VEHICLE DETECTION ON SMALL SCALE DATA BY GENERATIVE DATA AUGMENTATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HILMI KUMDAKCI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MODELLING AND SIMULATION

FEBRUARY 2021

VEHICLE DETECTION ON SMALL SCALE DATA BY GENERATIVE DATA AUGMENTATION

submitted by **HILMI KUMDAKCI** in partial fulfillments of the requirements for the degree of **Master of Science in Modelling and Simulation Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, Graduate School of **Informatics** _____

Assist. Prof. Dr. Elif Sürer
Head of Department, **Modelling and Simulation** _____

Prof. Dr. Alptekin Temizel
Supervisor, **Modelling and Simulation, METU** _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: HILMI KUMDAKCI

Signature :

ABSTRACT

VEHICLE DETECTION ON SMALL SCALE DATA BY GENERATIVE DATA AUGMENTATION

Kumdakcı, Hilmi
M.S., Department of Modelling and Simulation
Supervisor : Prof. Dr. Alptekin Temizel

February 2021, 54 pages

Scarcity of training data is one of the prominent problems for deep neural networks, which commonly require high amounts of data to display their potential. Data augmentation techniques are frequently applied during the pre-training and training phases of deep neural networks to overcome the problem of having insufficient data for training. These techniques aim to increase a neural network's generalization performance on unseen data by increasing the number of training samples and provide a more representative distribution to the system during training. In this work, we focus on improving vehicle detection in aerial images by proposing a data augmentation method that does not need any extra supervision than the bounding box annotations of the vehicle instances in the training data. The methods we used are based on a conditional Generative Adversarial Network (cGAN). The proposed method is not exclusive and can be used in association with classical augmentation techniques to further improve object detection performance. We showed that the proposed data augmentation method increases the Average Precision by up to 25.2%, 32.7%, and 25.7% when integrated with Pluralistic, PSGAN, and DeepFill respectively.

Keywords: Data Augmentation, Generative Adversarial Networks, Aerial Imaging, Object Detection

ÖZ

KÜÇÜK ÖLÇEKLİ VERİLERDE ARAÇ TESPİTİ İÇİN ÜRETKEN METODLARLA VERİ ARTIRMA

Kumdakcı, Hilmi
Yüksek Lisans, Modelleme ve Simülasyon Bölümü
Tez Yöneticisi : Prof. Dr. Alptekin Temizel

Şubat 2021 , 54 sayfa

Eğitim verilerinin yetersiz olması durumu, derin sinir ağlarının potansiyellerini açığa çıkarmasında en önde gelen sorunlardan biridir. Veri eksikliği probleminin üstesinden gelmek için, derin sinir ağlarının eğitim öncesi ve eğitim aşamaları sırasında veri artırma tekniklerine sıklıkla başvurulur. Bu teknikler, eğitim örneği sayısını ve örneklerin dağılımını artırarak bir sinir ağının belirli bir test seti için performansını artırmayı amaçlamaktadır. Bu çalışmada, sadece eğitim verilerindeki araç sınıfına ait nesnelerin sınırlayıcı kutu etiketleri kullanılarak bir veri artırım yöntemi önerilip, havadan çekilmiş az miktarda bulunan görüntülerde araç tespiti performansı geliştirilmeye odaklanılmıştır. Önerilen yöntemin koşullu üretken çekişmeli ağların (cGAN) farklı varyasyonlarıyla beraber çalışabileceği ve performansı iyileştirilebileceği gösterilmiştir. Önerilen yöntemdeki modüller eşsiz olmamakla birlikte değiştirilebilir veya modifiye edilebilir. Bununla birlikte obje tespit performansını daha da arttırmak için klasik data artırımı teknikleri ile birlikte kullanılabilir. Önerilen veri artırma yöntemi, Pluralistic, PSGAN, ve DeepFill üretken ağları ile birlikte kullanıldığında ortalama hassasiyet hesaplamasında sırasıyla 25.2%, 32.7%, ve 25.7% seviyelerine kadar artışlar sağladığı gösterildi.

Anahtar Kelimeler: Veri Büyütme, Çekişmeli Üretici Ağlar, Havadan Görütüleme,

Nesne Tespiti

To my family...

ACKNOWLEDGMENTS

Foremost, I would like to present my gratitude to my research supervisor, Prof. Dr. Alptekin Temizel for giving me the opportunity to do research and providing his valuable guidance and constant support in this study.

Thank you to Mine Tosun, for all her love and limitless support. I would also like to thank my mother, my father, my brother, and my little sister. Without their support, I wouldn't be able to persist throughout my journey.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
LSTM	Long Short Term Memory
CGAN	Conditional Generative Adversarial Network
IoU	Intersection over Union
AP	Average Precision
mAP	mean Average Precision

CHAPTER 1

INTRODUCTION

State-of-the-art results for visual domain tasks such as face detection [1][2], object detection [3][4][5][6], visual question answering [7], image captioning [8], image segmentation[9][10], are generally obtained with a task specific supervised algorithms. Supervised algorithms can be evaluated based on their accuracy, speed, and the number of parameters depending on the problem type and requirements. These algorithms are expected to achieve a satisfactory generalization performance, by performing well on unseen test samples, different from those available in training, and this is a fundamental performance definition in several real-life scenarios. For a given input vector x, a corresponding output vector y with a corresponding type of annotation format specific to the given visual task is provided. A typical supervised algorithm searches for the optimal parameters to find the ideal fit between x and y, based on a predefined set of rules and learning directions in the training phase. Other than supervised methods, unsupervised methods are a popular set of machine learning algorithms. These methods aim to discover hidden patterns covered by existing input data and use these hidden patterns to infer an output without direct supervision and dependence on labeled data. Another paradigm, reinforcement learning, is based on learning parameters through the decisions of an agent and the output of these decisions to maximize the reward. In reinforcement learning, instead of directly feeding the network with annotated data, the aim is to make the parameters learnable through an intelligent agent by discovering the environment itself.

Whether supervised or not, in order to successfully train a neural network and obtain its optimal parameters, a large amount of data is required for most tasks. This training

data can be provided into the neural network to train it from scratch or continue to update its weights using pretrained models which are trained with different datasets early on. For the target task, data collection may not be sufficient as the raw data obtained from the source may not be in the desired format. In such cases, some data preprocessing steps are needed to prepare data for use in training and testing, which brings extra costs. On the other hand, when the data collection process is not feasible and it does not allow collecting the required amount of data, due to the high amount of neural network parameters, models may end up learning suboptimal node values and still manage to fit the input-output relationship dictated by supervision in the training phase. This problem is very common in the training of neural networks and is defined as the overfitting problem. From the perspective of computer vision, even if the existing data seem sufficient to train a model and get meaningful results, there may be shortcomings in the algorithm's robustness. At this stage, testing of the trained model with different unseen scenarios is an important step to check whether there is overfitting or not.

Another problem that should be overcome is the difference between distributions of training and unseen test data in terms of their domains. It happens when there is not an easy way to collect the data coming from test distribution but there is an abundance of training data that belong to a different domain. When the training data is different from unseen test scenarios, then neural networks tend to learn the complex patterns for training samples only and might fail in test scenarios, especially when there are significant amounts of domain shift. For example, if the aim is to detect pedestrians from a street view and training samples only contain images taken under the sun, the model will be biased towards detecting pedestrians under sunny conditions and the performance in different weather conditions will be inferior. This is because of the fact that sunlight affecting pedestrian instances' pixel distribution in the given training data. To improve robustness and remedy the unbalanced data distribution, more synthetic or real data representing different pixel distributions could be added as an alternative to designing more domain specific algorithms.

Social and other commercial types of media platforms have a major role in supplying various types of multimodal, abiding by their own data usage rules. However, even though large amounts of data can be collected through them, this data cannot be

used without task-specific annotations. While there are several publicly available annotated data sources, such as Pascal VOC [11] and COCO [12], their annotated classes are highly distributed and the amount of data is not sufficient for various specific visual tasks. While there are several available commercial data sources as an alternative to publicly available or self-collected datasets, they have high costs and usage restrictions. This necessitates collecting data for specific tasks, which comes with a high cost in terms of the acquisition, annotation, and legal complications of capturing the data in many cases.

To mitigate the challenges stated above, many people attempted to apply geometrical and color based transformations into training samples to increase the number of samples for training. More recently, the usage of generative networks have become widespread in many fields of computer vision, and they are used to generate synthetic data for the data augmentation problem and add diversity to existing data.

1.1 Problem Statement and Motivation

Image classification, object detection, and object segmentation are some of the most important computer vision problems. Image classification aims at classifying images based on predefined classes. When the location information is also provided, in addition to the class information, the problem can be defined as object detection. This location information could be in the form of bounding boxes which can be oriented or based on its extreme points as shown in Fig. 1.1. The performance of an object detection algorithm is mainly evaluated by its accuracy in terms of localization and classification based on predetermined classes, as well as the computational complexity of the algorithm. Data related issues, mainly its quality and scale, are fundamental in both training and testing phases to obtain a model with sufficient performance. While quality can be defined regarding the amount of noise in the dataset, the scale can be considered the amount of the training and test data. The noise may exist in images themselves as well as their corresponding annotations. In images, noise can be described as deviations from ideal data. The source of this noise could be data acquisition or preprocessing steps. Label-noise may appear as a result of mislabeling due to human errors or environmental factors. The scale of the data is another fundamen-

(a) Oriented bounding box

(b) Extreme point based bounding box

Figure 1.1: Different bounding box annotation types

tal factor in the development of successful methods. Due to the wide range of image classification and object detection research, there are different available approaches to solve different scale problems based on the amount of data [13], [14], [15], and different size of instances for object detection task [16][17], etc. Supervised methods, where the aim is to detect and localize target instances, require the existence of correctly annotated target data. Unfortunately, for some target instances, the required amount of data might not be available for free or more samples need to be collected to reach target performance. In such cases, the most common and inexpensive solution is to apply data augmentation techniques to existing samples.

There are several traditional data augmentation methods in the literature and these methods have been proven to be effective in many research areas and commercial projects. However, even though there are several traditional methods to augment data to improve performance, there are no formal rules that could help obtain the best augmentation strategy to be followed under different conditions. There may not be room for improvement for some problems using data augmentation, and it is also possible to get worse performance than using only raw data. Hence, evaluation of these augmentation methods should be carefully done by specifying the proper evaluation criteria and repeating experiments with and without applying data augmentation techniques. Another main drawback of traditional data augmentation techniques is that

they rely only on the image pixel intensities, not the given image's deep features or semantics.

To benefit from deep features of the image to be augmented, attributes of instances or images should be extracted and formulated with different methods. Using convolutional layers is one of the methods of how objects in an image or image itself map into feature maps. It has been shown that convolutional neural networks(CNNs) [18] are performing well on extracting features from images [19] and they can be used for image clustering purposes based on the extracted deep features. To test the quality of these extracted deep features, a decoder module can be used to reconstruct the original image from these features, which is forming autoencoders together with the encoding part. Training of these autoencoder networks for feature extraction tend to give better results when it is trained with the similar type of images which feature extraction is aimed to be performed on. In this way, more discriminative features can be observed in deep features.

Recently, several neural networks have used these types of deep features and learned weights to generate new instances which are grouped as generative networks. These approaches basically try to learn underlying distribution of training data, and by feeding a noise as an input, they result in generating samples with similar looking but with differences in semantic features regard to training samples. By adding this type of variety and preserving training data distribution, these networks help to generate synthetic but realistically looking samples.

1.2 Scope and Contributions of the Thesis

The main focus of this study is improving detection performance for vehicle detection in aerial images when there are limited amounts of data. For this purpose, a data augmentation scheme, which employs generative networks, has been proposed. The results of the study have been experimentally validated using different generator structures and different data amounts for different use-case scenarios.

