令和 4 年度　　茨城大学大学院理工学研究科情報工学専攻

修士学位論文

領域適応手法を利用した悪天時の自動車の物体検出

所属　　情報工学専攻

著者　　内間けんじ (21NM710X)

指導教員　　新納浩幸教授

令和 4 年 2 月 1 日 ()

令和 4 年度　茨城大学大学院理工学研究科情報工学専攻　修士学位論文

領域適応手法を利用した悪天時の自動車の物体検出

著者

内間けんじ (21NM710X)

指導教員

新納浩幸教授

論文要旨

近年物体検出の技術は様々な分野で利用されている。例として、自動運転では運転中での自動車や人の検知・認識を画像映像からの物体検出にて行うこと等が挙げられる。このような物体検出を行うためには物体検出器の学習を行う必要があり、そのためにデータセット用意する必要がある。先の例の場合では運転中の前方車載画像のデータセット等を用意することとなる。そして用意したデータセットを用いて学習した物体検出器を使用することで検出したい対象を認知、位置の予測をすることができるようになる。しかしこのような物体検出の学習で用いられるデータセットは大量に必要となる。目的の対象領域に必要なデータセットがない場合は新たに作成する必要があるが、データセットを自作する際にはアノテーションの作業等で大きなコストがかかる。

本稿では、走行車両の前方車載画像における自動車の物体検出において、アノテーションされた好天候時のデータセットはあるが、アノテーションされた悪天候時のデータセットがない場合を想定し実験の設定を行った。この設定の場合では悪天候時領域での物体検出を行う際、本来ならば訓練データとして対象領域である悪天候時データセットを用意する必要があるが先述の通りそのコストのために困難である。　そこで本実験では入力画像から対象領域の特徴を保持する偽画像を生成する CycleGAN を用いて目的の偽のデータセットを作成し、アノテーションのコストを省いたデータ拡張を行う。

CycleGAN によりアノテーションされた好天時領域の画像から悪天候時領域の偽画像を生成し、好天候時データセットのアノテーションを偽の悪天候時データセットに対応させて学習を行い、「悪天候時と好天候時の物体検出」と「悪天候時のみの物体検出」を想定したテストデータで評価値を測定しデータ拡張により精度が向上することを確かめる。

2023 Major in Computer and Information Sciences,
Granduate School of Science and Engineering, Ibaraki University

DetectingAutomobileObjectsinBadWeatherUsingDomainAdaption

Auther

Kenji Uchima (21NM710X)

Adviser

Hiroyuki Shinnou

Abstract

In recent years, object detection technology has been used in various fields. One example is the detection and recognition of vehicles and people during driving in automatic driving by object detection from image video. In order to perform such object detection, it is necessary to prepare a dataset. In the case of the previous example, it is necessary to prepare a data set of forward in-vehicle images during driving. Then, by using an object detector trained with the prepared dataset, the system can recognize the object to be detected and predict its position. However, a large number of datasets are required for this type of object detection training. If the necessary dataset is not available for the target region of interest, it must be newly created at great cost.

In this paper, we set up a case where there is an annotated good weather dataset but no bad weather one for car object detection in the forward onboard image of a moving vehicle. In this setup, when performing object detection in the bad weather region, it is necessary to prepare a bad weather dataset as training data, but as mentioned earlier, this is difficult due to the cost.

Therefore, in this experiment, a fake dataset is created using CycleGAN, which generates a fake image from the input image that retains the features of the target region, and data expansion is performed.

We generate false images of bad weather regions from images of good weather regions annotated by CycleGAN, train the annotations of the good weather dataset to correspond to the false bad weather dataset, and measure the evaluation values on test data assuming these regions to confirm that the accuracy is improved by data expansion. We will confirm that the accuracy improves with data expansion.

目次

第 1 章

序論

ある画像について物体検出を行うためには、あらかじめある画像と同じ領域の特徴を保持する大量のアノテーションされた画像データセットを用意して物体検出モデルの学習を行う必要がある。アノテーションデータとは、画像のどこに何が写っているかの記録を保持しているデータを指す。

物体検出を行う目標に対して利用可能なデータセットが存在する場合はそのまま学習を行えば良い。しかし目標に対するデータセットが存在しない場合は初めから自作する必要がある。そういった場合で大量のアノテーションデータを手作業で用意するのには多くの時間と労力が要求され大変なコストがかかることになる。例として本研究では悪天候時の自動車の物体検出において悪天候時の画像データセットをあらかじめ用意することができない場面を想定している。

このような状況でも自動車の物体検出を行うために、CycleGAN によってアノテーションされた好天時のデータから悪天候時の偽画像データを生成し、そのアノテーションデータを悪天候時の画像データセットに対して再利用することで自動車の物体検出モデルの学習を行い物体検出を行う。物体検出を行う場面の想定として「悪天候時と好天候時の物体検出」と「悪天候時のみの物体検出」の二つの場面に対する評価を行うことでデータ拡張による効果を確認する。

第 2 章

物体検出

物体検出とは入力された画像に対して、物体がどのような位置に存在しているかを、それがどのような物体であるかを推定するための手法である。物体検出を行うためには検出器のモデルを学習させる必要があり、それにはアノテーションされた大量のデータセットが必要になる。データセットは画像データとそれに対する物体の位置・種類の情報を含むアノテーションデータからなる。物体の位置はバウンディングボックスによって区別されている。アノテーションデータには物体の周りを囲む箱の座標が記されており、たいていの場合は四角い領域の対角の 2 点座標、あるいは領域中心の座標と領域の縦・横の長さで構成される。

2.1 CNN

CNN(Convolutional Neural Network) とは主に画像の分野においてよく使われるネットワークである。畳み込みニューラルネットワークとも呼ばれ、畳み込み層とプーリング層が交互に重なる構造をしている。通常のニューラルネットワークとして良く用いられる全結合層では入力データを単純に並べてから学習を行うが、それでは全ての数値が一次元的に配置されてしまうため同じように捉えられてしまう。それに対して CNN ではデータを入力する際に三次元的な情報が保持されながら学習は行われるため、画像データの位置情報が失われることがない。

畳み込み層では入力された画像にたいしてあらかじめ設定されたフィルターに従って画像の変換を行う。フィルターは入力画像よりもサイズが小さく、それらを入力画像に対しスライドするように移動する。その際フィルターが重なった場所で値を掛け合わせ

