GENERATING SYNTHETIC TRAINING DATA FOR OBJECT DETECTION USING MULTI-TASK GENERATIVE ADVERSARIAL NETWORKS

Y. Lin *, K. Suzuki, H. Takeda, K. Nakamura

Dept. of R&D, KOKUSAI KOGYO CO., LTD., 2-24-1 Harumi-cho, Fuchu-shi, Tokyo, 183-0057, JAPAN
(utei_rin, kumiko_suzuki, hiroshi1_takeda, kazuhiro_nakamura)@kk-grp.jp

Commission II, WG II/5

KEY WORDS: Mobile Mapping System, Object Detection, Convolutional Neural Networks, Generative Adversarial Networks, Multi-Task Training, Synthetic to Real

ABSTRACT:

Nowadays, digitizing roadside objects, for instance traffic signs, is a necessary step for generating High Definition Maps (HD Map) which remains as an open challenge. Rapid development of deep learning technology using Convolutional Neural Networks (CNN) has achieved great success in computer vision field in recent years. However, performance of most deep learning algorithms highly depends on the quality of training data. Collecting the desired training dataset is a difficult task, especially for roadside objects due to their imbalanced numbers along roadside. Although, training the neural network using synthetic data have been proposed. The distribution gap between synthetic and real data still exists and could aggravate the performance. We propose to transfer the style between synthetic and real data using Multi-Task Generative Adversarial Networks (SYN-MTGAN) before training the neural network which conducts the detection of roadside objects. Experiments focusing on traffic signs show that our proposed method can reach mAP of 0.77 and is able to improve detection performance for objects whose training samples are difficult to collect.

1. INTRODUCTION

In recent years, images, including panoramic images, and point cloud collected by Mobile Mapping System (MMS) are used to generate HD maps, which can be applied to autonomous driving, smart city, etc. The HD Maps conclude specific information on roadside environments, including road objects (road lines, crosswalk, etc.) and roadside objects such as traffic signs. However, data creation of these objects still heavily depends on operators' manual work, which is costly in terms of both time and money.

Benefit from advances that deep learning technology has achieved in recent years, several have been proposed to extract objects of interest from images or point clouds using CNN based algorithms. (Wolf et al., 2019) proposed a method to detect manholes and road markings by semantic segmentation using images rendered from point clouds. A CNN algorithm proposed by (Mori et al., 2018) classifies the categories of pole-like objects. Meanwhile, CNN based algorithms are highly dependent on the quality of training dataset, whose creation is both time-consuming and costly. Besides, the number of roadside objects in the real condition is highly imbalanced. Take a traffic sign as an example. Figure 1 shows the sample numbers of each category of traffic signs which are collected from MMS in Japan(Lin et al., 2018). Approximately 70 categories of traffic signs can hardly collect enough samples to train a CNN model. On the other hand, the rest 20 categories of traffic signs contribute more than 90% samples of the dataset. This is known as a long-tail phenomenon which could cause a significant performance drop.

To tackle the aforementioned problem, we propose a method to generate synthetic training samples for training an object detector. The flowchart of our proposed method is shown in Figure 2. To be more specific, we propose an SYN-MTGAN architecture, which transfers the style between synthetic data and real data, to generate training samples that can improve the detection performance of objects whose real training samples are difficult to collect in the real scene. The proposed SYN-MTGAN generates training samples along with predicting the category of the generated samples. In this study, we focus on traffic signs to verify the effectiveness of our proposed method.

