei, L. What's the point: Semantic segmentation with point supervision. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 549–565.
15. Shimoda, W.; Yanai, K. Distinct class-specific saliency maps for weakly supervised semantic segmentation. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 218–234.
16. Tzeng, E.; Hoffman, J.; Darrell, T.; Saenko, K. Simultaneous deep transfer across domains and tasks. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 4068–4076.
17. Long, M.; Cao, Y.; Wang, J.; Jordan, M.I. Learning transferable features with deep adaptation networks. *arXiv* **2015**, arXiv:1502.02791.
18. Tzeng, E.; Hoffman, J.; Saenko, K.; Darrell, T. Adversarial discriminative domain adaptation. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; Volume 1, p. 4.
19. Luo, Z.; Zou, Y.; Hoffman, J.; Fei-Fei, L.F. Label efficient learning of transferable representations acrosss domains and tasks. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 165–177.
20. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
21. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; doi:10.1109/iccv.2017.244.
22. Patel, V.M.; Gopalan, R.; Li, R.; Chellappa, R. Visual Domain Adaptation: A survey of recent advances. *IEEE Signal Process. Mag.* **2015**, *32*, 53–69. [CrossRef]
23. Saenko, K.; Kulis, B.; Fritz, M.; Darrell, T. Adapting visual category models to new domains. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2010; pp. 213–226.
24. Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; Lempitsky, V. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.* **2016**, *17*, 2096–2030.
25. Ganin, Y.; Lempitsky, V. Unsupervised domain adaptation by backpropagation. *arXiv* **2014**, arXiv:1409.7495.
26. Bousmalis, K.; Silberman, N.; Dohan, D.; Erhan, D.; Krishnan, D. Unsupervised pixel-level domain adaptation with generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3722–3731.
27. Hoffman, J.; Tzeng, E.; Park, T.; Zhu, J.Y.; Isola, P.; Saenko, K.; Efros, A.A.; Darrell, T. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv* **2017**, arXiv:1711.03213.
28. Vazquez, D.; Lopez, A.M.; Marin, J.; Ponsa, D.; Geronimo, D. Virtual and real world adaptation for pedestrian detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 797–809. [CrossRef] [PubMed]
29. Peng, X.; Saenko, K. Synthetic to real adaptation with generative correlation alignment networks. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 12–15 March 2018; pp. 1982–1991.
30. Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; Webb, R. Learning from simulated and unsupervised images through adversarial training. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2107–2116.
31. Shafaei, A.; Little, J.J.; Schmidt, M. Play and learn: Using video games to train computer vision models. *arXiv* **2016**, arXiv:1608.01745.
32. Hoffman, J.; Wang, D.; Yu, F.; Darrell, T. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv* **2016**, arXiv:1612.02649.
33. Zhang, Y.; David, P.; Gong, B. Curriculum domain adaptation for semantic segmentation of urban scenes. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2020–2030.

34. Chen, Y.; Li, W.; Van Gool, L. Road: Reality oriented adaptation for semantic segmentation of urban scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7892–7901.
35. Sankaranarayanan, S.; Balaji, Y.; Jain, A.; Lim, S.N.; Chellappa, R. Unsupervised domain adaptation for semantic segmentation with gans. *arXiv* **2017**, *2*, arXiv:1711.06969.
36. Tsai, Y.H.; Hung, W.C.; Schulter, S.; Sohn, K.; Yang, M.H.; Chandraker, M. Learning to adapt structured output space for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7472–7481.
37. Huang, H.; Huang, Q.; Krahenbuhl, P. Domain transfer through deep activation matching. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 590–605.
38. Zhang, Y.; Qiu, Z.; Yao, T.; Liu, D.; Mei, T. Fully Convolutional Adaptation Networks for Semantic Segmentation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6810–6818.
39. Oliehoek, F.A.; Savani, R.; Gallego, J.; van der Pol, E.; Groß, R. Beyond Local Nash Equilibria for Adversarial Networks. *arXiv* **2018**, arXiv:1806.07268.
40. Goodfellow, I.J. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv* **2016**, arXiv:1701.00160.
41. Liu, M.Y.; Breuel, T.; Kautz, J. Unsupervised Image-to-Image Translation Networks. In Proceedings of the NIPS Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
42. Zhu, J.Y.; Zhang, R.; Pathak, D.; Darrell, T.; Efros, A.A.; Wang, O.; Shechtman, E. Toward Multimodal Image-to-Image Translation. In Proceedings of the NIPS Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
43. Yi, Z.; Zhang, H.; Tan, P.; Gong, M. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2868–2876.
44. Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-to-Image Translation with Conditional Adversarial Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017. [CrossRef]
45. Bashmal, L.; Bazi, Y.; AlHichri, H.; AlRahhal, M.M.; Ammour, N.; Alajlan, N. Siamese-GAN: Learning Invariant Representations for Aerial Vehicle Image Categorization. *Remote Sens.* **2018**, *10*, 351. [CrossRef]
46. Xu, B.; Wang, N.; Chen, T.; Li, M. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv* **2015**, arXiv:1505.00853.
47. Srivastava, N.; Hinton, G.E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
48. Ulyanov, D.; Vedaldi, A.; Lempitsky, V. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv* **2016**, arXiv:1607.08022.
49. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**, arXiv:1502.03167.
50. Xiang, S.; Li, H. On the effects of batch and weight normalization in generative adversarial networks. *arXiv* **2017**, arXiv:1704.03971.
51. Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; Sang, N. BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation. In *Lecture Notes in Computer Science*; Springer International Publishing: Cham, Switzerland, 2018; pp. 334–349. [CrossRef]
52. Real-Time Semantic Segmentation on Cityscapes. Available online: https://paperswithcode.com/sota/real-time-semantic-segmentation-cityscap (accessed on 28 March 2019).
53. Real-Time Semantic Segmentation on Cityscapes. Available online: https://github.com/GeorgeSeif/Semantic-Segmentation-Suite (accessed on 28 March 2019).
54. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. TensorFlow: A system for large-scale machine learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
55. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]

56. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
57. Chollet, F. Keras. 2015. Available online: https://github.com/fchollet/keras (accessed on 6 June 2019).