て合算した値を出力した二次元の特徴ベクトルを抽出する。

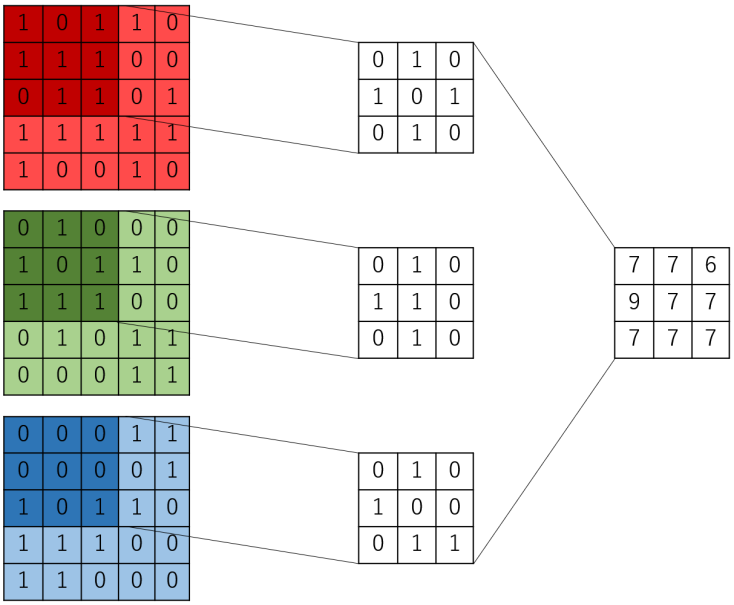

図 2.1: 畳み込み層とフィルタの例

図 3..1 の例では入力 (5,5,3) に対して (3,3) のフィルターがチャンネルずつに用意されている。入力に対してフィルターを枡掛けしていくと、始めは 3+3+1=7 の値が得られることがわかる。その後特徴ベクトルに 7 の値を記録してフィルターは一つとなりにずれて再度升掛けの演算を行っていく。最終的には (3,3) の特徴ベクトルが得られていることが分かる。

これらの操作に加えて畳み込み演算を行う際に別のパラメータを設定することがある。

畳み込み層でフィルターを入力した画像よりはみ出るように使用したい場合、入力画像の外周を別の値で埋めることをパディングという。特に入力画像の外周を 0 で囲むように拡張することをゼロパディングといい、これによって入力画像と同じサイズの出力画像を得ることができる。

入力画像上でフィルターをスライドさせる場合は通常 1 マスずつずれるように移動する。しかしあらかじめストライドの値を 1 より大きく設定することによって移動幅を変えることができる。これによって出力されるサイズを 1/ストライドの値にすることができるため、出力画像を小さくするときに使用される。

プーリング層では畳み込み層で出力された特徴マップの特定の領域から局所的な値を取り出して特徴マップを出力する。例として指定された領域内から最大となる値を出力

していく MaxPooling や領域内の平均の値を演算して取り出し AveragePooling が存在する。プーリング層によって値を取り出すことによって画像の特徴的な部分を保持するようにデータを小さくすることができる。

代表的な CNN として 2012 年に開発された Alexnet [1] や 2014 年の VGG などが挙げられる。

2.2　R-CNN

R-CNN は物体検出モデルの一つである。CNN では画像全体から特徴量を抽出するが、R-CNN ではあらかじめ領域を提案して抽出しリサイズした後に CNN への入力画像としている。領域の提案は selective search などで行われる。その後領域候補にたいして CNN で特徴量を計算しそれぞれの領域になにが映っているかを SVM によって分類する。

selective search は画像中の式特徴量を計算して画像中で類似している領域を pixel レベルで分割してバウンディングボックスを作成する手法である。これによって画像を 2000 個程度の領域へと分割している。

CNN 部分では転移学習を行って AlexNet や VGG などといった学習済のネットワークを使用することができる。

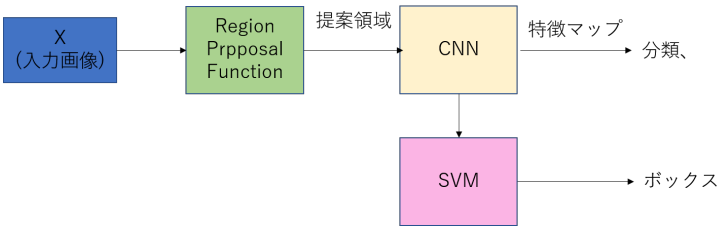

図 2.2: R-CNN の構造

2.3　Faster R-CNN

Faster R-CNN は R-CNN をベースに考案された物体検出モデルである。R-CNN やその系統である Fast R-CNN では画像中の領域提案に先述した selective search や他の手法を利用していた。しかしそれら外部のアルゴリズムを使うとコストが大きくなってしまい計算速度が遅くなることが問題であった。そこで Faster R-CNN ではそういった

アルゴリズムを使用する代わりにネットワーク内に RPN(領域提案ネットワーク) を追加することで End-to-End で学習を実施できるようになり高速化を実現をしている。

RPN は学習済のモデルよって出力された特徴マップに対して物体の候補領域とスコアを出力するネットワークである。特徴マップ上で Anchor を定義する。それらを中心に slaiding window という小さな領域で特徴マップ上を走査し複数の Anchor Boxe s を作成する。その後各 Anchor Boxe s についてそれが物体であるか背景であるかと物体である場合 Truth とどれだけ離れているかの誤差評価して学習を行う。物体か背景かの判断には IOU 指標を用いる。IOU とは 2 領域の共通面積と 2 領域の合計面積の商であり、この値が 1 に近いほど正解に近いと判断できる。っこのとき IOU が 0.3 未満である場合は背景と判断し 0.7 より多きければ物体であると判断する。正解データとのずれに関しては Box の中心座標と縦横の長さの誤差から測る。

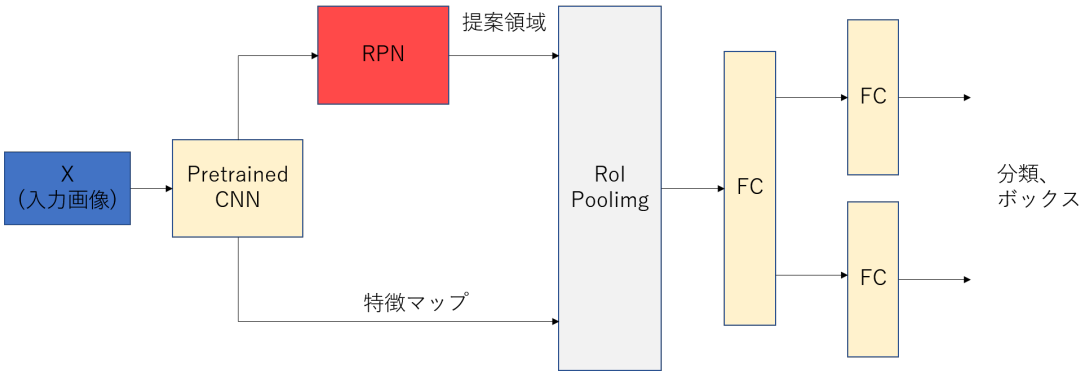

図 2.3: Faster R-CNN の構造

2.4　評価指標

機械学習を行う際には目的に対してそれぞれの手法でモデルの学習することとなる。それによりいくつかのモデルを作成した後、各手法の精度を測るために評価指標を用いる必要がある。特に物体検出における物体検出モデルの評価を行う際には IoU や mAP という指標が使われることが多く、本研究では mAP を使用して評価とモデルの比較を行っている。