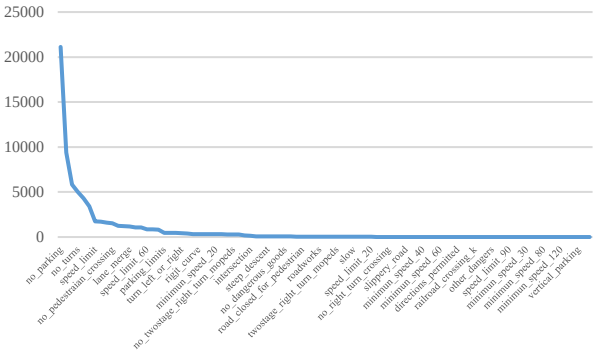


Figure 1. The number of Traffic Sign Samples in Real Scene

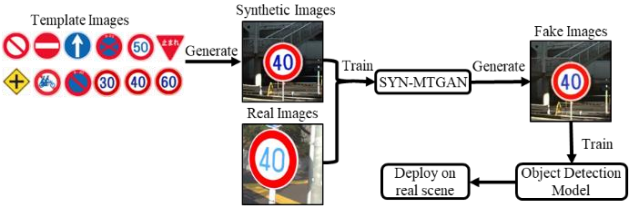


Figure 2. Flowchart of our proposed Method

* Corresponding author

2. RELATED WORK

2.1 Object Detection

In the Object Detection task, the category and position of a target object are predicted at the same time. R-CNN (Girshick et al., 2014) first adopted CNN to tackle this task by a two-stage strategy, which generates proposals of target region using Selective Search (Uijlings et al., 2013) and classifies the category of each region using AlexNet (Krizhevsky et al., 2012), a classic CNN architecture for image classification. R-CNN was improved in terms of performance and speed in the following years. Faster R-CNN (Ren et al., 2015) proposed a trainable neural network, Region Proposal Network (RPN), to generate proposals of a target region. This framework influences numerous methods that have been proposed in the following years. On the contrary, one-stage strategy methods focus on the real-time processes. SSD (Liu et al., 2016) and YOLO (Redmon et al., 2016) can be considered as representative ones. The key idea of a one-stage method is that predicting proposal regions and category of the target object at the same time using a single network, meanwhile a two-stage strategy predicts these two tasks in separate networks.

2.2 Synthetic to Real

Both one-stage and two-stage Object Detection algorithms suffer from imbalanced training samples during training. A common solution is to carry out the hard negative mining. Online Hard Example Mining (OHEM) (Shrivastava et al., 2016) proposed to use hard negative samples more frequently than normal ones during training according to a loss. Instead of collecting samples from real scenes, another solution is generating synthetic data from a simulator. (Hinterstoisser et al., 2017) trained an object detector model using synthetic data and pre-trained features of real data. (Abu Alhaija et al., 2018) augmented real scene images by synthetic images to train a network for driving scene recognition. (Cordts et al., 2016) trained Faster R-CNN using a large synthetic dataset and a small real scene dataset. (Wu et al., 2017) raised the proportion of real scene data by combining synthetic data created from the Grand Theft Auto game and KITTI (Geiger et al., 2012) data for training a neural network. Furthermore, to reduce the gap between synthetic and real scenes' distribution, (Shrivastava et al., 2017), (Bujwid et al., 2018), (Huang et al., 2018) and (Yu et al., 2019) developed GAN-based frameworks to generate more realistic synthetic images by projecting or mapping real feature distribution onto synthetic images. (Zheng et al., 2018) employed this idea on depth estimation and revealed pleasant results. Domain randomization or domain adaptation is further applied to minimize the gap. (Tremblay et al., 2018) used domain randomization to force the network to focus on learning semantic features other than features, i.e. color, brightness, and texture, which are usually determined by the dataset. Such strategies are proved to be effective to minimize the gap between synthetic data (source domain) and real scene data (target domain).
However, these proposed algorithms cannot solve the aforementioned problem when the target domain does not exist, which is caused by a lack of real scene samples. Creating synthetic data that is similar to real scene data is needed to solve the problem that we are facing, meanwhile, there is little research that focuses on transferring synthetic images to the real one for object detection tasks, especially multiple similar categories exist.

2.3 Generative Adversarial Networks

Generative Adversarial Networks (GAN) was proposed to generate images (Goodfellow et al., 2014) and has achieved significant results in many computer vision tasks. Commonly, GAN consists of two neural networks, the Generator and the Discriminator. The Generator generates a synthetic image from a random vector. On the other hand, Discriminator classifies if the input image is a synthetic image created by the Generator or a real one. The training of Generator and Discriminator is carried out simultaneously, which is called adversarial training. And eventually, the Generator is trained to generate synthetic images whose appearances are close to real images. The quality of images generated by GAN has been improved notably. (Karras et al., 2019) proposed a Generator with hierarchical architecture to recover details such as eyes and expressions of human faces. Besides using random vectors, GAN can also use images and natural languages as input to generate images. (Reed et al., 2016) used natural languages to generate a user-preferred image using GAN. GAN also had been proved to be effective to solve tricky Object Detection tasks, for instance small object detection. SOD-MTGAN (Bai et al., 2018) used a multi-task Discriminator to help Generator to generate better super-resolution images of a human head, which achieved higher performance on human head detection task, especially a small head. The experiments (Bai et al., 2018) show that GAN is useful to generate images that are able to improve the performance of Object Detection. Additionally, GAN can also transfer the style between two sets of images. CycleGAN with two Generators and Discriminators, proposed by (Zhu et al., 2017), learns the style and transfers it between two sets of images. The architecture that CycleGAN proposed enables the model to learn consistency loss between two Generators and results in generating more stable and better outputs. These previous studies of GAN show that images generated by GAN can be utilized as training samples to train an object detector.