This thesis has two main contributions: As a data augmentation framework, a scheme of the generative network combined with a detector network is developed to improve

object detection task performance. A comprehensive comparative study is conducted to understand the effect of the detector network and the amount of existing data in the proposed scheme. Some methods and results of this thesis has been published in [20].

1.3 Outline

- In Chapter 2, we provide a comprehensive review of traditional and deep learning based data augmentation techniques in literature
- In Chapter 3, Our data augmentation scheme that consists of a generative network combined with an object detection network is presented with a comprehensive experimental work
- In Chapter 4, we conclude the thesis and illustrate possibilities for future improvements and investigations

CHAPTER 2

DATA AUGMENTATION LITERATURE

Due to a large number of parameters in deep learning-based systems, overfitting and lack of generalization problems are likely to happen. Hence, in various vision-related tasks, accessing large amounts of data is a key for achieving higher performance in real-life scenarios, which makes data collection and acquisition a critical element. On the other hand, data acquisition is a costly process and it is desirable to achieve good performance on smaller datasets. Supervised image classification algorithms aim to assign input images into predefined classes along with their confidence. Hence, the annotation required for this task involves the labeling of images with their respective classes. Object detection algorithms aim to return bounding box coordinates which are fully enclosing the objects along with their class information and their respective confidence. Supervised training of these algorithms necessitate annotations of object locations and object classes for each target object in an image. These bounding boxes should contain an object's location information to enclose all pixels that define the object. A comparison of the requirements of these tasks reveals that object detection tasks require much more time and budget in terms of annotation effort. To overcome these budget and timing issues related to the scale of the data, there are different approaches to make this process more feasible. These approaches can be investigated under three topics, which are traditional data augmentation, rendering based data augmentation, and neural network based data augmentation. Traditional data augmentation methods rely only on the pixel values of images, whereas neural network based methods also take the semantic information into account. Rendering based data augmentation techniques make use of 3D modeling software programs to insert 3D models into 2D images, where the bottleneck is the amount of 3D data in

this scenario. The key advantages and drawbacks of using these different approaches will be discussed in the following subsections.

2.1 Traditional Data Augmentation Methods

Even though they do not guarantee to improve the performance of a model on a test set, classical data augmentation methods are widely accepted as a primary solution to overcome data scarcity issues due to their ease of implementation. These techniques are also generally useful and applicable for the object detection task, which is the main focus of the thesis. These methods can be mainly grouped as geometrical and color based transformations. Geometric methods alter an image or appearance of an object by geometric transformations. When applied to an image, the shape of the object within the image is preserved but the orientation and position may change based on the location where the object location within an image. If bounding boxes are required for the task, their raw annotations need to be modified according to the geometric transformation type applied to the class instance. On the other hand, color-based changes in the images or patches do not require modifications in bounding box annotations, unlike the geometric methods, since they change the values of given channel pixels without changing their location. Within these main groups, each transformation method has its own advantages, depending on the problem, each might work much better than other similar techniques under different experimental setups. When the main requirement for the problem is robustness to varying illumination conditions which might be due to weather conditions and if the object is geometrically simple, color-based augmentation techniques might work better than geometric augmentation techniques since the model would still be able to capture the shape properties. Hence, if the orientation of a different object is changing in test images, then geometric transformations especially rotating, might be more helpful to improve the overall performance of the model.

The main disadvantage of these traditional methods is that they only rely on pixel intensities and do not consider the semantics of an image in the augmentation stage as their focus is not on exploring how deep features can be used and extracted from images. Even though it has these drawbacks, they are commonly used in indus-

trial applications and academic works as they provide promising results while being generic and directly applicable to any class. In the work conducted by Taylor and Nitschke [21], a benchmark consisting of different traditional data augmentation techniques, namely flipping, rotating, cropping, color jittering, edge enhancement, and fancy PCA, have been compared. According to their results, cropping is the most effective one, whereas geometric ones(flipping, rotating, cropping) mostly outperformed photometric methods(color jittering, edge enhancement, fancy PCA) in the Caltech101 dataset. Another traditional data augmentation method [22], called *Random Erasing*, is based on removing random parts of images. They have shown that *Random Erasing* method outperforms no-augmentation in different tasks, such that image classification, object detection, and person re-identification. In the remainder of this section, the most popular traditional data augmentation methods are explained with examples.

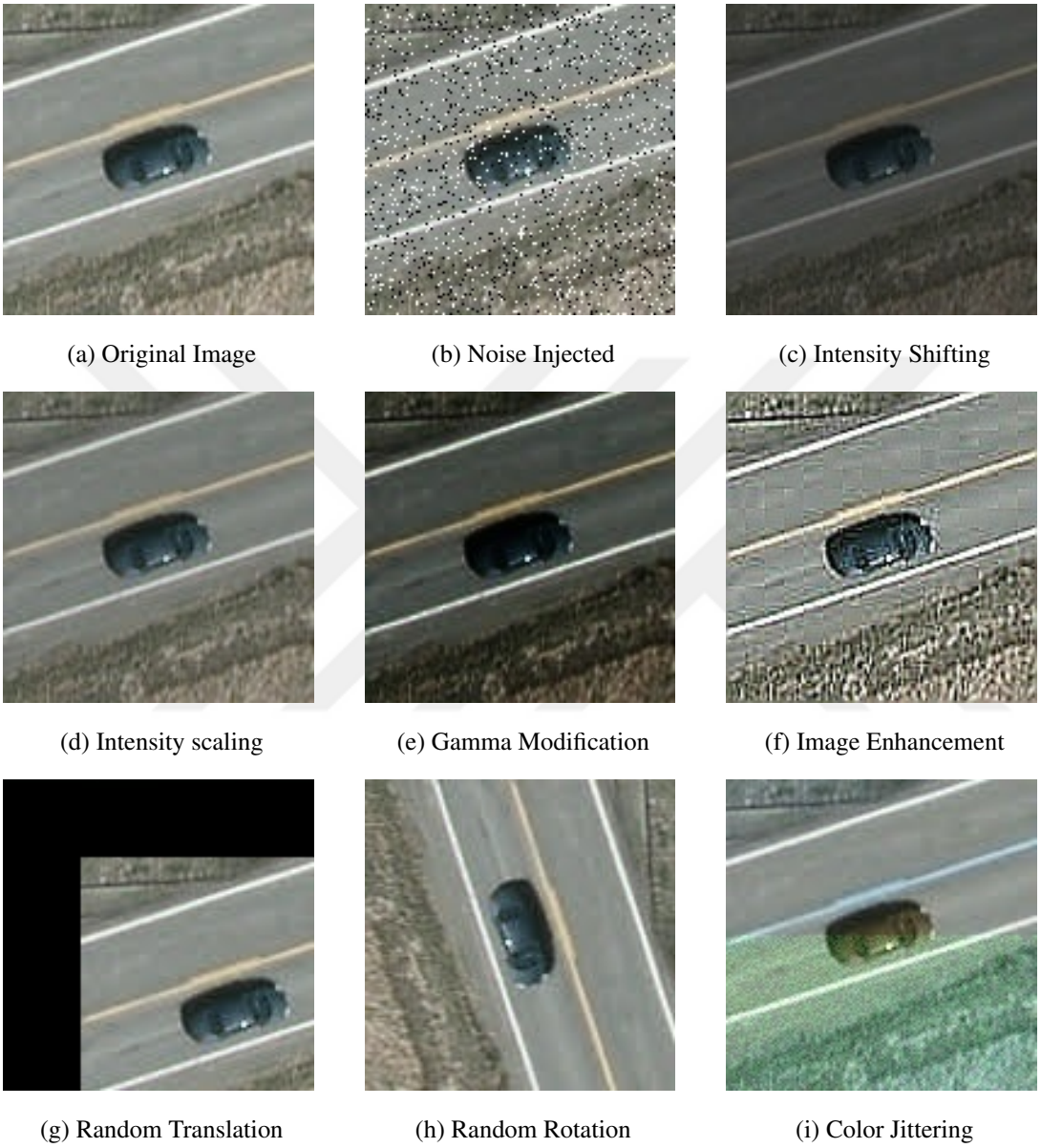
2.1.1 Noise Injection

Noise can be defined as random variations in the captured visual data through a camera system. Noise addition into training images can be seen as one of the most applied strategies for data augmentation. It may help the data to be robust against different cameras where their ISO settings may cause noise appearance. There are different types of noise addition methods such as Gaussian, Poisson, and Salt-and-pepper. The type of noise to augment the data should be determined based on the system specifications and requirements. In the figure 2.1b, Salt-and-pepper noise can be seen. In general, noise injection can be defined as in the equation 2.1:

$$I_{augmented} = I_{original} + N(\mu, \sigma) \tag{2.1}$$

2.1.2 Intensity Shift

Sometimes the train and test distribution might not match in terms of intensity values and this may create accuracy problems. This mismatch problem can be prevented by shifting the intensity of images or instances. To apply this intensity transformation



(a) Original Image (b) Noise Injected (c) Intensity Shifting

(d) Intensity scaling (e) Gamma Modification (f) Image Enhancement

(g) Random Translation (h) Random Rotation (i) Color Jittering

Figure 2.1: Effects of various data augmentation techniques

for instances, there needs to be a segmentation annotation exists. Intensity shift corresponds to the addition of each pixel intensity with a constant value. In the figure 2.1c, a patch becomes darker by a negative amount of intensity shift. If we define a signed constant *c* for shifting, the formulation can be described as in the equation 2.2:

$$I_{augmented} = I_{original} + c \tag{2.2}$$

2.1.3 Intensity Scaling

Intensity scaling corresponds to the element-wise multiplication of each pixel intensity with a constant value. In the figure 2.1d, a patch gets darker with a scale factor less than 1. If it is more than 1, the image gets brighter. This scale factor should be carefully selected since the intensity value will be clipped into the upper bound in the case of exceeding the upper limit. The formulation can be described as in the equation 2.3:

$$I_{augmented} = I_{original} * N(\mu, \sigma) \tag{2.3}$$

2.1.4 Gamma Correction

Gamma correction is a nonlinear intensity transformation technique that tries to rearrange pixel ranges. The algorithm is based on the γ parameter and reduce the number of possible intensity ranges to reduce the number of allocated bits in some applications other than data augmentation purpose. In the figure 2.1e, γ value has been set such that the image looks darker within a small amount of discrete number range. The gamma correction can be formulated as in the equation 2.4:

$$I_{augmented} = I_{original}^{1/\gamma} \tag{2.4}$$

2.1.5 Image Enhancement

Image enhancement modifies the edges in the images by intensifying them. This transformation might help CNNs to discover descriptive features. In the equation 2.1f, the coefficients can be played to arrange according to desired enhancement level. An example to enhance the version of an image can be found in 2.1f. The example transformation matrix and the equation can be expressed as in 2.5:

$$I_{augmented} = \begin{vmatrix} -1 & -1 & -1 \\ -1 & 10 & -1 \\ -1 & -1 & -1 \end{vmatrix} * I_{original} \tag{2.5}$$

2.1.6 Random Translation

Random translation can be performed by defining two variables, t_x and t_y. These variables are used to indicate the amount of shift along the specified dimension. As in the given equation 2.6, transformed locations become $x\prime$ and $y\prime$. As shown in Fig. 2.1g, this method might cause to lose some info from the image. In this scenario, the suitable border padding method should be used to fill the lost area of the image.

$$x\prime = x + t_x, \quad y\prime = y + t_y \tag{2.6}$$

2.1.7 Random Rotation

Random rotation requires one parameter, which is the rotation angle θ. The rotated coordinates can be obtained by multiplying x and y locations as given in equation 2.7.In the given figure 2.1h, 90° rotation applied into the original image which is shown in 2.1a by equating $\theta = \pi/4$ in equation.