IoU (Intersection over Union) とは、検出器の予測した物体のバウンディングボックスに対して正解のものがどれだけ重なり合っているかを示すことで評価を行う指標である。予測と正解のバウンディングボックスが完全に重なっている時 IoU は 1.0、全く重な

らなかった場合は 0.0 となり、予測がより正解に近いほど値が 1.0 に近似する。IoU の計算式は、検出器が予測した領域を A、正解の領域を B としたとき以下のようになる。

$$IoU = \frac{A \cap B}{A \cup B} \tag{2.1}$$

上記の式は重なった AB 領域の全体の面積で AB 領域の共通してる面積を割った値が IoU となることを示している。

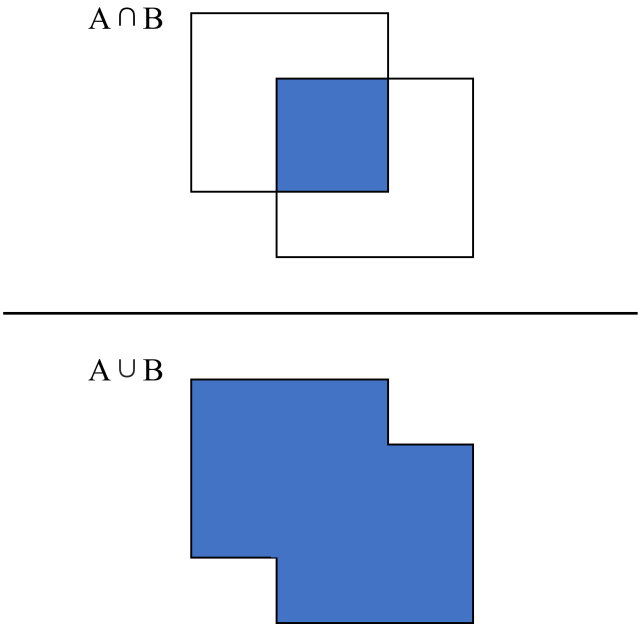

図 2.4: IoU

mAP(mean average precision) は全ての種類のクラスに対する AP(Average Precision) の平均でとった評価指標である。AP とは、特定のクラスで検出したそれぞれのオブジェクトを信頼スコアのランク順に並べて Precision - Recall 曲線下の面積によって測る評価指標である。物体検出を行う際に Precision と Recall を次の計算式によって算出する。

$$Precision = \frac{\text{正解した予測領域の数}}{\text{予測領域の数}} \tag{2.2}$$

$$Recall = \frac{\text{正解した予測領域の数}}{\text{正解領域の数}} \tag{2.3}$$

Precision は予測した領域がどれだけ正解するかという適合度を示しており、Recall は全ての物体に対してどれだけ正解するかという再現率を示している。このとき予測が正解しているかどうかについては予測領域と正解領域の重なりからあらかじめ閾値を設定し IoU を用いて判別することとなる。検出した物体について信頼スコア順に並べ Precision と Recall を表にプロットしできた曲線の下側の面積を求めることで AP の値を算出することができる。したがって AP とは適合度と再現率が共に高い予測の数が多いほど大きな値となる評価指標であることが言える。また、これら N 個のクラスの AP から平均をとったものが mAP となる。

$$AP = \int_0^1 Precision(Recall) dRecall \tag{2.4}$$

$$mAP = \frac{1}{N} \sum_{class=1}^{N} AP_class \tag{2.5}$$

Rank	Correct	Recall	Precision
1	True	0.2	1
2	True	0.4	1
3	False	0.4	0.65
4	True	0.6	0.75
5	True	0.8	0.8
6	False	0.8	0.65
7	False	0.8	0.55
8	False	0.8	0.5
9	True	1	0.55
10	False	1	0.5

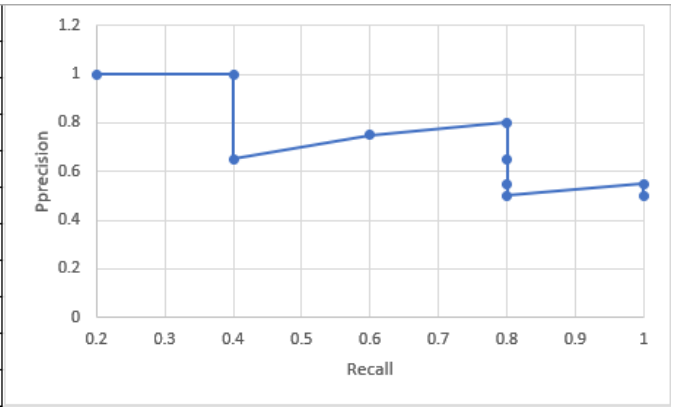

図 2.5: precision-recall 曲線の例

また各再現率でその再現率以上における適合率の最大値をその再現率の適合率とおいて平均の計算をした値を適合補完率という。下図 2.6 の霊の場合は 5 つの点をとっているため 5 点の補完適合率なる。通常は 0〜1 で 0.1 刻みにとる 11 点補完適合率や COCOmAP ではさらに拡張した 101 点補完適合率が使われている。