3. METHODOLOGY

3.1 Overview

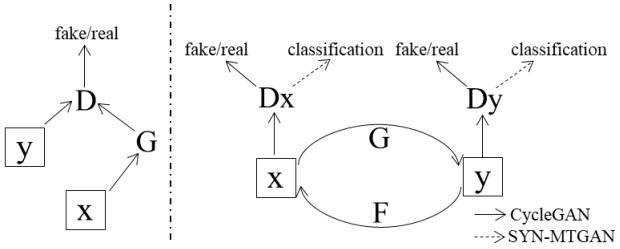
We propose an SYN-MTGAN architecture, inspired by SOD-MTGAN and CycleGAN, to generate synthetic data as training data for Object Detection while real scene samples are difficult to collect. Specifically, our method takes fake images and real images of target objects as input data for training the SYN-MTGAN. It learns the distributions of features, such as texture, illumination et al. that exist in the real scenes but difficult to reproduce by a simulator, from real images, and transfers them to fake images. There are two key points in our proposed method. First, the Discriminator (hereinafter, called D) of SYN-MTGAN not only classifies the input as fake or real but also predicts the category of the target object at the same time, which is i.e. a multi-task neural network. Second, we propose a new loss function in order to encourage SYN-MTGAN to only transfer the style of foreground, other than background.

We conducted experiments using the images that SYN-MTGAN generated to train an Object Detection model, comparing to the model that is trained by real scene data, to verify its effectiveness.

3.2 SYN-MTGAN Architecture

The architecture of a normal GAN is shown in Figure 3 (a). In normal GAN, fake images (hereinafter, called y') generated by Generator (hereinafter, called G) from input x, along with real images (hereinafter, called y), are fed into D for classification, which predicts whether the input is the generated y' or y (fake/real).

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2020-443-2020 | © Authors 2020. CC BY 4.0 License.



(a) Normal GAN (b) CylceGAN and Proposed SYN-MTGAN

Figure 3. Architecture of GAN

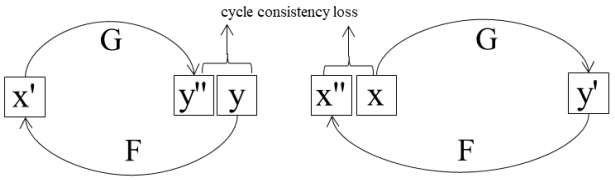


Figure 4. Flowchart of the Cycle Consistent Training

In this study, intending to improve the quality of generated images and stabilize the training process of GAN, we designed a GAN architecture, inspired by CycleGAN. The architecture of CycleGAN is shown in Figure 3 (b), which is an end-to-end framework for image style transfer. CycleGAN contains 4 networks, two Generators, and two Discriminators. The basic idea of CycleGAN is training the 4 networks to learn the style of image x and transfer this learned style to image y, which as a different style, using the cycle consistent loss. CycleGAN also learns the style of y and transfers it to x, which is proved effective for boosting performance. In the training process of CycleGAN, a Generator G is trained to generate a fake image y' by learning feature distribution (style) of real image y and transferring it to synthetic image x, meanwhile another Generator F is trained to generate x'' by mapping from real image feature distribution to fake ones. As a result, x'' is remapped to its original style and should be as similar to x as possible. This is called a forward cycle. This process can be considered as a recovery of y. y' and y are the input to Discriminator Dy for fake/real classification. The reverse cycle performs oppositely. Generator F is trained to generate another kind of fake images (hereinafter, called x') which learn the style of synthetic image x and transfer it to real image y, meanwhile x' is remapped as y'', similar to its original style, by G. x' and x perform as the input to Discriminator Dx for classifying fake/real. The cycle consistent loss calculates the divergence between x', x and y', y, and encourages the Generators to learn the mapping of two distributions respectively. The flowchart of cycle consistent training is shown in Figure 4. Besides, the purpose of G and F is to learn the distribution from two sets of images. Note that it is possible to train the CycleGAN-style architecture using unpaired images, which can reduce the cost of preparing synthetic and real images.

The D of our proposed SYN-MTGAN is a multi-task architecture. Besides fake/real classification as general GAN Discriminator does, it also predicts the category of target objects, in both Dx and Dy. The classification task predicts the category of a target object, similar to normal image classification.

3.3 Network Architecture

The details of the network architecture are shown in Figure 5. The G adopts Encoder-Decoder style, which contains two convolutions with a stride size of 2, nine Residual Blocks (He et al., 2016), and two transposed convolutions with a stride size of 1/2. Instance normalization (Ulyanov et al., 2016) is adopted in both G and D as it showed better performance in an image style transfer. The architecture of D is also based on CycleGAN. To be specific, four convolutions with a stride size of 2 and one convolution with stride 1 are implemented to extract feature map, whose size is 1/16 of the original input image. This feature map is shared with two parallel branches, discriminative recognition and classification. The classification branches are connected to the feature map by one convolution and two fully connected layers.

3.4 Loss Function

We adopted five loss functions to train the SYN-MTGAN. Adversarial loss and cycle consistency loss, the same as CycleGAN, are adopted to optimize the G and D. We adopted classification loss to optimize G and D. We propose a revised version of identity loss of CycleGAN to strength G to focus on the target area other than the background.