$$x\prime = x * \cos(\theta) - y * \sin(\theta), \quad y\prime = x * \cos(\theta) + y * \sin(\theta) \tag{2.7}$$

2.1.8 Color Jittering

Color jittering can be defined as randomly changing the hue component of an image when it is projected into the HSV domain. As shown in Fig. 2.1i, random jitter operations have been applied into different locations of an image. When the random constant is defined as c, the augmentation equation becomes:

$$I_{augmented} = HUE(I_{original}) + c \tag{2.8}$$

2.2 Data Augmentation with Rendering

Availability of sufficient computational power and open 3D repositories on the web made the data augmentation using various rendering tools possible. With the use of rendering tools, in addition to the orientation and shape of the object, its illumination, pose, and scale can also be controlled. By modification of these parameters, a large number of samples can be rendered and inserted into the training data. In [23], a technique is proposed to create datasets automatically for viewpoint estimation. They have shown that training with synthetic images suffers from domain adaptation problems, but combining synthetic ones with natural examples gives improved results with respect to using only natural ones. [24] proposes to generate synthetic training data for optical flow estimation task, which is hard to find the desired amount of annotation since it needs a pixel-accurate labeling process. They have rendered chair instances to perform experiments. They have outperformed previous datasets in different test scenarios that do not contain chairs. So, it has been shown that variety on the data is the key to create a dataset rather than having domain specialized data, at least for optical flow estimation tasks. In [25], it is proposed to synthesize randomized data in terms of visual factors, such as lightning, pose, and textures. With less emphasis on realism, it outperformed VKITTI[26] dataset by training both datasets with different object detection algorithms and compared their accuracy. Test data has been selected as KITTI[27], where VKITTI is aimed to be rendered as close as possible to the KITTI dataset. It has been shown that a variety of data is more important than limited realistic data to get better results.

Image rendering based approaches commonly focus on only one class, and each additional require additional effort, especially when there is no open 3D repository for that respective class. Also, these rendering tools can only be used for vision based tasks, and cannot be applied to different domains, such as text and audio.

2.3 Data Augmentation using Neural Networks

Recently, neural networks have been used to extend the scope of data augmentation by generating objects from different backgrounds or determining the suitable locations for objects to be inserted within an image. In Section 2.3.1, some neural network approaches to find best matching between contextual area and segmented object masks are mentioned. In Section 2.3.2, data augmentation in feature space, exploiting the semantics of the data, is described. In Section 2.3.3, the use of generative networks for data augmentation is described.

2.3.1 Data Augmentation by Cutting and Pasting

In [28] [29], a matching score between an object and different regions of an image is calculated for proposing the location to insert an object within a segmented mask. Context-based CNN is employed with these approaches to assign a score between the context and the object. For the best matching pair of object and region, the object is scaled and blended into patches which obtain the highest score from the matching algorithm. To prevent boundary artifacts, they use segmented annotated instances while choosing their data to be augmented. This set of methods are based on copying instances into different images without any modification, which corresponds to without generative modeling.

2.3.2 Data Augmentation in Feature Space

Traditional data augmentation methods work in input space and hence they do not make use of the discriminative features that can be learned through statistical methods or with the help of neural networks. Neural networks allow reducing dimension-

ality by mapping inputs into low dimensional features by exploiting discriminative features. Visualizing the lower-dimensional data allows investigation of feature space and evaluation of the accuracy of the feature extraction step. Devries and Taylor [30] performed interpolation, extrapolation, and noise addition with the extracted feature vectors and provided an improvement in the task of visual classification in classifying feature vectors.

2.3.3 Data Augmentation using Generative Networks

Generative Adversarial Networks(GANs)[31] are introduced by Goodfellow et al. and has been quickly adopted by the researchers. The proposed baseline algorithm has been improved and iterated in many directions to solve different tasks from different domains. While some effort has been put to create domain specific solutions, there have been base improvements in the training of GANs as well to enable researchers and ML practitioners to generate objects with high quality and more diversity. Other than synthetic object generation within a specific class, there are also several works to refine generated objects with given specified viewpoints and angles indicated by textual or other input types[32]. There are some works[33][34] try to generate high-resolution images where the training of networks take longer than most of the works as a drawback. Conditional based style and domain transfer task is also one of the most popular applications where GANs are used[35][36][37]. GANs have many other applications such as object detection [38][39], single image and video based superresolution[40][41], photo inpainting[42], frame prediction in videos [43][44], , image blending [45][46], face aging [47],text-to-image synthesis [48] , generation of cartoon characters [49][50] and so on.While most of the works are built on some modifications upon base theoretical knowledge on GANs, there are some key works worth mentioning under this section.

2.3.3.1 Generative Adversarial Network

GANs are considered to be the most successful type of generative models in many domains. A typical GAN consists of two differentiable neural networks, which are

called generator *G* and discriminator *D*. These two differentiable neural networks have clear purposes: the generator network, *G*, aims to generate synthetic samples to deceive discriminator network by capturing the distribution of training data by using prior noise distribution, which is called latent noise vector and represented by *z*, while the discriminator *D* is trying to differentiate real and fake images and not to deceive by *G*. Here fake represents the data generated by *G*, and real represents the data coming from training samples. *G* is trained such that it minimizes to probability of $(D(G(z)) \approx 0$, whereas *D* is trained to maximize the probability of being correct $(D(G(z)) \approx 0$,$(D(x) \approx 1$. As a result, the minimax value function we want to optimize becomes as :

$$\begin{aligned} \min_G \max_D V(D,G) \; &= E_{x \sim p_{data}(x)}[log\ D(x)] + \\ &\quad E_{z \sim p_z(z)}[log\ (1 - D(G(z)))] \end{aligned} \tag{2.9}$$

where p_{data} represent the distribution of real data and p_z is the distribution of latent noise vector. Optimum state for this minimax game can be reached when $p_g = p_{data}$ where generated data distribution is becoming equal to the distribution of real training data. Although GANs are very successful in terms of improvement in generated samples, there are some significant problems with the training phase. Due to the fragility of optimization between generator and discriminator, the generator does not always produce various outputs because their training might fail, which results in a mode collapse. It mainly happens, if one of the networks becomes overpower with respect to the other one. In a mode collapse situation, the generator network rotates through a small set of output types and can not produce outputs with a large variety. The main reason behind the mode collapse can be seen as totally independent training of generator and discriminator and the lack of similarity measurement calculation within a given mini-batch. Moreover, this lack of similarity calculation pushes the generator to generate a single mode generation which gives the highest response from the discriminator. After the discriminator has learned this mode, the generator shifts to another location, which will lead the generator to the oscillation phase. To mitigate these training problems, Wasserstein loss[51] and mini-batch discrimination has been proposed later on.

2.3.3.2 Conditional Generative Adversarial Network

Conditional GAN [52] is the version of GAN which is trained with a condition *c*, which can be class label, image or text. The latent vector *z* and *c* are given as input to the generator whereas x_{real} and *c* are inputs to the discriminator where x_{real} denotes the real data. So, the formula becomes:

$$\begin{aligned} \min_G \max_D V(D,G) \; &= E_{x\sim p_{data}(x)}[log\ D(x|c)] + \\ &\quad E_{z\sim p_z(z)}[log\ (1 - D(G(z|c)))] \end{aligned} \tag{2.10}$$

It can be seen that the only difference is being conditioned on *c* in the equation. This paper has also shown CGANs can be used in multimodal scenarios by making experiments on image tagging data.

2.3.3.3 Deep Convolutional Generative Adversarial Networks

GANs have been extended with new loss functions and optimization techniques to improve performance. Radford et al. introduced Deep Convolutional Generative Adversarial Networks[53] with significant improvements which brought with some architectural modifications that are relying on applying novel training techniques.

These changes can be summarized as :

- It uses strided convolutions in discriminator and fractional-strided convolutions for the generator instead of pooling layers
- Batch Normalization has been used for both networks, generator and discriminator.
- ReLU has been replaced with all activation functions in generator,except the Tanh in the output.
- Leaky ReLU has been proposed for all layers in discriminator.

As the variety and quality improvements can be observed from the samples generated by DCGAN, the parameters to train this network should still be selected carefully. The authors also have shown that discriminator can perform as a feature extractor module that can be used for classification tasks and got reasonably well accurate results.

Other than these improvements, the authors have shown the interpolation between latent vectors is possible to capture smooth transitions. They have also shown that visual clues such as being happy and wearing glasses can be transferred into newly generated samples by arithmetic operation between latent vectors which have been defined with z in the previous subsection.

2.3.3.4 Wasserstein Generative Adversarial Networks

Wasserstein Generative Adversarial Networks(WGANs) has been introduced by Arjovsky et al. [51] by proposing a new loss function. Given the real data distribution p_{real} and the model distribution p_G, the main motivation behind WGAN is to analyze the distance between these two distributions in a different way. Unlike the original GANs introduced by Goodfellow et al. which use Jensen-Shannon (JS) divergence, they proposed to use Earth-Mover (EM) distance or Wasserstein-1 distance to use as a distance function between these two distributions, p_{real} and p_G. When we incorporate the generator network *G(z)* as model distribution p_G, the formulation which should be optimized can be defined as:

$$\begin{aligned} \min_G \max_D V(D,G) \; &= E_{x\sim p_{real}(x)}[D(x)] - \\ &\quad E_{z\sim p_z(z)}[D(G(z))] \end{aligned} \tag{2.11}$$

where *D* is called critic instead of a discriminator in this case. The parameters of the critic are represented as θ_D which is lying in a compact space. To satisfy the requirement of lying in a compact space, Arjovsky et al. suggest clipping the critic weights with a suggested range [-0.001,0.001] , after each gradient update.

After optimization, the Wasserstein-1 distance shows better results compared to JS

divergence. Hence, due to the continuity and differentiability of EM distance, it is suggested that to train critic till optimality, to get more reliable gradients. Also training the critic till optimality prevents the mode collapse problem.

2.3.3.5 GAN Approaches for Data Augmentation

Other than significant developments stated in sections related to the theoretical foundation of GANs, there are also purpose specific GANs that aim to perform data augmentation. In [54], the ideal locations to place an object together with the best fitting pose of an instance for that scene are estimated. The idea is to provide locations for inserting objects into semantic maps using semantic segmented images to train generator network, since the purpose is placing generated objects visually plausible places in semantic maps, this idea is not defined as an augmentation for detection or classification tasks. [55] handles this contextual instance insertion problem by using a neural network based architecture having two discriminators and one generator. One of the discriminators is responsible for generating the instances and the other generates suitable patches for the generated instances. It uses a spatial pyramid pooling layer to generate varying size instances. However, this approach has to deal with instances with artifacts due to the nature of generative networks.VS-GAN [56] also uses a 2 stage discriminator strategy for vehicle detection using least square loss in the training of generator, and validates the augmentation with YOLOv3 and RetinaNet detectors. Since it is required to generate high quality synthetic samples, it was trained with a large scale dataset having car instances. DetectorGAN [57] has added a detector network into the generator-discriminator loop of PSGAN network to have more realistic outputs. This approach is highly dependent on training the discriminator branch which might require training parameters to be changed per subject. Another approach [58] for medical studies keeps geometric and intensity information intrinsically while generating instances. In aerial images, the work that is most related to ours, [59] augments aerial images using an image-to-image translation by conditional GANs. It is based on mapping the layout into another one while keeping instances that require layout annotation. In our framework, we do not aim to propose a new generator model but use generator and detector modules separately which is a generic approach for the problem. It prevents the generator from overfitting and

provides diverse and realistic augmentations. Also, its parameters allow selecting the cost and quality trade-off.