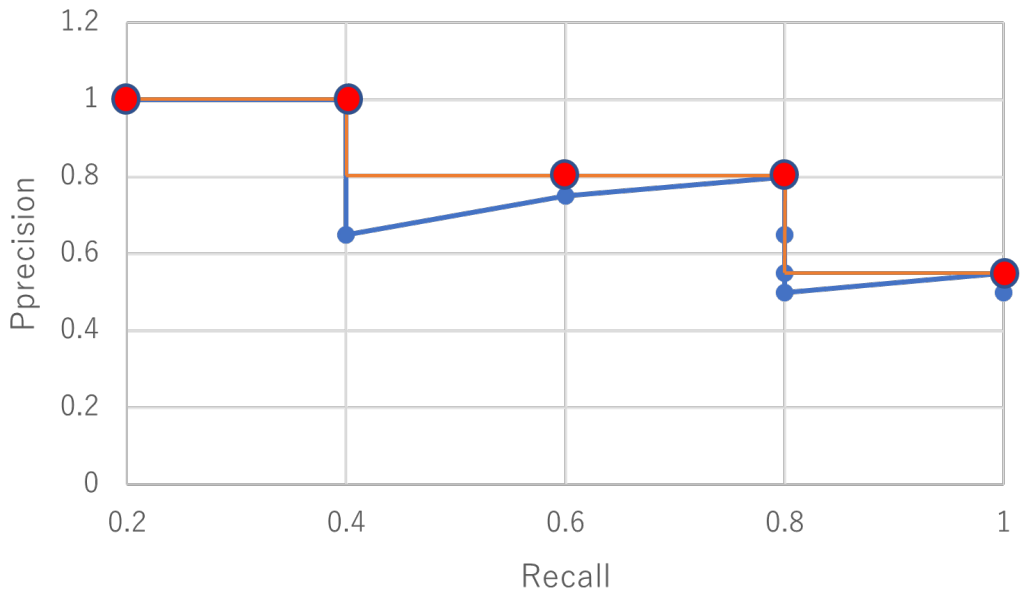

図 2.6: 5 点の補完適合率の例

第 3 章

生成モデル

3.1 GAN

GAN(Generative Adversarial Nets) [**?**] とは生成モデルを訓練させるための手法であり、敵対的生成ネットワークとも呼ばれ 2014 年に考案された。ここでの生成モデルとは、訓練データを学習しそれに近いデータを生成するためのモデルである。つまり GAN の目的は訓練データのサンプル $xp_{data}(x)$ の分布より $p_{data}(x) = p_g(z)$ となる分布を生成するモデルを得ることである。

GAN では生成モデルを得るために同時に識別モデルの学習も行うことで生成モデルの精度を高めている。

ここで GAN の二つのネットワークについて生成モデルを Generator、識別モデルを Discriminator と呼ぶ。GAN では生成モデル G と識別モデル D を競わせるように学習を行うことで互いに精度を高め合うようにできている。この二つのネットワークは互いに競い合うように訓練が行われるため敵対性とも呼ばれている。Generator のよって生成された画像を Discriminator が識別を行うが、この際 Discriminator は画像が生成されたデータか用意されたデータを正確に測ろうとし、Generator は Discriminator を欺こうと学習を行う。

G と D に対して GAN の目的関数を次のように表す。

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\big[\log D(\boldsymbol{x})\big] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}\big[\log(1 - D(G(\boldsymbol{z})))\big] \tag{3.1}$$

識別モデル D は入力されたデータが訓練データのものか生成モデルが生成したデータなのかの判別を行う。

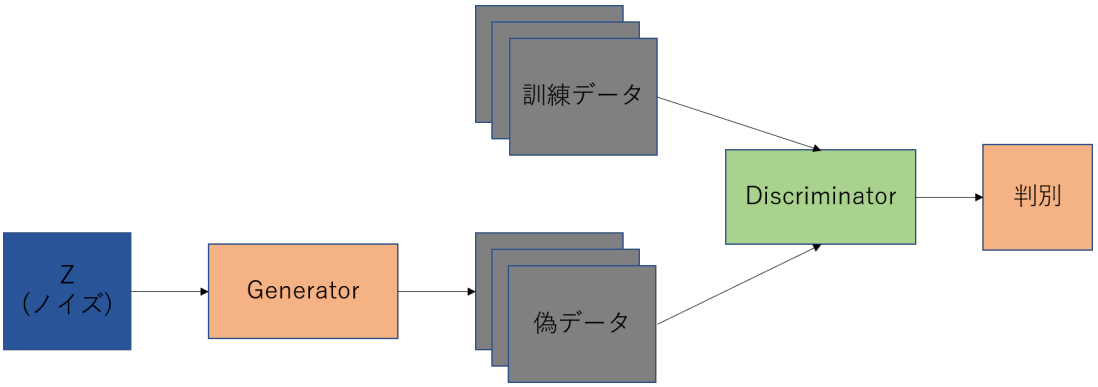

図 3.1: Faster R-CNN の構造

識別モデルの損失関数は次のように表せる。

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(\boldsymbol{x}^{(i)}) + \log(1 - D(G(\boldsymbol{z}^{(i)}))) \right] \tag{3.2}$$

識別モデル D は訓練時、事前分布 $_z(\boldsymbol{z})$ サンプリングを行いデータ $\left\{ \boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(m)} \right\}$ を得る。次に生成データを得た後、訓練データセットから観測データ $\left\{ \boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(m)} \right\}$ を抽出する。そしれ先の損失関数より勾配を計算しパラメータを更新している。

生成モデルの損失関数は以下のように表せる。

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \left[\log(1 - D(G(\boldsymbol{z}^{(i)}))) \right] \tag{3.3}$$

生成モデル G は訓練時、識別モデルと同様に事前分布 $p_z(\boldsymbol{z})$ でサンプリングを行いデータ $\left\{ \boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(m)} \right\}$ を得て、先の損失関数で勾配計算をしてパラメータの更新を行っている。

3.2　Conditional GAN

CGAN(Conditional Generative Adversarial Nets) は GAN が発表されてすぐ年内に提案された生成モデルである。CGAN では GAN の Generator や Discriminator の概念や構造がベースになっている。GAN との違いは条件付が行われているところにある。CGAN の目的関数は次のように表される。

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})} \left[\log D(\boldsymbol{x}|\boldsymbol{y}) \right] + \mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})} \left[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y}))) \right] \tag{3.4}$$

目的関数からわかるように、CGAN は GAN の目的関数に条件ベクトル y を加えたものになっていることが分かる。これらは Discreminator と Generator に画像の入力と同

時に教師データのラベルを入力して条件付けしているためである。通常の GAN では入力された画像データのみで訓練されるため、訓練データに近い分布をランダムで出力するのみにとどまる。しかし CGAN では入力データであるラベルの情報を入力するころで生成器がどういった画像を生成するかと識別機が何について正否の判別をするかをあらかじめ操作することができる。