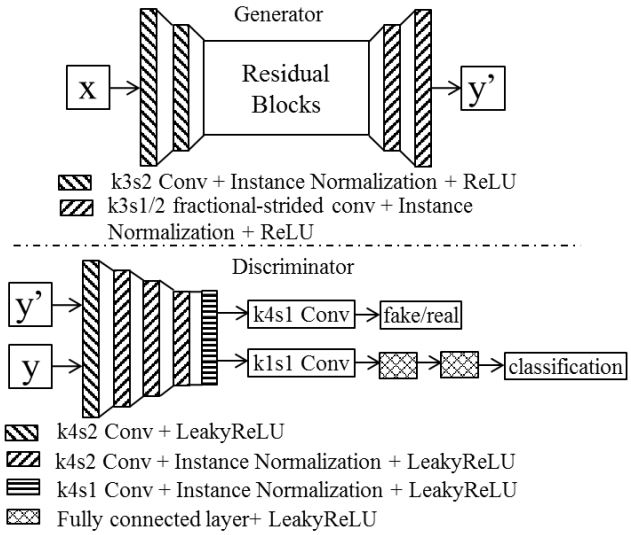


Figure 5. Details of Network Architecture. Conv denotes convolution. k means kernel size. s denotes stride size. For example, k3s2 denotes convolution with kernel size of 3 and stride size of 2.

3.4.1 Adversarial Loss: Adversarial loss (hereinafter, called L_{GAN}) encourages the G to generate fake images that are as close as possible to real ones that can fool D, and D not to be fooled by images that G generated. The equation is shown in Equation (1), which $D_y(G(x_i))$ denotes the probability of generated real image $G(x_i)$, or y', being a real image.

$$L_{GAN} = -\sum_{i=0}^{m} \log\left(D_y(y_j)\right) - \sum_{j=0}^{n} \log\left(1 - D_y(G(x_i))\right) \quad (1)$$

where x_i, y_j = training sample
n, m = numbers of training sample

3.4.2 Cycle Consistency Loss: The CycleGAN calculates the difference between x and x", y and y" as cycle consistency loss (hereinafter, called L_{cyc}) to encourage G and F to learn the perfect distribution of synthetic and real images, as Equation (2) shows. $\text{F}(\text{G}(x_i))$ denotes the images x_i'', generated by F from y_i', which are generated by G. Smaller loss indicts that G and F can recover x" or y" better. L_{cyc} uses L1 loss to calculates the difference.

$$L_{\text{cyc}} = \sum_{i=0}^{n} \left\| \text{F}(\text{G}(x_i)) - x_i \right\|_1 + \sum_{j=0}^{m} \left\| \text{G}\left(\text{F}(y_j)\right) - y_j \right\|_1 \quad (2)$$

3.4.3 Identity Loss: The objective of G and F is to learn the distribution of target features from two sets of images. However, the Generator of GAN tends to transfers the feature it learns and transfers the feature to the whole image, including the background. CycleGAN uses identity loss (hereinafter, call L_{iden}), which calculates the L1 loss between the input and output of the Generator, to encourage G and F to focus on learning the features of target object other than background. We propose a revised version of L_{iden}, shown in Equation (3). Proposed L_{iden} only focuses on the change of background by blocking the foreground target object using the bounding box information.

$$L_{iden} = \sum_{i=0}^{n} \| \text{F}(x_i^*) - x_i^* \|_1 + \sum_{j=0}^{m} \left\| \text{G}(y_j^*) - y_j^* \right\|_1 \quad (3)$$

where x_i^*, y_i^* = background area of an image

3.4.4 Classification Loss: The classification branch in the Discriminator is reported to encourage the Generator to generate fake images that are easier to be classified by Object Detection (Bai et al., 2018). We calculate classification loss (hereinafter, called L_{cls}) using Cross Entropy Loss for each D. The equation of L_{cls} for Dy is shown in Equation (4), where $D_y(G(x_i))$ and $D_y(y_i)$ denote the probabilities of the generated fake image y_j' and the real image belonging to the true category respectively.

$$L_{cls} = \sum_{i=0}^{n} -log\left(D_y(G(x_i))\right) - \sum_{j=0}^{m} log\left(D_y(y_j)\right) - \sum_{j=0}^{m} -log\left(D_x(F(y_j))\right) - \sum_{i=0}^{n} log\left(D_x(x_i)\right) \quad (4)$$

3.4.5 Overall Loss Function:
Equation of overall loss function is shown in Equation (5), which calculates the weighted sum of five aforementioned loss functions.

$$LOSS = L_{GAN} + \lambda L_{\text{cyc}} + \alpha L_{identity} + \mu(L_{cls}) \quad (5)$$

where λ, α, μ = weights

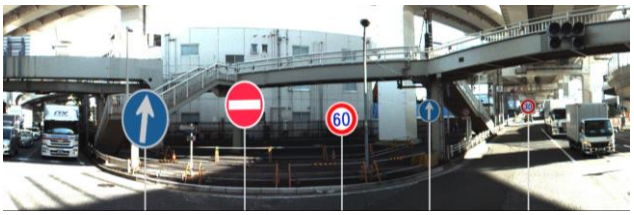
4. EXPERIMENTS

We take a traffic sign as the target object to evaluate our proposed method, SYN-MTGAN, in this section. More specifically, we focus on the traffic signs collected by MMS panorama images, which is due to two reasons. (1) A traffic sign is a common object along the roadside. And the distribution of the number of traffic signs, as shown in Figure 1, causes the training data collection problem aforementioned. (2) A traffic sign is an important object to be included in an HD map, meanwhile, its detection still remains an open challenge.