CHAPTER 3

DATA AUGMENTATION BASED ON GENERATIVE NETWORKS

For the problem of visual object detection, context plays a crucial role in recognizing the extent of an object and usually provides a significant clue when determining the class of an object within an image. For example, in COCO[12] dataset, images predominantly containing sky are is more likely to contain instances from plane or bird classes. Information about the context of an instance, such as surrounding pixels belonging to the sky helps the model classify these instances more effectively. It has to be noted that this statement is only valid if the images in the training set have images obtained in their natural context.

There are several generative or non-generative methods that have been proposed to augment the data. From the generative methods, image inpainting networks and multiple discriminator based approaches are essentially focusing on proposing contextually plausible images by enforcing with different losses and different modules as we have discussed in Section 2.

Many recent works are focusing on producing highly diverse and high-resolution objects that belong to target classes using generative networks. However, in these methods, contextual consistency is not the key element, differently from the method proposed in this thesis. Most of the generative networks aim to generate realistic instances by a particular focus on an improvement in variety and quality of generated image samples. Based on the requirements of applications, GANs can be used to generate images with different output sizes. As proposed in [33], high-resolution images ranging between 256×256 and 1024×1024 can be synthesized using CelebAMask-

HQ dataset which consists of 30,000 high-resolution face images. However, having fewer data results in a trade-off between image quality and size of images to be generated. Increasing the size of images to be generated requires a large number of images in training to keep the quality of generated samples at similar levels. Moreover, even several methods have been proposed to improve the performance of GANs, the problem of unstable training with two or more networks together remains. As a result of this issue, lack of diversity, and insufficient realism of generated samples remain as the major problems. Even though there are several methods and tricks to improve training such as minibatch discrimination [60] and gradient penalty [61], it is not a guaranteed way to generate samples with semantically realistic and diversified.

Visual object detection task requires bounding box information which is including center coordinates, height, width, and class labels. Based on the stated requirements, an effective generative data augmentation method for object detection need to satisfy two key requirements: the generated objects are desired to be contextually plausible, and the location of the generated objects should be available to use in training. Depending on the generative algorithm, the background part of the generated instance can be either modified or unmodified. Hence, these selected patches which consist of generated sample and background, need to be seamlessly integrated into images. To perform this integration and augment the data by providing required annotations, we have proposed a new data augmentation framework. The proposed method consists of a generator network and a detector network. To provide contextual consistency, image inpainting networks have been employed for a large part of experiments. For image inpainting, we have selected Pluralistic Image Completion and DeepFill as generator networks. We have also used PSGAN as a generator network as it consists of multiple discriminators and Tiny YOLOv3 has been chosen as a detector network throughout the experiments. In the following three subsections, the selected networks and performance metrics for the experiments are explained.

3.0.1 Pluralistic Image Completion

Pluralistic Image Completion [62] is an algorithm for one-to-many image completion tasks. In image completion, there is usually only one ground truth training instance

(a) Masked input

(b) Output samples

Figure 3.1: Sample outcomes obtained by filling mask using Pluralistic

per label which results in generated samples to have limited diversity. To overcome this, Pluralistic uses two parallel paths, one is reconstructive and the other is generative, both supported by GANs. Sample results of the algorithm are shown in Fig. 3.1. First, the input image is partially masked around the center to create holes as shown in Fig. 3.1a. The algorithm generates diverse, realistic, and reasonable images with completed holes (Fig. 3.1b). Let us define the original image as I_g, the partially masked image as I_m, and the complement image as I_c. While the classical image completion methods attempt to reconstruct the ground truth image I_g in a deterministic fashion from I_m, Pluralistic aims to sample from $p(I_c|I_m)$. The reconstructive path combines information from I_m and I_c, which is used only for training. The generative path infers the conditional distribution of masked regions for sampling. Both of the paths follow Encoder-Decoder-Discriminator architecture. By using self-attention layer, it exploits short and long term context information and makes the generated masks more plausible with the patch rather than using pure GAN. In this work, we adopted Pluralistic network as the generator model to generate car instances on given backgrounds.

3.0.2 Pedestrian Synthesis GAN

Pedestrian Synthesis GAN [55] is an algorithm that is built on the GAN architecture with multiple discriminators. Original approach is based on augmenting pedestrian images. To reflect the PSGAN approach into our problem, the discriminator D_b is responsible for contextual plausibility and D_c is responsible for classification of the synthesized cars as real or fake. A general overview of the algorithm is provided in

Fig. 3.2.

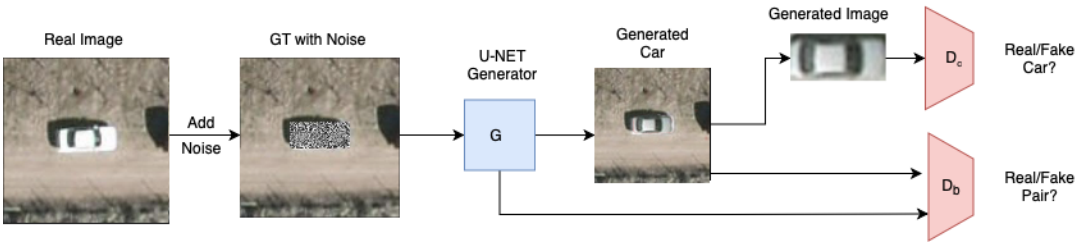


Figure 3.2: Schematic of PSGAN training

For generator G, U-Net [63] has been used. For D_c, a 5-layer convolutional network has been used with LeakyRelu and BatchNorm layers. To feed D_c with variable size inputs, SPP-layer has been used with a PatchGAN loss as in [36]. For learning background compatibility with instances using D_b network, modified DCGAN [53] is used with the following changes : 1) first convolutional layer made to accept 6-channel input which has stacked pair images; 2) PatchGAN [36] is used in discriminator, where D_b tries to classify N×N patches if they are real or fake ; 3) loss of LSGAN as in equation 3.1 is adopted.

$$\begin{aligned} L_{LSGAN}(G, D_b) \ &= E_{y \sim p_{gt.image}(y)}[(D_b(y) - 1)^2] + \\ &\quad E_{x,z \sim p_{noise.image}(x,z)}[(D_b(G(x, z)))^2] \end{aligned} \tag{3.1}$$

where x is the image with noise box and y is the ground truth image.
To generate more realistic car samples within the noise box z in the input image z, another adversarial procedure has been followed between G and D_c as in equation 3.2.

$$\begin{aligned} L_{GAN}(G, D_c) \ &= E_{y_c \sim p_{car}(y_c)}[log D_c(y_c)] + \\ &\quad E_{z \sim p_{noise}(z)}[log(1 - D_c(G(z)))] \end{aligned} \tag{3.2}$$

where z denotes the noise box in input image x and y_c is the cropped car image in the ground truth image y.

To control the differences between generated image and ground truth image, traditional l_1 loss has been applied as follows in equation 3.3:

$$L_{l1}(G) = E_{x,z\sim p_{noise.image}(x,z),y\sim p_{gt.image}(y)}[||y - G(x,z)||_1] \tag{3.3}$$

Together with l_1 loss, the combined loss becomes as in equation 3.4:

$$L(G, D_b, D_p) = L_{LSGAN}(G, D_b) + L_{GAN}(G, D_p) + \lambda L_{l1}(G) \tag{3.4}$$

where λ controls the importance of l_1 loss, and this parameter is set to 100 as it is selected as optimal number in the original paper.

3.0.3 Tiny YOLOv3

YOLO [64] is a single-stage object detection algorithm that uses a lightweight end-to-end network to predict bounding boxes and class probabilities directly from full images in a single pass. YOLOv1 introduced the concept of directly regressing object coordinates from the image instead of using region proposal networks such as Faster-RCNN [3]. Starting from YOLOv2, the system used anchor boxes for regressing object coordinates in a more accurate way, where these anchor boxes are calculated by clustering the size of instances in the training set. YOLOv3 is trained with a different class prediction loss formula and makes detection at three different scales.

We have selected and used Tiny YOLOv3 [6], which is a smaller version with a reduced number of layers and number of scales of predictions, for the experiments. This tiny model does two-scale prediction as opposed to three scales in the original YOLO. Since the aim of the proposed method is not to get the best detection performance among different models but to improve the base performance by augmentation without changing the architecture of the model, we selected Tiny YOLOv3 by considering its relatively low training/inference time.

3.0.4 Metrics

There are two commonly used metrics for evaluating the detection performance. Intersection over Union (IoU) is used for evaluating localization performance. It is the ratio of overlap (intersection) of predicted and ground-truth bounding boxes over the union of predicted and correct bounding boxes (Eq. 3.5) where B_g and B_p are the ground truth and predicted bounding boxes of the object.

$$IoU(B_g, B_p) = \frac{|B_g \cap B_p|}{|B_g \cup B_p|} \tag{3.5}$$

The result is between 0 and 1 indicating the ratio of correct prediction. If the prediction score is above the threshold, it is counted as a correct prediction and it increments the number of true positives by 1. If it is less than the predetermined threshold, then it will add 1 into the number of false positives.

Average Precision (AP) is the area under the precision-recall curve. It is commonly used to evaluate detection performance. Considering the correct predictions as true positives (TP), incorrect predictions as false positives (FP), and no predictions for an instance as false negatives (FN), we can formulate precision, recall, and AP as in Eq. 3.6:

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN} \qquad AP = \int_0^1 p(r)dr \tag{3.6}$$

3.1 Proposed Data Augmentation Method

The proposed method consists of 2 stages: training and augmentation. The training stage involves independent training of a generative network and a detector network. Separate training of these networks provides better stability during training. At the augmentation stage, the generative network is used to generate new samples and the detector is used to assess the feasibility of these samples for augmentation. An augmented training set is formed using the samples which are deemed feasible after this

assessment. This training set then can be used during the training of a detector to improve the detection performance.

3.1.1 Training Stage

The generator and detector networks are trained with image patches since the aim is to generate patches containing new instances and subsequently feed them into detector to evaluate their viability. Both these networks are fed with the same patches, which are extracted from the available training images. At the end of the training stage, the generator network learns to generate realistic target instances filling in the given patches. The detector is trained with bounding box annotations and the best model parameters which has the highest average precision is selected for the augmentation stage. If new class has been wanted to be included in data augmentation process, same procedure which is shown in 3.3 should be performed with that specific class instances.

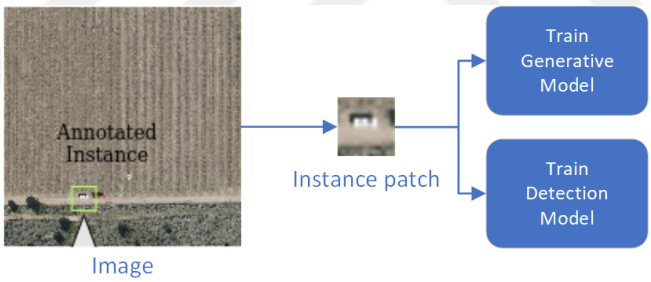


Figure 3.3: Schematic of training stage

3.1.2 Augmentation Stage

At this stage, the aim is to generate new object instances on the original images. After the training stage has been completed for both networks, patches with predefined sizes from random locations are extracted and their central areas (corresponding to half the patch dimensions) are masked with zeros. If the masked holes intersect with any other existing instances, the patch is discarded and a new image is used for the patch extraction while iterating over the train images. The patches are fed into the

trained generative model to generate synthetic object instances, which are expected to be located at the center of the patches. Then, the generated data is fed to the detector to evaluate whether the generated sample is acceptable to use for augmentation. For this evaluation purpose, the confidence score of the detector model, which reflects the confidence in identifying the generated instance, is used as a threshold. The generated sample is accepted if the confidence score is higher than the predetermined threshold. If the generated sample is not realistic or it has artifacts, it is expected to be assigned a low confidence score by the detector. If that is the case, the augmentation stage starts over with the next image from the training set since the current image may not be suitable for augmentation. If the generated sample is accepted by the detector, the original image is modified and augmented with the generated instance at the center of extracted patch coordinates. The augmented set is formed by adding a predetermined number of new instances into the raw train set. The schematic of the proposed framework is shown in Fig. 3.4.