CGAN は条件付けを行ったのみでそれ以降の学習では通常の GAN とほとんど同様である。

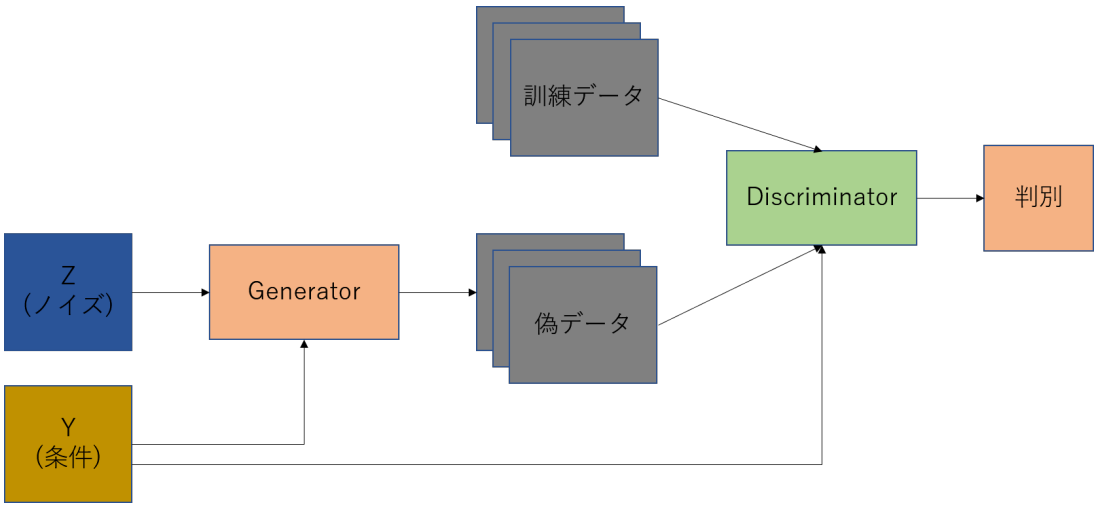

図 3.2: CGAN の構造

3.3　DCGAN

DCGAN [8] は 2015 年に通常の GAN よりも精度を向上させるものとして提案された手法である。オリジナルの GAN と DCGAN との主な違いはネットワークに全結合層を使っているか CNN を使っているかどうかの違いである。

DCGAN のアーキテクチャでも GAN と同様に Generator と Discriminator が存在しており、それらのネットワークでは CNN 層が使われている。

DCGAN では GAN の学習が安定しない問題に対し batch normalization が行われている。各々の層でデータ分布を正規化することによって学習速度向上を促したり過学習を防いでいる。また Discriminator の活性化関数には ReLU の代わりに Leaky ReLU が導入されている。ReLU は入力が 0 未満の場合出力の値を 0 とするため、$f(x) = \max(0, x)$ と表す。対して Leaky ReLU では 0 未満場合でも係数を掛けることによってマイナスと

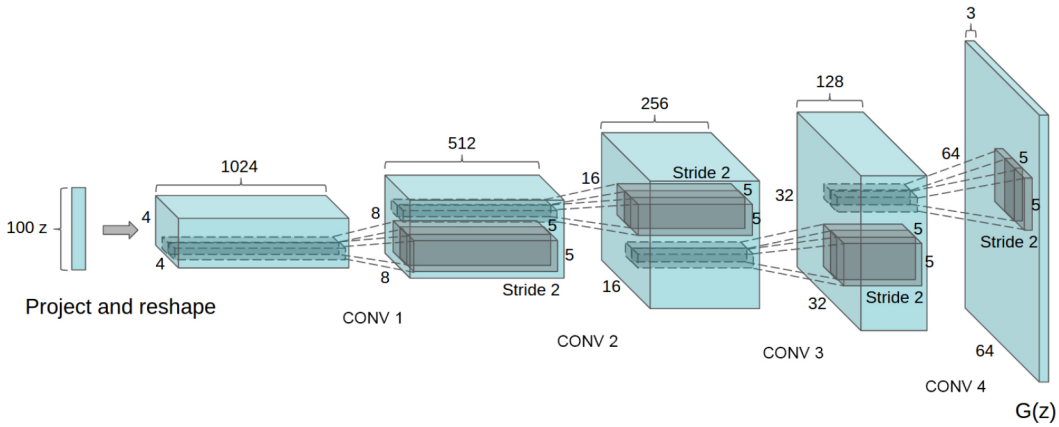

図 3.3: DCGAN

して値が保存されるようにしている。これは $f(x) = \max(ax, x)$ と表せる。これによって計算した際勾配が 0 になり誤差逆伝搬による学習が終了してしまうことを回避している。

3.4　pix2pix

pix2pix [?] は 2018 年に提案された生成モデルの手法であり、そのベースとして先に述べた DCGAN、CGAN が用いられている。pix2pix という名前は「画素から画素へ (from pixel to pixel)」という意味ある。これは、通常の GAN では訓練時に 1 枚の画像データを仕様して学習を行うのに対して、pix2pix では 2 枚のペアになっている画像を用いて学習を行うことから名付けられている。ペアとなっている画像の一方は条件画像となっており、これはつまり CGAN で扱っていた条件ベクトル y の代わりに使われているということが分かる。

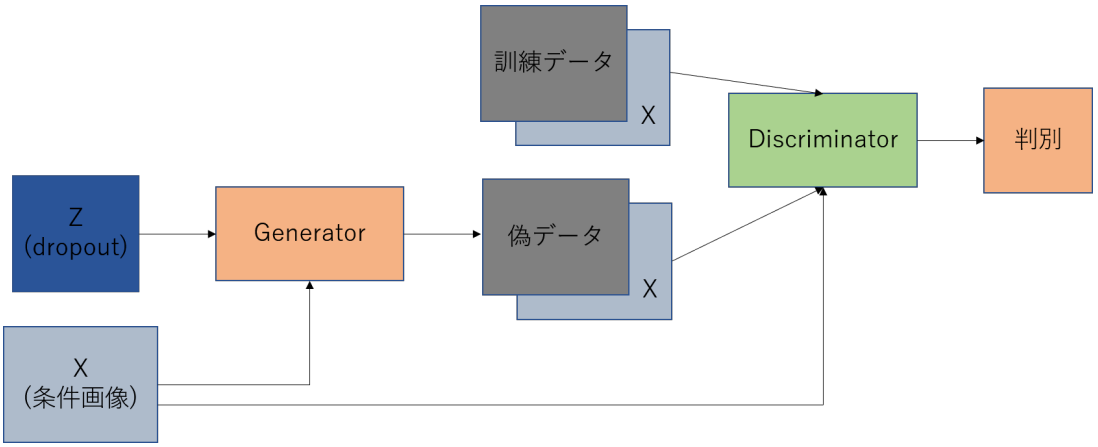

図 3.4: pix2pix の構造

pix2pix の Generator には U-Net [10] が使用されている。U-Net とは 2015 年に考案された Semantic Segmentation の手法のために用いられるネットワークである。Semantic Segmentation とは対象画像中の画素の情報を元に物体をピクセルレベルでカテゴリ分類する手法のことである。U-Net は全結合層を持たない左右対称の Encoder–Decoder 構造になっており、始めにエンコードで畳み込みを行った後デコードを行うが、その時にエンコード時に得たト特徴マップの一部を再使用している。エンコーダーの各層で出力された特徴マップをデコーダーの一部に連結するスキップ接続により物体の位置情報を保持することができる。

pix2pix では Generator に対するノイズ Z の入力を生成期の中間層にて dropout を行う事によって内的に与えている。dropout とは特定の層の出力の値を 0 に大体することである。これによって外的なノイズ Z の量を調整しなくても汎用的にノイズを与えることができるようになる。