4.1 Dataset

Traffic Sign Number	#					50					30	40	60
Synthetic Image x	20,000	1693	1692	1657	1699	1662	1633	1657	1629	1682	1676	1609	1711
Real Image y	23,736	290	2011	1616	892	831	3715	28	30	10922	1727	1615	59
Synthetic Scene Image X	4,000	1677	1740	1628	1701	1680	1666	1696	1710	1655	1610	1594	1643
Real Scene Image Y	30,668	373	2082	2572	1502	1390	6312			18162	2919	2689	
Test Data	570	81	52	82	86	75	103	38	48	233	53	59	21

Table 1. The number of Samples



(a) Example of Synthetic Scene Image X

(b) Example of Real Scene Image Y

(c) Example of Template Image of Traffic Sign

(d) Example of Synthetic Image x

(e) Example of Real Image y

Figure 6. Example of Our Dataset

We prepared both synthetic and real images of a traffic sign since there are few open data of MMS panorama images that are collected in both urban and rural areas in Japan, which are necessary to train SYN-MTGAN. We collected 30,688 and 570 panorama images from several urban areas, inside and near Tokyo Prefecture, Japan, as train and test data respectively and labeled traffic signs with bounding boxes as training data. We targeted 12 categories of them, which concludes categories that have sufficient or insufficient samples. Categories with insufficient samples was included in test data only. The sample quantity of each category of real images is listed in Table 1. The procedure for generating a synthetic image x is as follows. (1) Collect one template image from the internet, as shown in Figure 6 (c), for each category of traffic signs. (2) Render the template image onto panorama images where traffic signs do not exist. The rotation degree, size, and brightness are randomly determined during the rendering. Result (hereinafter, called Synthetic Scene

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2020-443-2020 | © Authors 2020. CC BY 4.0 License.

Image X) examples of step (2) are shown in Figure 6 (a). (3) Crop a traffic sign region to generate synthetic image x (hereinafter, called Synthetic Image X). In addition, the location where Synthetic Image X locates in a panorama image is randomly determined. The category is exported as ground truth for training SYN-MTGAN. Real image y is generated by step (3) from real scene panorama images (hereinafter, called Real Scene Image Y). Examples of x and y are shown in Figure 6 (d) and (e). We generated 20,000 images of x and 23,736 images of y for training SYN-MTGAN. The sample quantity of each category is listed in Table 1. The distribution of size in pixel of synthetic image x and real image y is shown in Figure 7. The size of most training data samples is larger than 32^2 pixels, which indicates that the object size, more specifically the resolution of the object, is not a major difficulty in this study.

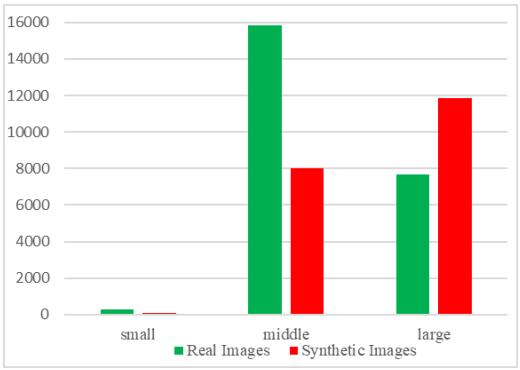


Figure 7. Size Distribution of Training Data. Small, middle, large refers objects whose size are smaller than 32^2 pixels, between 32^2 and 96^2 pixels, and larger than 962 pixels respectively.

The evaluation of the proposed SYN-MTGAN was conducted by comparing the performance of traffic detection using the classic Object Detection method, Faster R-CNN. Faster R-CNN was trained by three sets of data, Synthetic Scene Image X, Real Scene Image Y, and the result of SYN-MTGAN. The number of images par the category is shown in Table 1. Three categories of a traffic sign are not included in Y due to their number is less than 100.