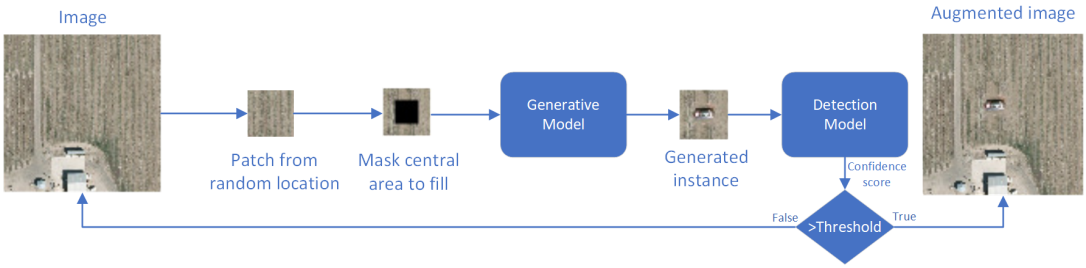


Figure 3.4: Schematic of augmentation stage

3.2 Experimental Evaluation

3.2.1 Dataset

We used Vehicle Detection in Aerial Imagery (VEDAI) [65] dataset which has 1272 RGB color images at 1024×1024 resolution. All images are annotated with bounding boxes with the following set of labels: *Car, Truck, Pickup, Tractor, Camping Car, Boat, Plane, Bus, Vans, Other*. We have selected *Car* instances for our experiments due to its variety in terms of color, orientation, and size.

There are a total of 1377 car instances in the dataset. Object instances larger than

48×48 pixels (less than 3% of all instances) are discarded as mentioned below. We divided the dataset as 500 and 772 for training and testing respectively. Only a part of the training set was used for experiments since we aim to improve the performance on small training datasets. 96×96 patches have been extracted around car instances from the training images which have a total of 490 car instances. The images have been down-scaled to the default input resolutions of Tiny YOLOv3, which is 416×416 for performance evaluation.

3.2.2 Patch Size Selection for Context Based Augmentation

We analyzed the instance sizes for car class in the dataset to select the appropriate patch size. The size histogram of all car instances in the dataset is shown in Fig. 3.5. As can be seen from this figure, 48×48 area covers more than 97% real instances with the best quality generated samples and we selected the instance size as 48×48 considering the best coverage and quality. Larger areas would cover all instances, but, in that case, most patches would have disproportionately large background area compared to the area of generated instances. In [62], it is reported that image completion works the best when the original image is double the size of the generated part. Hence, we selected the patch size as 96×96. The experiments show that generative models fail when the patch size is larger and generation time increases exponentially. Also, the boundary artifacts are more pronounced when patch sizes are smaller.

3.2.3 Detector Module

Tiny YOLOv3 has been adopted as the detector module. It has 13 convolutional layers and the first six layers are followed by max-pooling layers. The official implementation has been used with the pretrained weights obtained by training on ImageNet [66] and default parameters for 96x96 resolution. Based on our tests, pretrained weights provide faster convergence, more stable training, and better generalization. It also prevents overfitting due to single class training. The color based data augmentation configuration, including saturation, exposure, hue, and jitter augmentations are also enabled. We trained the model for 2000 epochs and observed that the training takes

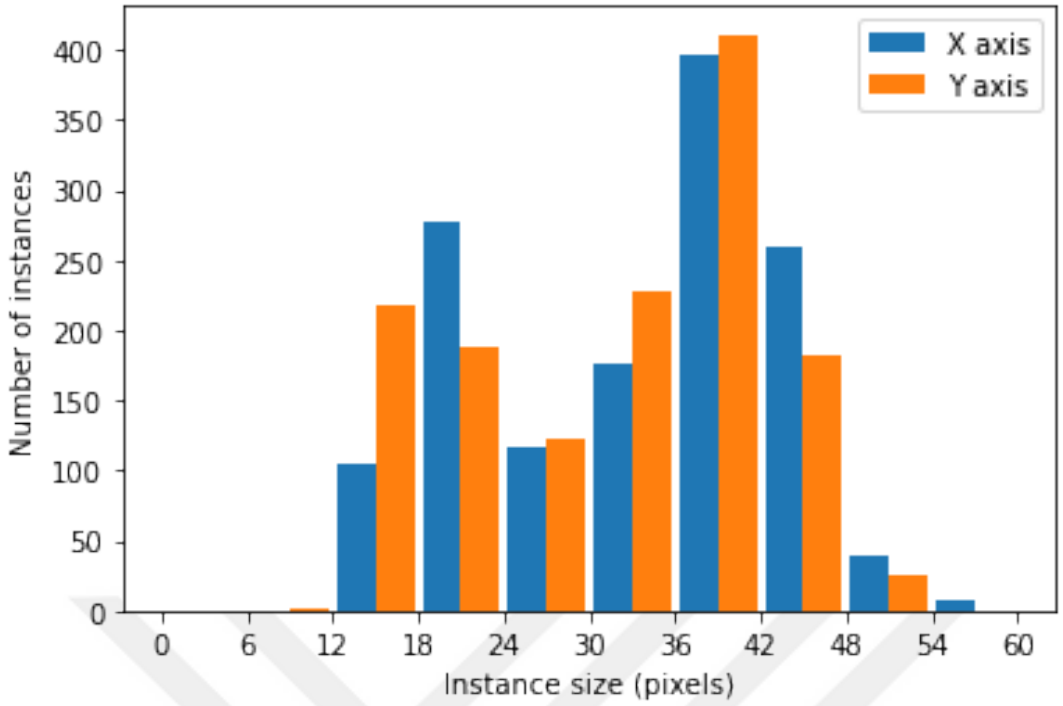


Figure 3.5: Distribution of size of bounding boxes for the car class

less than 1 hour on an NVIDIA GTX 1080 TI GPU for 96×96 patches. This training has been done separately for each experiment configuration using 200, 300, 400, and 500 images. These detectors have been kept the same throughout the experiments even the generators have changed. After the augmentation stage, for the final performance evaluation, the detector is trained with the default input size of the network implementation (416×416), which takes around 7 hours. The workflow of the proposed method is summarized in Algorithm 1.

3.2.4 Generator Module

As generators, original implementations of Pluralistic[1], Pedestrian-Synthesis-GAN(PSGAN)[2], and DeepFill[3] algorithms have been used. Pluralistic and DeepFill are image inpainting networks, whereas PSGAN is a contextual GAN with two discriminators, where they used additional background discriminator to assure contextual consistency.

[1] https://github.com/lyndonzheng/Pluralistic-Inpainting
[2] https://github.com/yueruchen/Pedestrian-Synthesis-GAN
[3] https://github.com/JiahuiYu/generative_inpainting

Algorithm 1: Workflow of proposed method

input : Generative Network g, Detection Network d

$w \leftarrow 96$ `// Determined patch size`

Training Stage:

for $j \leftarrow 1$ **to** *Number of instances* **do**
  $p_j \leftarrow$ Extract $w \times w$ instance patches
end

Train Generative Network g with extracted instance patches $p_{1...n}$

Train Detector Network d with extracted instance patches $p_{1...n}$

Augmentation Stage:

for $j \leftarrow 1$ **to** *Augmentation count* **do**
  Select an image to augment
  $p \leftarrow$Extract patch from random location
  Skip to another image if p intersects with another instance
  $p \leftarrow$ Mask the central $w/2 \times w/2$ area
  $p' \leftarrow$ Generate instance with Generative Network $g(p)$
  $o \leftarrow$ Evaluate generated instance with Detector Network $d(p')$
  if $o >$ *Acceptance threshold* **then**
    Augment the image with the generated instance
  end
end

3.2.5 Experimental Results

3.2.5.1 Using Pluralistic as Generator

Pluralistic uses Residual Blocks as its building elements. Each Residual Block consists of two convolutional layers and a residual connection with a convolutional layer. The Encoder has 5 Residual Blocks, Decoder and Discriminator have 5 and 6 Residual Blocks respectively with an attention layer in the middle. Training of this network takes around 3 hours for 200 epochs on an NVIDIA GTX 1080 TI GPU.

The model is able to generate realistic and diverse outputs without mode collapse. In order to examine the distribution of the generated samples, we generated 5000 samples sequentially and evaluated their score with the detector's confidence response. Some examples of generated samples are shown in Fig. 3.6 and the histogram of confidence scores can be seen in Fig. 3.7. As it can be seen in Fig. 3.6, the quality of the samples increases with the increasing confidence score threshold. An important observation is that the samples which are generated in an improper context, such as samples placed at the sea, get quite low confidence scores which, confirming the effect of context in detector evaluation. The equal-ranged histogram bins exhibit no large difference, indicating a good diversity over generated samples. Also, it shows that the generator does not overfit to detector vulnerabilities.

We first conducted an experiment to determine the detector threshold for the networks by augmenting 500 images with 1000 generated instances. The result of this experiment can be seen in Table 3.1 for different confidence thresholds and IoU values. The confidence threshold decides if the generated sample is good enough to augment. When there is no threshold (i.e., threshold set to 0), the result is the worst as expected since all generated samples are accepted to be used in augmentation regardless of their quality and their contextual fitness level. In this case, the process is completed in 1000 iterations as there are no rejected samples. Considering the average of different(0.2,0.5,0.7) IoU performances, a threshold of 0.9 gives the best results. At this threshold, it takes 5902 iterations to generate 1000 accepted instances, i.e. 1 sample is accepted from every $\sim$5.9 generated samples. It takes significantly longer time and the diversity of the accepted samples reduces above this threshold. As it can be

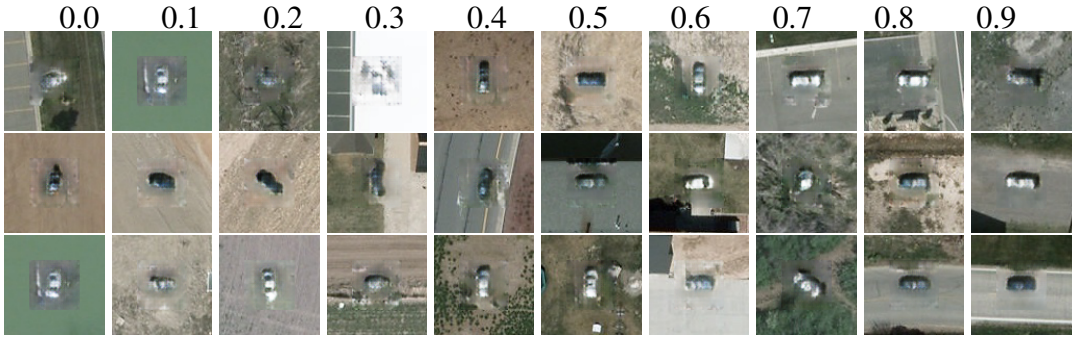


Figure 3.6: Some Samples generated with Pluralistic with their confidence scores.

Table 3.1: Average Precision results for different confidence thresholds and IoU values where 500 images are augmented with 1000 instances using Pluralistic. Augmentation iterations indicate the number of trials to reach 1000 accepted instances.