損失関数は Discrimintor は通常の GAN と同じように表されるが、Generator は通常の GAN に L1 損失を加えたものになっている。L1 損失及び Generator の損失関数は次の通りである。

$$\mathbb{L}_{\boldsymbol{L1}} = \mathbb{E}_{x,y,z}\left[||\boldsymbol{y} - \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{z})||\right] \tag{3.5}$$

$$G* = arg\min_{G}\max_{D} \mathbb{L}_{\boldsymbol{cGAN}}(\boldsymbol{D}, \boldsymbol{G}) + \lambda\ \mathbb{L}_{\boldsymbol{L1}(\boldsymbol{G})} \tag{3.6}$$

また Discrinminator では PatchGAN により画像全体をそのまま俯瞰して使うのでは

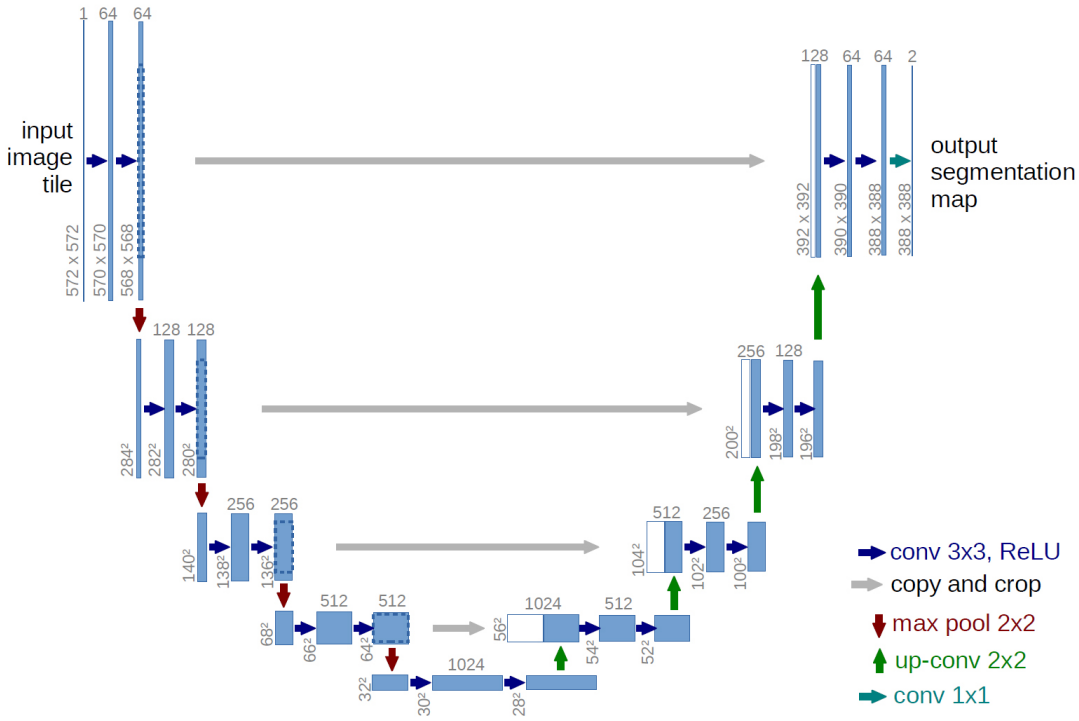

図 3.5: U-net

なく、入力画像をパッチ単位で分割してから学習識別を行っている。

3.5　CycleGAN

CycleGAN は 2017 年に提案された生成モデルの手法である。CycleGAN の入力画像は pix2pix 同様に image-to-image translation であり二枚の画像を用意して画像を生成する。pix2pix は画像は正解か不正解か分かるように画像がペアになっている必要のある教師あり学習である。しかし大量のペアになっている画像のデータセットを作成するのには時間がかかりそのコストが高いことが pix2pix の課題であった。一方 CycleGAN では画像がペアになっている必要がなく入力された画像を対象の画像へと変換することができる。すなわち教師なし学習を行う。CycleGAN は対象領域から目的領域への領域適応を行うことによって教師なし学習を実現している。つまり二つの異なる領域の画像から対応関係を導き学習を行っている。

CycleGAN は構造として通常の GAN の Discriminator と Generator を複数組み合わせた構造になっている。CycleGAN ではドメイン X の画像 x に対するドメイン Y の画像 y について、X から Y の写像を学習するだけでなく Y から X への写像も学習を行う。

それぞれの写像 $G : X \to Y$ と写像 $F : Y \to X$ と表す。これらの写像に対してドメイン Y の画像 y であるかドメイン X である画像 x から生成した画像 $G(x) = Y'$ を識別する識別器 D_Y、一方ドメイン X の画像 x であるかドメイン Y である画像 y から生成した画像 $F(y) = X'$ を識別する識別器 D_X が存在する。生成器 G と識別器 D_Y について敵耐性損失は次のようになる。

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log 1 - D_Y(G(x))] \tag{3.7}$$

この式の目的は G の最小化と D_Y の最大化 $\min_G \max_{D_Y} \mathcal{L}_{GAN}(G, D_Y, X, Y)$ である。どうように生成期 F と識別器 D_X について敵耐性損失は次のようになる。

$$\mathcal{L}_{GAN}(F, D_X, Y, X) = \mathbb{E}_{x \sim p_{data}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{data}(y)}[\log 1 - D_X(F(y))] \tag{3.8}$$

同様にに $\min_F \max_{D_X} \mathcal{L}_{GAN}(F, D_X, X, Y)$ を求められる。

通常の GAN のように生成器と識別器を各々学習してもうまく相互に影響を与えることができない。そこで CycleGAN では次のようなサイクル一貫性損失が考えられている。サイクル一貫性損失では変換した画像を逆変換した時の一貫性を保つために学習を行う。

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}||F(G(x)) - x||_1 + \mathbb{E}_{y \sim p_{data}(y)}||G(F(y)) - y||_1 \tag{3.9}$$

第 4 章

関連研究

4.1 領域適応

関連研究である [4] では GAN による領域適応を行いノイズから画像生成を行いそれらによってソースドメインからターゲットドメインへの領域適応を行う手法を提案している。当研究ではそれまで従来行われていた手動のアノテーションによる物体認識タスクのコストを解決するためにピクセルレベルでの教師なし領域適応を試みている。そのためにソースドメインからターゲットドメインの変換を行う生成器 G の学習を行うためにモデルに GAN を採択した。この際 GAN の学習を行うと同時にタスクの目的である画像の分類器の精度が同時に上がるように生成器と識別器の学習を行う。

これらの手法によって 3 つの実験とそれらの評価を行っている。一つ目は 1 から 10 の手書きデータセットである MNIST から同様の手書きデータセットである USPS への領域適応、二つ目は MNIST から背景色が反転されているデータセット MNIST-M への領域適応、最後に様々なポーズで撮像された室内環境での小さなオブジェクトのデータセットである LineMod から CAD モデルを黒い背景に様々なポーズでレンダリングすることで作成した Synthetic Cropped LineMod をソースとしターゲットをオリジナルの Cropped LineMod とした領域適応を行っている。これらは TensorFlow2 にて Adam optimizer を用いて学習されている。結果としてこれらの実験においてソースのみや他の手法より等研究による手法が最も高い精度での物体認識を達成したことで GAN を用いた領域適応による物体認識タスクの解決は有効であることが示されている。