4.2 Implementation Details

Our implementation of proposed SYN-MTGAN is based on open source machine learning platform PyTorch (Paszke et al., 2017). We mostly follow the original CycleGAN's implementation to train SYN-MTGAN. We use an image size of 256×256 for both x and y. We set the batch size to 24 and the epoch number to 120 for training. Weights of the loss function, which are hyper-parameters, are set inspiring by original SOD-MTGAN and CycleGAN experiments settings: $\lambda = 10, \alpha = 0.5, \mu = 0.5$ Faster R-CNN is implemented on another open-source machine learning platform, Chainer (Tokui et al., 2015). We use an image size of 3,000×1,125 to train Faster R-CNN. The initial learning rate is set to 0.0001 and reduced every 2 epochs. The epoch number is 5 as we use pre-trained VGG16 (Simonyan et al., 2015) on ImageNet (Russakovsky et al., 2015) as a backbone network for training Faster R-CNN, which leads to quick convergence. The rest of the parameters follow the original Faster R-CNN's implementation details.

4.3 Experiment Results

We carry out several experiments to evaluate our proposed method. First, we compare the performance of Faster R-CNN on traffic sign with four kinds of training data, SYN-MTGAN result (fake image y'), synthetic scene image X, the result of original CycleGAN and real scene image Y. Second, we make an ablation study to evaluate three components of proposed SYN-MTGAN.

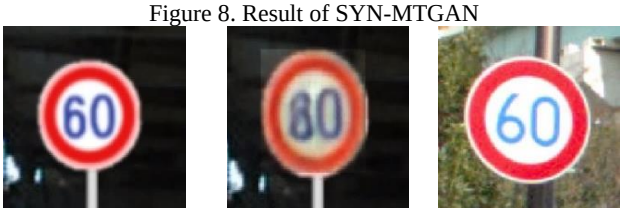
4.3.1 SYN-MTGAN Evaluation Results: Examples of the result images of SYN-MTGAN are shown in Figure 8, along with the inputs which are synthetic image x and real image y. The examples show that SYN-MTGAN has the ability to transfer the style of real images, i.e. texture, resulting in generating more realistic images compared to a synthetic image x. Besides, the results of SYN-MTGAN indict that it can reproduce a more realistic edge than x. The edge between traffic sign and background in x is sharper than y. The reason is that x is generated by rendering a template image onto a real scene panorama image, while the edge is blurred using a Gaussian filter with fixed parameters. This blur process cannot reproduce the real scene. Furthermore, the brightness of the traffic sign is edited to correspond with background by SYN-MTGAN, which makes them more natural comparing to synthetic image x, as shown in Figure 8 (c).

(a) Example of Input (Real Image y)

(b) Example of Input (Synthetic Image x)

(c) Example of SYN-MTGAN Result

Figure 8. Result of SYN-MTGAN



(a) Synthetic Image (b) SYN-MTGAN Result (c) Real Image

Figure 9. Example of the Sign of Speed Limit of 60 km/h

The quantitative evaluation result on real images dataset of traffic signs collected by us (clarified in Section 4.1) is shown in Table 2. Faster R-CNN trained using the samples created by our proposed SYN-MTGAN reached mAP of 0.77, which is better than training Faster R-CNN using the result of the original CycleGAN and the synthetic scene image X. Even though a real scene image Y reached higher performance, SYN-MTGAN reached considerable accuracy on the categories that real scene images cannot collect, i.e. crossing ahead and no riding double. However, some categories, i.e. speed limit of 30 and 60 km/h, do not obtain expected performance. The reason is considered as an unstable generation for these categories, especially with numbers on them. For instance, some generated images of speed limit of 60 km/h are difficult to recognize as Figure 9 shows. This unstable result may also cause the generation of unnatural fake

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2020-443-2020 | © Authors 2020. CC BY 4.0 License.

images such as the crossing ahead in Figure 8 (c). This problem remains as future work for us.

Methods	mAP													
Synthetic Image	0.55	0.70	0.40	0.60	0.79	0.33	0.70	0.71	0.74	0.79	0.14	0.68	0.05
CycleGAN	0.71	0.73	0.69	0.78	0.78	0.69	0.91	0.85	0.86	0.78	0.34	0.86	0.29
SYN-MTGAN (ours)	0.77	0.88	0.72	0.84	0.87	0.72	0.91	0.91	0.85	0.89	0.58	0.83	0.28
Real Image	0.89	0.90	0.79	0.90	0.91	1.00	0.91			0.91	0.78	0.95	

Table 2. Accuracy of Faster R-CNN

4.3.2 Ablation Studies: To analyze the importance of each component of our proposed method, we conduct ablation studies by removing one of the proposed components, which is the classification branch and the proposed identity loss. We take original CycleGAN as a baseline. Furthermore, we also evaluate the localization branch proposed by SOD-MTGAN (Bai et al., 2018) which predicts the bounding box of target objects, similar to Faster R-CNN. Specifically, it predicts the central point's 2D coordination, width and height of the target object in an image. The localization branch is proved to be effective for encouraging the Generators to generate better super-resolution images and boosting Object Detection performance. We place the localization branch in the Discriminators, which is parallel to the discriminative branch and classification branch and calculates its loss, localization loss (hereinafter, called L_{loc}), using Equation (6). Specifically, we use a Smooth L1 Loss (Ren *et al.*, 2015), shown in Equation (7), for both Dx and Dy to calculate the loss of predicting bounding box. The weight for L_{loc} is set to 0.5 inspiring by original SOD-MTGAN.