Acceptance threshold	IoU 0.2	0.5	0.7	Average	Augmentation iterations
0.0	56.20	36.11	7.26	33.19	1000
0.1	55.55	40.67	7.97	34.73	1172
0.2	55.42	39.51	9.23	34.72	1288
0.3	56.23	39.65	8.76	34.88	1423
0.4	56.38	44.16	10.31	**36.95**	1709
0.5	56.04	44.05	10.08	36.72	1818
0.6	56.16	42.29	10.81	36.42	2189
0.7	57.63	43.41	9.57	36.87	2667
0.8	57.64	43.15	9.90	36.90	3498
0.9	57.08	43.81	10.57	**37.15**	5902

seen in Fig. 3.9, when the threshold value gets lower, the size histogram of generated instances becomes more uniform and when the threshold value gets higher, the histogram becomes more impulsive. When threshold value sets into higher values, the augmented dataset does not contribute to the robustness and the generalization performance of the detector and may cause overfitting because of the lack of diversity. The second best threshold 0.4 provides a good balance between average precision and the number of iterations. It also has the best precision for the default IoU value (0.5). Considering the significantly less execution time and the diversity of the accepted samples, 0.4 has been selected as the detector threshold. As it can be seen from the Table 3.1, when the IoU threshold is set to value 0.7, percentage increase in the AP value gets significantly higher compared to other IoU thresholds.

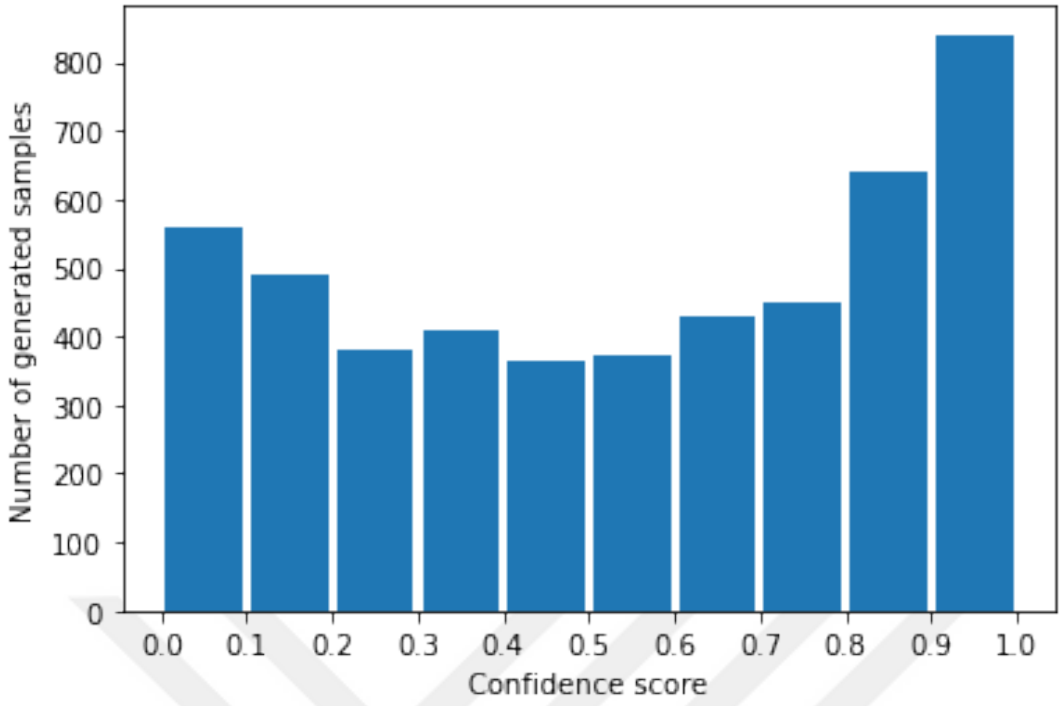


Figure 3.7: Histogram of confidence scores for 5000 samples generated with Pluralistic

There are two main factors for the evaluation of augmentation performance: total number of training images and number of synthetic car instances embedded into these training images. We have tested the system with a varying number of training images to observe its effects on the performance. The number of training images has been selected to be 200, 300, and 400, where they contain 251, 334, and 424 car instances respectively. The number of training instances is increased by adding 1, 2, and 3 new instances per image by augmentation, i.e. when there are 400 training images, we added 400, 800, and 1200 synthetic object instances (cars) to approximately increase the existing number of instances by a factor of $2\times$, $3\times$, and $4\times$. Note that all of the synthetic object instances are placed on the original images, keeping the number of images the same. As per the above example, there are still 400 images but they have higher number of object instances than they originally have.

The results shown in Table 3.2 reveals that the performance of the detector network improves with the generative augmentation in all cases compared to the baseline (i.e. when there is no generative augmentation). Average Precision at IoU > 0.5 perfor-

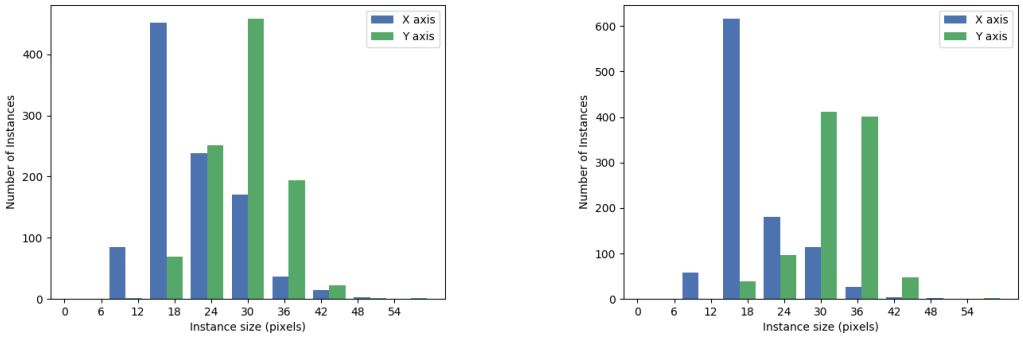
Table 3.2: Detection performances (AP) when Pluralistic is used as the generator model.

Dataset Images	Augmented Instances	IoU 0.2	IoU 0.5
200	-	49.85	31.43
	200	52.15	37.65
	400	51.71	37.04
	600	52.27	39.14
300	-	53.76	33.25
	300	55.39	39.76
	600	56.01	41.62
	900	56.60	40.61
400	-	55.46	36.14
	400	56.03	40.24
	800	56.02	42.18
	1200	56.87	43.83

mance increases by 24.5%, 22.1%, and 21.2% respectively when there are 200, 300, and 400 images in the dataset and they are augmented with up to 3 new synthetic instances per image. Examples from the augmented dataset can be found in Fig. 3.8 where it can be seen that the augmented instances are realistic, diverse, and coherent with the rest of the image. Both visual and quantitative results demonstrate that the proposed method can be used as an augmentation strategy to improve performance especially when there is limited number of training images.

As it can be seen in Fig. 3.10, the precision-recall curve for each augmentation setting has been drawn for 300 images. When the IoU threshold has been set as 0.2, while the recall value gets higher, the precision values are getting closer to each other. This means that all the models behave similarly when the confidence threshold gets lower. Another observation is the plots are more distinctive when the IoU threshold has been set as 0.5, which indicates that the proposed augmentation method actually improves localization performance.

Figure 3.8: Raw training images (left) are augmented with 2 separate car instances (middle). Augmented samples are highlighted with red arrows and their zoomed versions are shown on the right.



(a) Distribution of sizes when threshold = 0.1

(b) Distribution of sizes when threshold = 0.9

Figure 3.9: Comparison of size for different threshold values

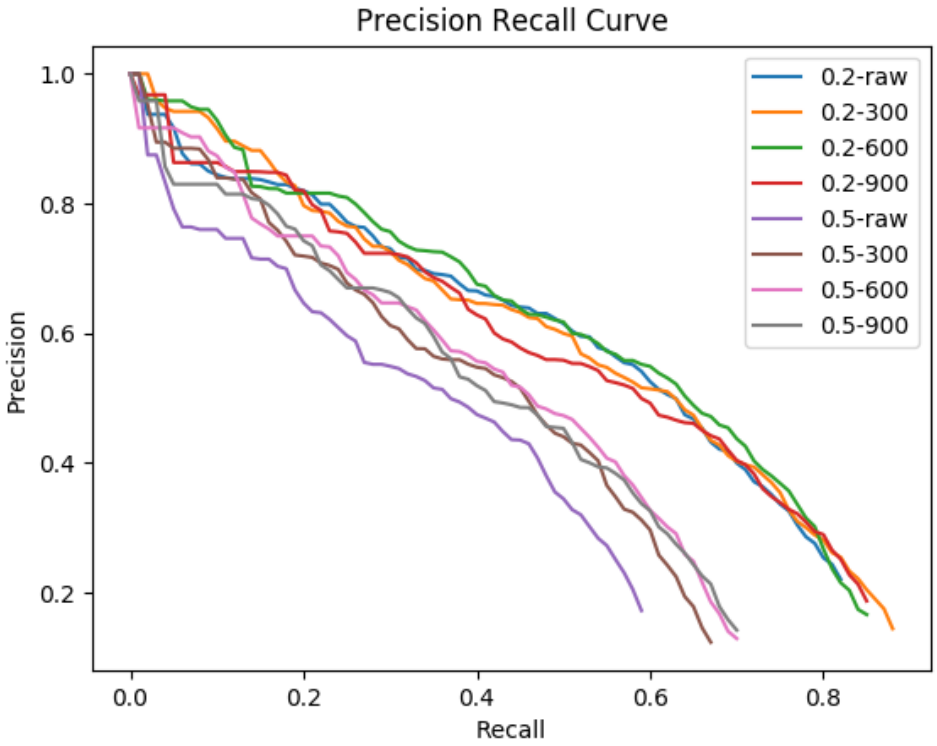


Figure 3.10: Precision-Recall curve for Pluralistic

3.2.5.2 Using PSGAN as Generator

We have also used PSGAN for our experiments where the network has one generator responsible for generating instances from the selected noisy area in the center area of the selected patch and two discriminators that are responsible for the realism of generated instances and the contextual plausibility with the background. The architecture which is described in Section 3.0.2 is used with the default parameters. The noise region has been created randomly by assigning values between 12 and 48 within 96×96 patches. Training of this network takes around 2 hours for 200 epochs on an NVIDIA GTX 1080 TI GPU.

Some examples of generated samples are shown in Fig. 3.11 and the histogram of confidence scores can be seen in Fig. 3.12. As we have observed from those generated samples, the diversity has been preserved even the confidence distribution is weighted towards higher values with respect to Pluralistic. As it can be seen in Fig. 3.11, a similar case is observed with Pluralistic results is directly applicable in this scenario also, regarding the quality of generated samples. It can be also observed that the context of the images has a significant role in assigning confidence scores. For example, even though the visual quality of the topmost car sample of 0.1 confidence column is satisfactory, a low confidence score is assigned to it due to low contextual plausibility since the instance is placed at the sea. As seen in Fig. 3.12, the histogram of confidence scores is mostly accumulated at the threshold value of 0.9 and higher. However, as it can be seen from Fig. 3.11 and Fig. 3.14c, the samples are generated with different angles and sizes as well as different pixel intensity averages. Hence, it can be interpreted as PSGAN is generating highly confident detection samples without being stuck with similar samples due to training problems.