図 4.1: ((a)MNIST のソース画像、(b) 生成画像、(c)MNIST-M のデータセットから最も近しい画像

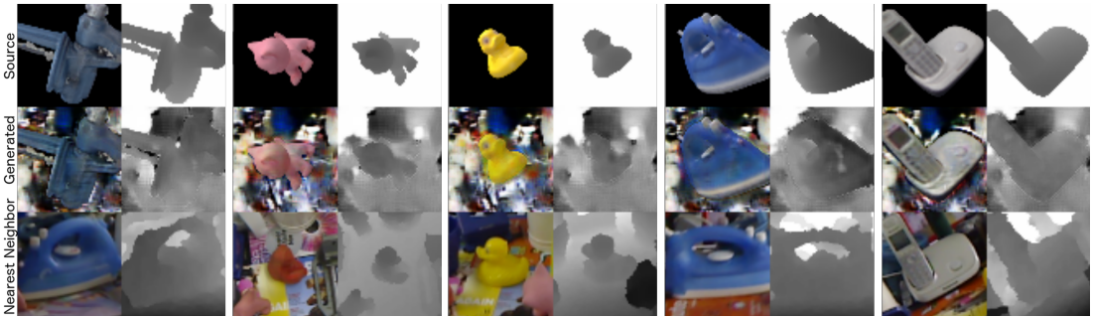

図 4.2: Synthetic Cropped LineMod よりオリジナルの Cropped LineMod に近い画像の生成

Model	MNIST to USPS	MNIST to MNIST-M
Source Only	78.9	63.6 (56.6)
CORAL [41]	81.7	57.7
MMD [45, 31]	81.1	76.9
DANN [14]	85.1	77.4
DSN [5]	91.3	83.2
CoGAN [30]	91.2	62.0
Our PixelDA	**95.9**	**98.2**
Target-only	96.5	96.4 (95.9)

図 4.3: 各手法における平均分類精度

ここで使用されているアーキテクチャは図の通りとなっており、ここでは PixelDA と呼ばれている。生成器 G は、合成画像 xs とノイズベクトル z を条件として画像を生成している。識別器 D は、実画像と偽画像の識別を行う。タスク別分類器 T は、画像にタスク別ラベル y を付与している。ストライド 1、64 チャンネルの畳み込みは、画像中では n64s1 と表示される。lrelu は leaky ReLU nonlinearity であり、ZBN はバッチ正規化層、FC は完全連結層を表している。

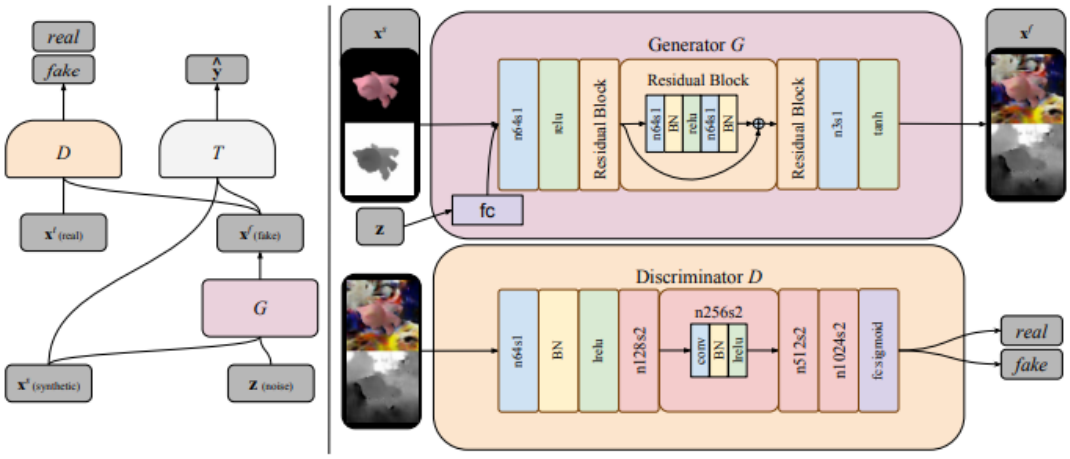

図 4.4: PixelDA のアーキテクチャ

4.2　GAN による領域適応

CycleGAN を用いた関連研究である [3] では、アノテーションされた昼間の画像データセットから偽の夜間データを生成して夜間領域の画像における自動車の検出システムの学習手法を提案している。この研究では CycleGAN を用いて昼間の画像から夜間の偽画像を生成し、それらに昼画像のアノテーションデータを対応させて物体検出器の学習行っている。実験では昼間データのみでの学習、偽の夜間データのみでの学習、昼間データと偽の夜間データの複合データセットによる学習、夜間データのみでの学習、夜間データと昼間データの複合データセットによる学習を行い 5 種類のモデルを作成し、それらのモデルの評価を比べていた。

検出器で使用されていたモデルは Faster-RCNN であった。また、モデルの評価指標には mAP を採用していた。

結論では、昼間の画像データのみで学習を行ったモデルよりも昼間データと夜間データの複合データを学習を行ったモデルの方が優れていることが述べられていた。これは昼間データで学習したモデルと昼間と偽の夜間の複合データで学習したモデルにも同じような結果が得られていた。