$$L_{\text{loc}} = \sum_0^n S_{L1}\left(D_y(G(x_i)) - v_i\right) + \sum_0^m S_{L1}\left(D_y(y_j) - v_j\right) + \sum_0^m S_{L1}\left(D_x(F(y_j)) - v_j\right) + \sum_0^n S_{L1}(D_x(x_i) - v_i) \quad (6)$$

in which,

$$S_{L1} = \begin{cases} 0.5\theta^2 & if(|\theta| < 1) \\ |\theta| - 0.5 & otherwise \end{cases} \quad (7)$$

where $\theta = (t_{i,x} - v_{i,x}, t_{i,y} - v_{i,y}, t_{i,w} - v_{i,w}, t_{i,h} - v_{i,h})$

$t_y^i = D_y(G(x_i)) = (t_{i,x}, t_{i,y}, t_{i,w}, t_{i,h})$ denotes the predicted bounding box of Dy

$v_i = (v_{i,x}, v_{i,y}, v_{i,w}, v_{i,h})$ denote the ground truth of bounding box of sample x_i

x = x coordinate of bounding box
y = y coordinate of bounding box
w = width of bounding box
h = height of bounding box

The results of ablation studies are shown in Table 3. The result of Case 4 reveals that our proposed loss function can contribute more than 7% of the performance boost for training Faster R-CNN compared to Case 2. This result might indicate that restricting the change of background and emphasizing the change of foreground during training a GAN for generating images is an effective approach to improve the performance when these images are used to train an Object Detection model.

Case 3 shows that the classification branch contributes a 4% performance boost than the localization branch compared to Case 1. Architecture with the classification branch and the proposed identity loss (Case3) reached the highest performance (mAP=0.77) among all test cases. This result indicates that the classification branch in Discriminator can encourage GAN to generate images which contain more information that is helpful to Object Detection. To be specific, such information can be well extracted by feature extract CNN of object detector such as Faster R-CNN and raise detection performance. On the other hand, the localization branch could only provide a limited performance boost for predicting the location of an object. The reason is considered as follows. Faster R-CNN predicts the bounding box of target objects from feature maps extracted by VGG16, whose size is 1/16 of the original image. This down-sampling process could cause ambiguity of bounding box prediction and degrade information that the localization branch has encouraged to generate.

Methods	loc	cls	loss	mAP												
Baseline (CycleGAN)	-	-	-	0.71	0.73	0.69	0.78	0.78	0.69	0.91	0.85	0.86	0.78	0.34	0.86	0.29
Case 1	✓	-	✓	0.73	0.83	0.69	0.82	0.82	0.76	0.91	0.88	0.84	0.81	0.44	0.83	0.18
Case 2	✓	✓	-	0.65	0.72	0.64	0.72	0.85	0.57	0.90	0.87	0.58	0.78	0.21	0.72	0.25
Case 3 (Proposed SYN-MTGAN)	-	✓	✓	0.77	0.88	0.72	0.84	0.87	0.72	0.91	0.91	0.85	0.89	0.58	0.83	0.28
Case 4	✓	✓	✓	0.72	0.82	0.74	0.73	0.81	0.70	0.91	0.86	0.86	0.80	0.41	0.82	0.23

where, loc = Localization Branch, cls = Classification Branch, loss = Proposed Identity Loss

Table 3. Result of Ablation Studies

Furthermore, training the classification branch and the localization branch simultaneously might aggravate the performance of Faster R-CNN according to the result of Case 2. The reason is considered as that, parameters of a discriminator with three tasks to predict at the same time are hard to be optimized simultaneously. Specific training schedule i.e. optimizes the parameters of a certain branch while keep others fixed, is a considerable solution to this problem. It remains as a future work for us.

5. CONCLUSION

In this study, we proposed a multi-task GAN architecture, SYN-MTGAN, to generate fake images from synthetic scene images, which can be used as training data for an object detector such as Faster R-CNN when the target objects in the real scene are hard to be collected. We carried out experiments based on traffic signs to evaluate our proposed method. The results showed that Faster R-CNN trained by the result of SYN-MTGAN with the classification branch and the proposed identity loss can reach mAP of 0.77. Some categories of the traffic signs which cannot be collected enough in the real scene have reached mAP of higher than 0.8, which indicates that the proposed SYN-MTGAN is an effective method to generate synthetic training data for rare roadside objects. Results of ablation studies indicate that our proposed components are effective. The multi-task training, especially the classification branch and loss function encourages GAN to generate synthetic images that are more suitable to be detected by deep learning-based Object Detection model.