We first used the same experimental setup in the previous section to determine the optimal detector threshold for the augmentation by augmenting 500 images with 1000 generated instances. The result of this experiment can be seen in Table 3.3 for different confidence thresholds and different IoU values. Considering the average of different(0.2,0.5,0.7) IoU performances, a threshold of 0.8 gives the best results. At this threshold, it takes 1852 iterations to generate 1000 accepted instances. Since the number of iterations required to generate sufficient acceptable samples does not

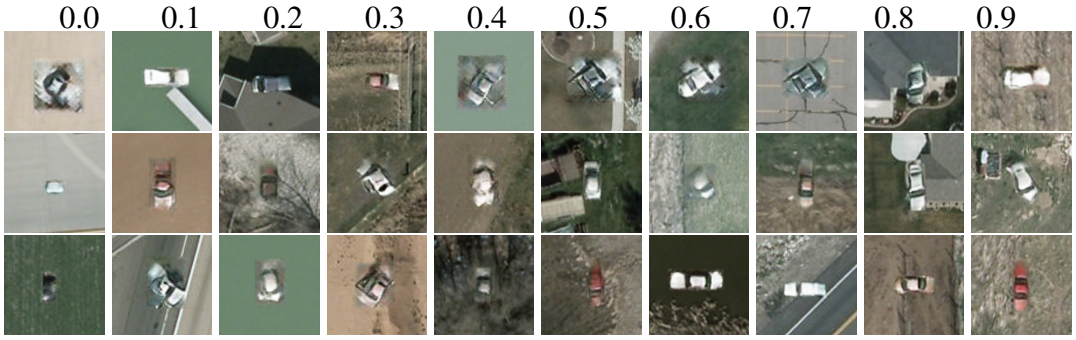


Figure 3.11: Some samples generated with PSGAN with their confidence scores.

get significantly lower below this threshold, 0.8 has been selected for the remainder of experiments. The same procedure with Pluralistic has been followed for the experiments and the number of training images has been selected to be 200, 300, and 400, where they contain 251, 334, and 424 car instances respectively. The number of training instances is increased by adding 1, 2, and 3 new instances per image by augmentation, i.e. when there are 400 training images, we added 400, 800, and 1200 synthetic object instances (cars) to approximately increase the existing number of instances by a factor of $2\times$, $3\times$, and $4\times$.

The results shown in Table 3.4 reveals that the performance of the detector network improves with the augmentation provided with synthetic samples generated by PSGAN in all cases compared to the baseline (i.e. when there is no generative augmentation). Average Precision at IoU > 0.5 performance increases by 32.2%, 32.7%, and 23.1% respectively when there are 200, 300, and 400 images in the dataset and they are augmented with up to 3 new synthetic instances per image.

As it can be seen in Fig. 3.13, the precision-recall curve for each augmentation setting has been drawn for 300 images. The improvement in detection performance is visible when the IoU threshold is set as value of 0.5.

3.2.5.3 Size and Aspect Ratio Comparison of Generated Instances

We analyzed the statistics of generated instances using Pluralistic and PSGAN. Based on the size distribution of instances which can be seen in Fig. 3.14, generated in-

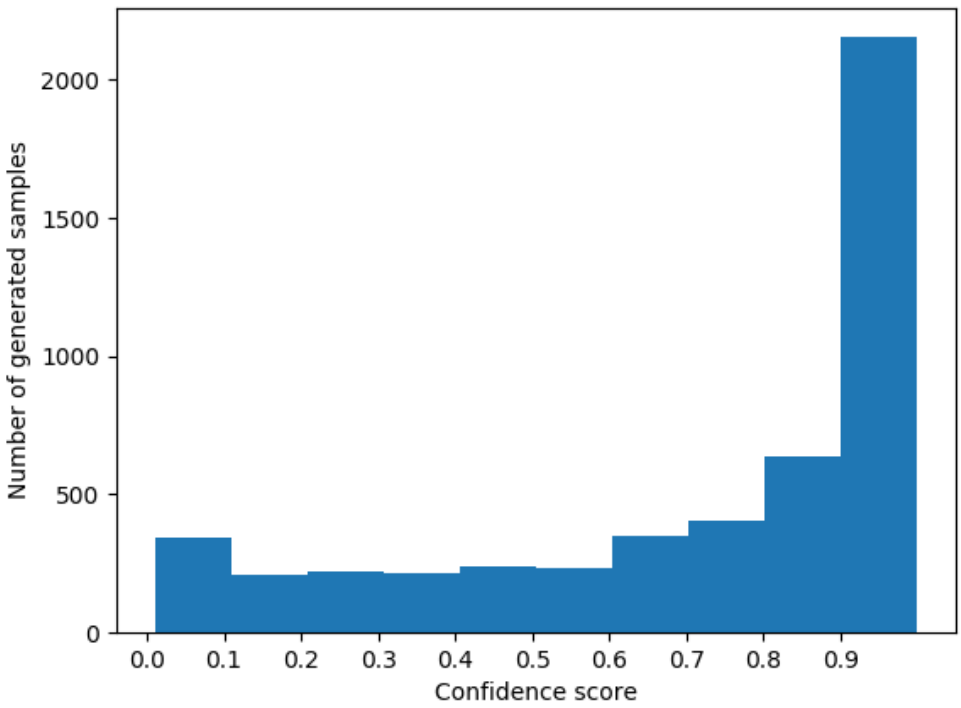


Figure 3.12: Histogram of confidence scores for 5000 samples generated with PS-GAN

Table 3.3: Average Precision results for different confidence thresholds and IoU values where 500 images are augmented with 1000 instances using PSGAN. Augmentation iterations indicate the number of trials to reach 1000 accepted instances.

Acceptance threshold	IoU 0.2	IoU 0.5	IoU 0.7	Average	Augmentation iterations
0.0	57.16	45.15	9.84	37.38	1000
0.1	57.62	47.60	9.89	38.37	1072
0.2	59.27	45.41	9.93	38.20	1142
0.3	59.14	46.49	9.97	38.53	1190
0.4	57.81	46.57	9.89	38.09	1274
0.5	58.76	45.69	9.65	38.03	1333
0.6	58.11	45.25	9.96	37.77	1451
0.7	59.15	45.33	9.83	38.10	1538
0.8	60.36	47.64	9.94	**39.31**	1852
0.9	60.46	46.65	9.77	38.96	2278

Table 3.4: Detection performances (AP) when PSGAN is used as the generator model.

Dataset Images	Augmented Instances	IoU 0.2	IoU 0.5
200	-	49.85	31.43
	200	55.11	40.10
	400	54.45	41.56
	600	54.53	38.73
300	-	53.76	33.25
	300	57.60	43.20
	600	57.36	44.13
	900	51.81	41.95
400	-	55.46	36.14
	400	57.68	43.82
	800	58.60	44.52
	1200	52.83	42.91

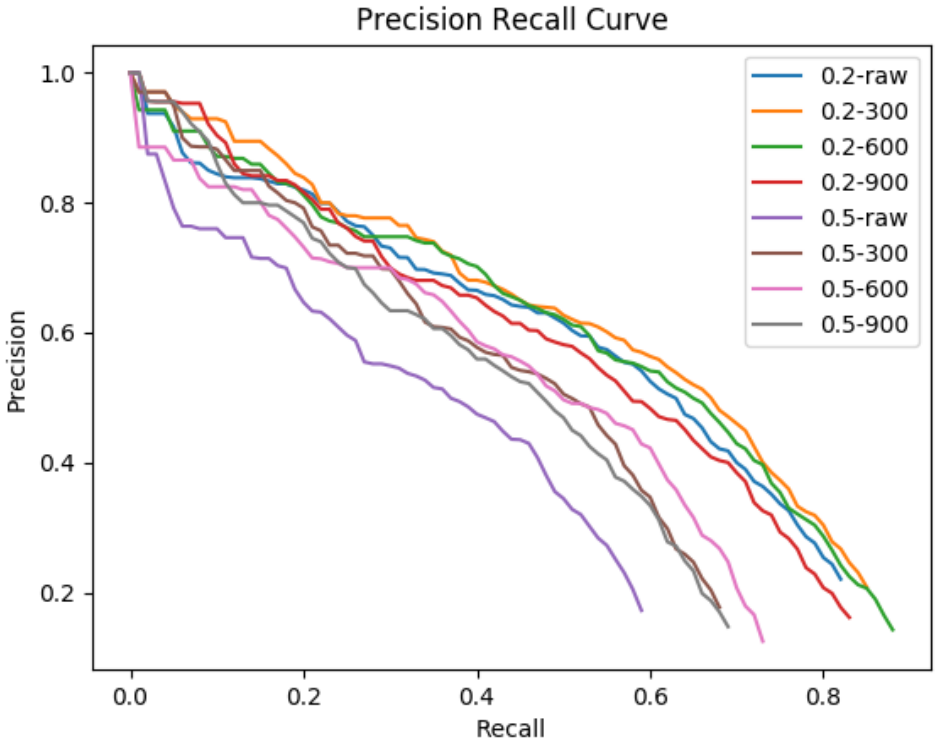
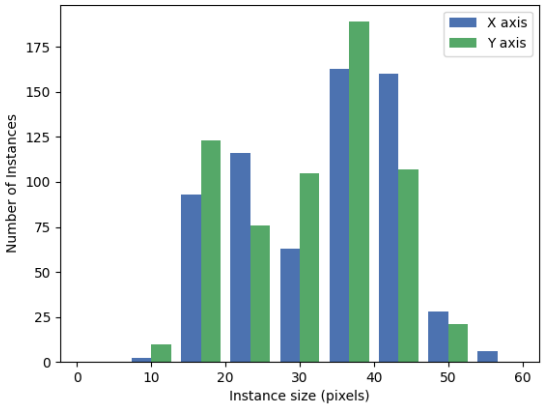


Figure 3.13: Precision-Recall curve for PSGAN

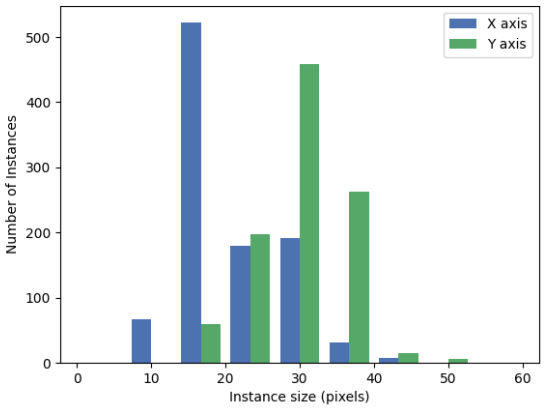
stances with PSGAN as in Fig. 3.14c seems to have more variety than instances which are generated with Pluralistic as shown in Fig. 3.14b, which positively affects the augmentation results. PSGAN generates varying sized instances as it assigns random noise into the background generates car instances within this noise region. Fig. 3.15 shows the distribution of aspect ratios. As can be seen in this figure, the size distribution of generated samples using PSGAN has more variety than the samples generated with Pluralistic.

3.2.5.4 Using DeepFill as Generator

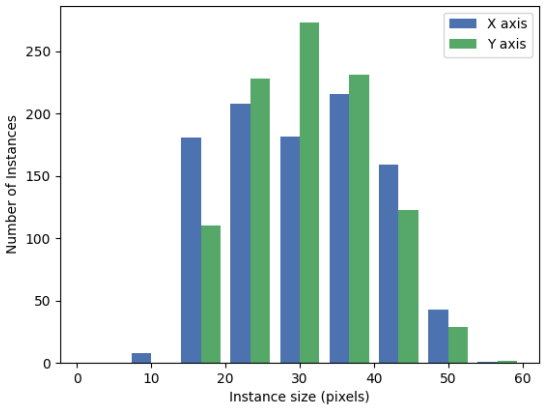
We have conducted additional experiments with a different image inpainting network, DeepFill [42], to validate that the proposed method can also be used with different generator models. The model is a feed-forward, fully convolutional neural network which can process images with multiple holes at arbitrary locations and with variable sizes. It has 2 autoencoders, the first one is for coarse inpainting and the following one is for refining the coarse generation. Two discriminators are used for local and global evaluation. It uses contextual attention and gated convolutions. The official DeepFill implementation accompanying the paper has been used for experiments. The proposed workflow has been integrated with DeepFill as generator instead of Pluralistic. Some examples of generated samples are shown in Fig. 3.16 and the results can be seen in Table 3.5 for 0.9 detector acceptance threshold. Average Precision at IoU > 0.5 increases by 25.1%, 25.7% and 25.6% respectively when there are 200, 300 and 400 images in the dataset and they are augmented with 2 new instances per image. It can be seen that the proposed method results in performance improvement for the detector model with similar gains to the variant with the Pluralistic, showing that the method can be used with different generative models.



(a) Train data

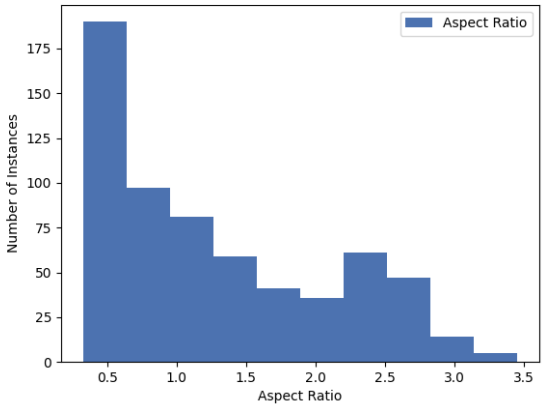


(b) Pluralistic

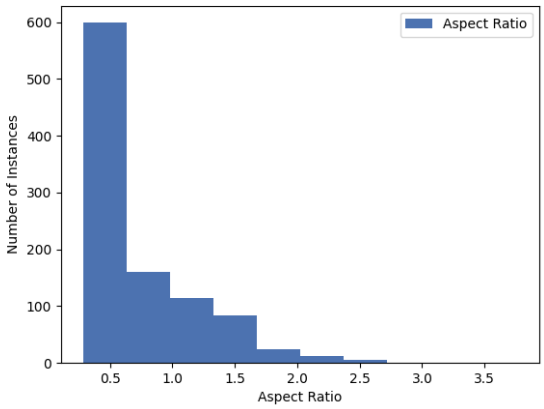


(c) PSGAN

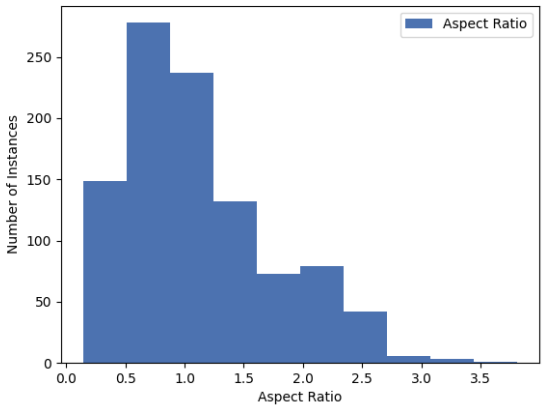
Figure 3.14: Size distributions in the raw and synthetic data



(a) Train data



(b) Pluralistic



(c) PSGAN

Figure 3.15: Aspect ratio distributions in the raw and synthetic data

Table 3.5: Detection performances (AP) when DeepFill is used as the generator model.

Dataset Images	Augmented Instances	DeepFill
200	-	31.43
	200	35.96
	400	39.31
	600	36.11
300	-	33.25
	300	38.93
	600	41.81
	900	41.96
400	-	36.14
	400	38.73
	800	45.40
	1200	42.19

Figure 3.16: Examples of generated samples with DeepFill.

CHAPTER 4

CONCLUSION

In this thesis, a new data augmentation framework has been proposed for object detection. The proposed framework consists of a separate generator and detector where the generator is responsible for generating high-quality instances by giving attention to context. The detector is responsible for the evaluation of generated samples and localizing these samples. We have shown that the proposed data augmentation framework is generic, and it works well in association with different generator models. The proposed framework is also flexible in terms of trade-offs in training and augmentation stages, such as the number of iterations vs. confidence threshold. Modular changes in generator will affect the timing and accuracy of the proposed method as well.

Experimental results have shown that object detection performance can be improved using the proposed generative data augmentation framework. Integrating a detector module with a predefined threshold is an essential part of this improvement. This threshold should be chosen carefully as it both affects the computational cost and accuracy of the proposed method. There is no linear relationship between the threshold value and accuracy improvement, but we have observed a trend that tends to make the algorithm more accurate with higher threshold values. Based on the accuracy comparison between the augmented model and baseline model, we have observed that the proposed method mainly improves the localization performance of vehicle instances more than recovering true negative samples.

As a future work, different classes other than vehicles and different object sizes can be investigated. The generator and detector modules can be replaced with other algo-

rithms to produce higher-quality and more diverse samples. A new benchmark based on improvement in detection performance can be created for context-based instance generation problem by trying different generator modules.

REFERENCES

[1] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[2] Y. Zhu, H. Cai, S. Zhang, C. Wang, and Y. Xiong, "Tinaface: Strong but simple baseline for face detection," *ArXiv*, vol. abs/2011.13183, 2020.

[3] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.

[4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, pp. 1–1, 07 2018.

[5] R. B. Girshick, "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[6] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018.

[7] S. Jha, A. Dey, R. Kumar, and V. Solanki, "A novel approach on visual question answering by parameter prediction using faster region based convolutional neural network," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. InPress, p. 1, 01 2018.

[8] J. Gu, G. Wang, J. Cai, and T. Chen, "An empirical study of language cnn for image captioning," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1231–1240, 2017.

[9] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[10] Y. Huang, Q. Wang, W. Jia, and X. He, "See more than once - kernel-sharing atrous convolution for semantic segmentation," *ArXiv*, vol. abs/1908.09443, 2019.

[11] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[12] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[13] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8419–8428, 2019.

[14] A. Bansal, K. Sikka, G. Sharma, R. Chellappa, and A. Divakaran, "Zero-shot object detection," in *ECCV*, 2018.

[15] Y. Niitani, T. Akiba, T. Kerola, T. Ogawa, S. Sano, and S. Suzuki, "Sampling techniques for large-scale object detection from sparsely annotated objects," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6503–6511, 2019.

[16] G. Cao, X. Xie, W. Yang, Q. Liao, G. Shi, and J. Wu, "Feature-fused ssd: fast detection for small objects," in *International Conference on Graphic and Image Processing*, 2018.

[17] J.-S. Lim, M. Astrid, H. Yoon, and S. Lee, "Small object detection using context and attention," *ArXiv*, vol. abs/1912.06319, 2019.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84 – 90, 2017.

[19] J. Guérin, O. Gibaru, S. Thiery, and E. Nyiri, "Cnn features are also great at unsupervised classification," *ArXiv*, vol. abs/1707.01700, 2017.

[20] H. Kumdakcı, C. Öngün, and A. Temizel, "Generative data augmentation for vehicle detection in aerial images," in *Pattern Recognition. ICPR International Workshops and Challenges*. Cham: Springer International Publishing, 2021, pp. 19–31.

[21] L. Taylor and G. Nitschke, "Improving deep learning with generic data augmentation," *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1542–1547, 2018.

[22] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *AAAI*, 2020.

[23] Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh, "How useful is photo-realistic rendering for visual learning?" *ArXiv*, vol. abs/1603.08152, 2016.

[24] N. Mayer, E. Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy, and T. Brox, "What makes good synthetic training data for learning disparity and optical flow estimation?" *International Journal of Computer Vision*, vol. 126, pp. 942–960, 2018.

[25] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "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)*, pp. 1082–10 828, 2018.

[26] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtualworlds as proxy for multi-object tracking analysis," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4340–4349, 2016.

[27] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012.

[28] N. Dvornik, J. Mairal, and C. Schmid, "On the importance of visual context for data augmentation in scene understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[29] ——, "Modeling visual contesxt is key to augmenting object detection datasets," in *IEEE Europe Conference on Computer Vision (ECCV)*, 2018.

[30] T. Devries and G. W. Taylor, "Dataset augmentation in feature space," *ArXiv*, vol. abs/1702.05538, 2017.

[31] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[32] C. Ongun and A. Temizel, "Paired 3d model generation with conditional generative adversarial networks," *ECCV*, vol. abs/1808.03082, 2018.

[33] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *ArXiv*, vol. abs/1710.10196, 2018.

[34] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *ArXiv*, vol. abs/1809.11096, 2019.

[35] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, 2017.

[36] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, 2017.

[37] Y. Choi, M.-J. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8789–8797, 2018.

[38] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan, "Perceptual generative adversarial networks for small object detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1951–1959, 2017.

[39] C. D. Prakash and L. Karam, "It gan do better: Gan-based detection of objects on images with varying quality," *ArXiv*, vol. abs/1912.01707, 2019.

[40] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 105–114, 2017.

[41] S. López-Tapia, A. Lucas, R. Molina, and A. Katsaggelos, "A single video super-resolution gan for multiple downsampling operators based on pseudo-inverse image formation models," *Digit. Signal Process.*, vol. 104, p. 102801, 2020.

[42] J. Yu, Z. L. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, "Generative image inpainting with contextual attention," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5505–5514, 2018.

[43] Y. Kwon and M. Park, "Predicting future frames using retrospective cycle gan," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1811–1820, 2019.

[44] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *NIPS*, 2016.

[45] B.-C. Chen and A. Kae, "Toward realistic image compositing with adversarial learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[46] H. Wu, S. Zheng, J. Zhang, and K. Huang, "Gp-gan: Towards realistic high-resolution image blending," *Proceedings of the 27th ACM International Conference on Multimedia*, 2019.

[47] Z. Huang, S. Chen, J. Zhang, and H. Shan, "Pfa-gan: Progressive face aging with generative adversarial network," 2020.

[48] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *ICML*, 2016.

[49] K. Hamada, K. Tachibana, T. Li, H. Honda, and Y. Uchida, "Full-body high-resolution anime generation with progressive structure-conditional generative adversarial networks," in *ECCV Workshops*, 2018.

[50] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "How to train your dragan," *ArXiv*, vol. abs/1705.07215, 2017.

[51] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *ArXiv*, vol. abs/1701.07875, 2017.

[52] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *ArXiv*, vol. abs/1411.1784, 2014.

[53] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[54] D. Lee, S. Liu, J. Gu, M.-Y. Liu, M.-H. Yang, and J. Kautz, "Context-aware synthesis and placement of object instances," in *NeurIPS*, 2018.

[55] X. Ouyang, Y. Cheng, Y. Jiang, C. Li, and P. Zhou, "Pedestrian-synthesis-gan: Generating pedestrian data in real scene and beyond," *ArXiv*, vol. abs/1804.02047, 2018.

[56] K. Zheng, M. Wei, G. Sun, B. Anas, and Y. Li, "Using vehicle synthesis generative adversarial networks to improve vehicle detection in remote sensing images," *ISPRS International Journal of Geo-Information*, vol. 8, no. 9, p. 390, 2019.

[57] L. Liu, M. Muelly, J. Deng, T. Pfister, and L.-J. Li, "Generative modeling for small-data object detection," *IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019.

[58] C. Han, K. Murao, T. Noguchi, Y. Kawata, F. Uchiyama, L. Rundo, H. Nakayama, and S. Satoh, "Learning more with less," *28th ACM International Conference on Information and Knowledge Management*, 2019.

[59] S. Milz, T. Rüdiger, and S. Süss, "Aerial ganeration: Towards realistic data augmentation using conditional gans," in *ECCV Workshops*, 2018.

[60] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *NIPS*, 2016.

[61] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *NIPS*, 2017.

[62] C. Zheng, T. Cham, and J. Cai, "Pluralistic image completion," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1438–1447, 2019.

[63] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *ArXiv*, vol. abs/1505.04597, 2015.

[64] J. Redmon, S. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[65] S. Razakarivony and F. Jurie, "Vehicle detection in aerial imagery : A small target detection benchmark," *J. Vis. Commun. Image Represent.*, vol. 34, pp. 187–203, 2016.

[66] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.