For future works: (1) evaluate the human performance in distinguishing the synthetic and real images to see if the generated images are really realistic, (2) evaluate settings of loss function weights, which are the hyper-parameters of SYN-MTGAN, (3) modify the architecture to be end-to-end trainable.

REFERENCES

Abu Alhaija, H., Mustikovela, S. K., Mescheder, L., Geiger, A., Rother, C., 2018. Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes. *Int J Comput Vis*, 126, 961-972. doi.org/10.1007/s11263-018-1070-x.

Bai, Y., Zhag, Y., Ding, M., Ghanem, B., 2018. SOD-MTGAN: Small Objec Detection via Multi-Task Generative Adversarial Network. *(eds) Computer Vision – ECCV 2018. Lecture Notes in*

Computer Science, 11217, 210-226. doi.org/10.1007/978-3-030-01261-8_13.

Geiger, A., Lenz, P., Urtasun, R.,2012. Are we ready for autonomous driving? the KITTI vision benchmark suite. *2012 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3354-3361. doi.org/10.1109/CVPR.2012.6248074.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 580-587. doi.org/ 10.1109/CVPR.2014.81.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative Adversarial Nets. *Advances in Neural Information Processing Systems 28 (NIPS2014)*, 2672-2680.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep Residual Learning for Image Recognition. *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778. doi.org/10.1109/CVPR.2016.90.

Hinterstoisser, S., Lepetit, V., Wohlhart, P., Konolige, K., 2017. On pre-trained image features and synthetic images for deep learning. In arXiv:1710.10710.

Karras, T., Laine, S., Aila, T., 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4396-4405. doi.org/10.1109/CVPR.2019.00453.

Krizhevsky, A., Sutskever, I., Hinton, G., 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 26 (NIPS2012)*, 1097-1105. doi.org/10.1145/3065386.

Lin, Y., Takeda, H., Suzuki, K., Takahashi, G., Nakamura, K., 2018. A Study on Object Detection from Omnidirectional Camera Image u sing Deep Learning. *Journal of Applied Survey Technology 29 (2018)*, 95-106. (published in Japanese)

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, R., Fu, C., Berg, C., 2016. SSD: Single Shot MultiBox Detector. *(eds) Computer Vision – ECCV 2016. Lecture Notes in Computer Science*, 9905, 21-37. doi.org/10.1007/978-3-319-46448-0_2.

Mori, Y., Kohira, K., and Masuda, H., 2018. Classification of Pole-like Objects Using Point Clouds and Images Captured by Mobile Mapping Systems, *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci*., XLII-2, 731-738. doi.org/10.5194/isprs-archives-XLII-2-731-2018.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, Adam., 2017. Automatic differentiation in PyTorch. *Advances in Neural Information Processing Systems 31 Autodiff Workshop (NIPSW2017).*

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H., 2016. Generative Adversarial Text to Image Synthesis. *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML2016)*, 48, 1060-1069. https://doi.org/10.5555/3045390.3045503.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788. doi.org/10.1109/CVPR.2016.91.

Ren, S., He, K. Girshick, R., Sun, J., 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems 29 (NIPS2015)*, 91-99.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., L, F., 2015. ImageNet Large Scale Visual Recognition Challenge. *Int J Comput Vis*, 115, 211–252. doi.org/10.1007/s11263-015-0816-y.

Shrivastava, A., Gupta, A., Girshick, R., 2016. Training Region Based Object Detectors with Online Hard Example Mining. *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 761-769. doi.org/10.1109/CVPR.2016.89.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. *in International Conference on Learning Representations 2015 (ICLR2015).*

Tokui, S., Oono, K., Hido, S. and Clayton, J. 2015. Chainer: A Next-Generation Open Source Framework for Deep Learning. *In Proceedings of Workshop on Machine Learning Systems (LearningSys) in Conference on Neural Information Processing Systems 29 (NIPS2015).*

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani,V., Anil, C., To, T., Cameracci, E., Boochoon, S., Birchfield, S., 2018. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1082-10828. doi.org/10.1109/CVPRW.2018.00143.

Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders. A.W.M., 2013. Selective Search for Object Recognition. *Int J Comput Vis*, 104, 154-171. doi.org/10.1007/s11263-013-0620-5.

Ulyanov, D., Vedaldi, A., Lempitsky, V., 2016. Instance normalization: The missing ingredient for fast stylization. In arXiv:1607.08022.

Wolf, J., Richter, E., Discher, S., Dollner, J., 2019. Applicability of Neural Networks for Image Classification on Object Detection in Mobile Mapping 3D Point Clouds. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci*., XLII-4/W15, 111-115. doi.org/10.5194/isprs-archives-XLII-4-W15-111-2019.

Wu, B., Wan, A., Yue, X., Keutzer, K., 2017. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. In arXiv:1710.07368.

Zhu, J., Park, T., Isola, P., Efros, A., 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Network. *2017 IEEE International Conference on Computer Vision (ICCV2017)*, 2242-2251.