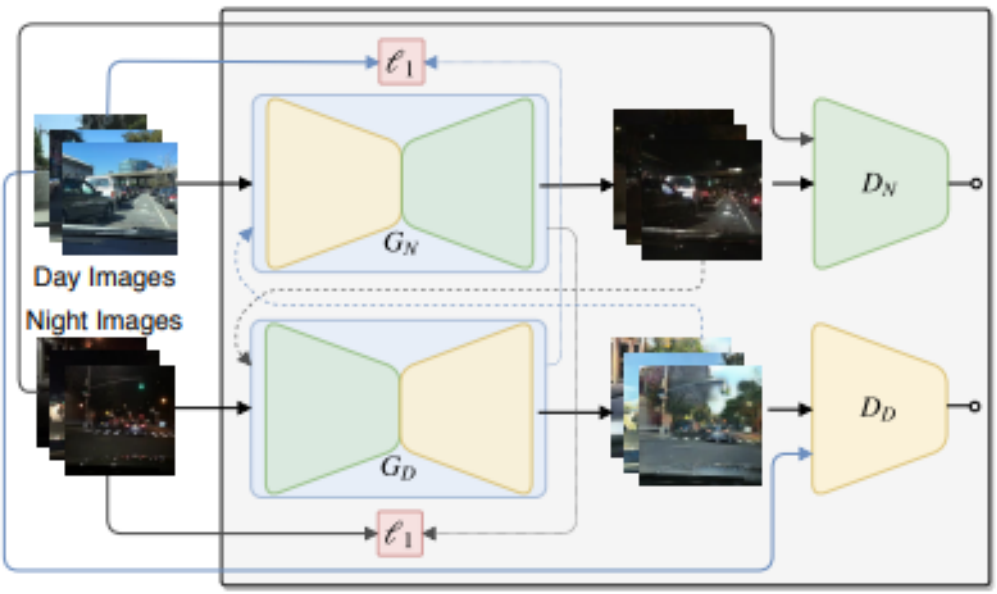

図 4.5: 論文中の CycleGAN のアーキテクチャ

図 4.6: 生成された偽の夜中の画像

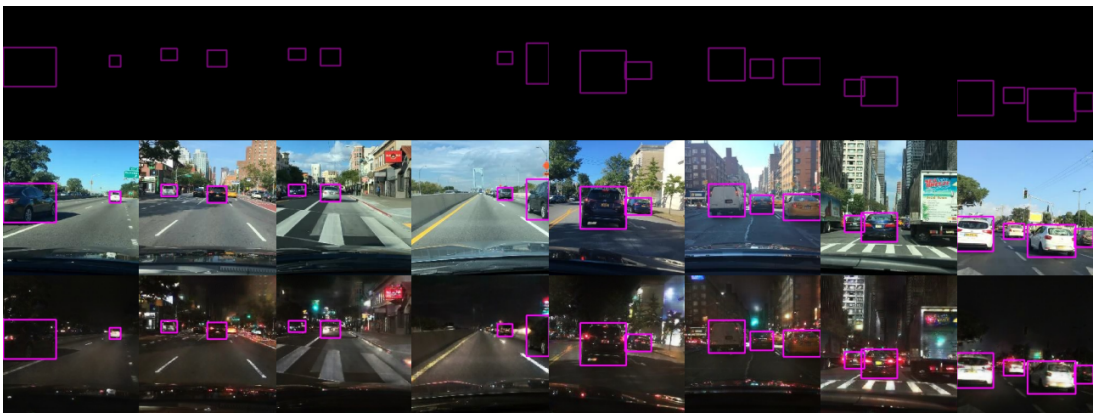

図 4.7: 日中データから変換した偽の夜中データとアノテーション

以下は得られた結果をグラフにしたものである。図 4.8 では夜と昼の画像は混合されたデータセットに対する学習評価の結果であり、これによると昼だけのデータセットで学習したモデルよりも昼と偽の夜の画像を含むデータセットがより良い結果を出してることから一定の効果を得た事を示している。また図 4.9 は夜のみのデータセットに対する学習評価の結果であり、同様に昼だけのデータセットで学習したモデルよりも偽の夜の画像および昼と偽の夜の画像のデータセットが良い結果を出してることから一定の効果を得た事を示している。

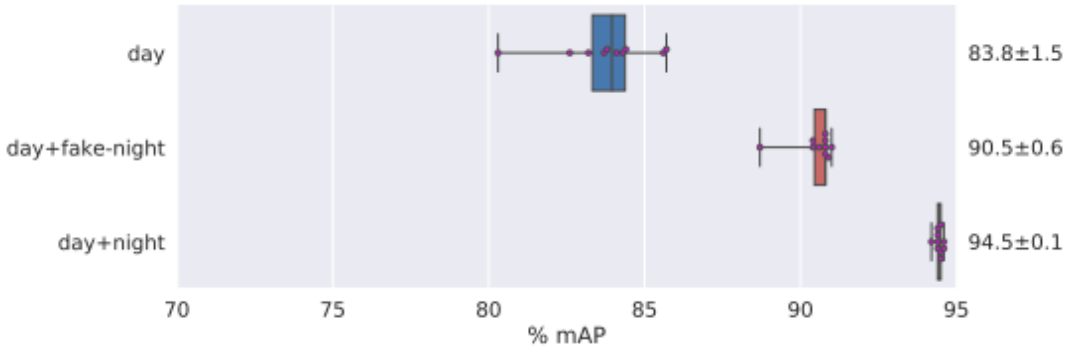

図 4.8: 昼と夜のテストデータに対する評価値

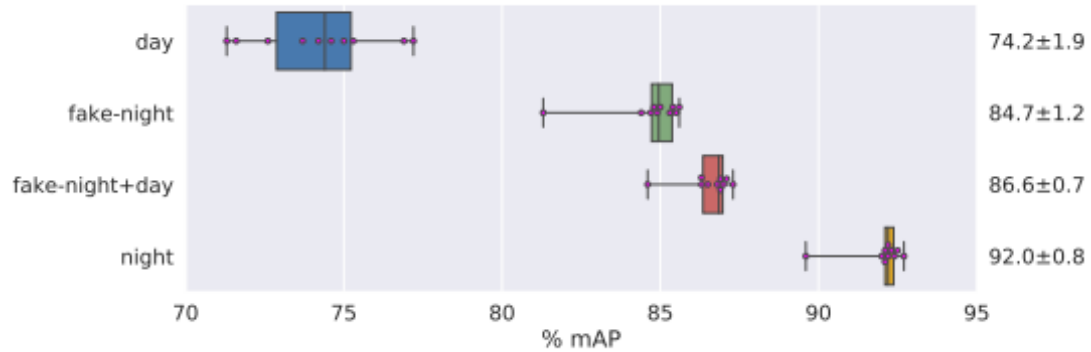

図 4.9: 夜のみのテストデータに対する評価値

第 5 章

Cycle GAN を用いた物体検出の領域適応

関連研究である [3] では CycleGAN を用いて昼間のデータセットと夜中のデータセットから学習して偽のデータセットを生成し、生成したデータセットによって自動車の物体検出の学習をしデータセットごとにモデルを評価している。

本実験ではそれらの方法とアイデアを踏襲し、自動車の物体検出について好天候時のデータセットと悪天候時データセットを利用してデータ拡張を行い、それによって学習した物体検出器の評価から拡張によってどのような効果を得られたかを測る。

好天候時のデータと悪天候時のデータセットについては BDD100K のものを採用する。BDD100K は前方車載カメラによって撮られた大量の自動車画像の及びそれらをアノテーションした情報を含むデータセットである。BDD100K のデータセットは様々な状況設定によってカテゴライズされており、悪天候時や好天候時、夜中や昼間の時間帯、高速道路や一般道路などで分けることができる。またアノテーション情報については物体のバウンディングボックスだけではなく物体のセグメンテーション情報や走行エリアのセグメンテーションなども含んでいる。BDD100K のデータセットは下記の場所で公開されている。

https://bdd-data.berkeley.edu/portal.html

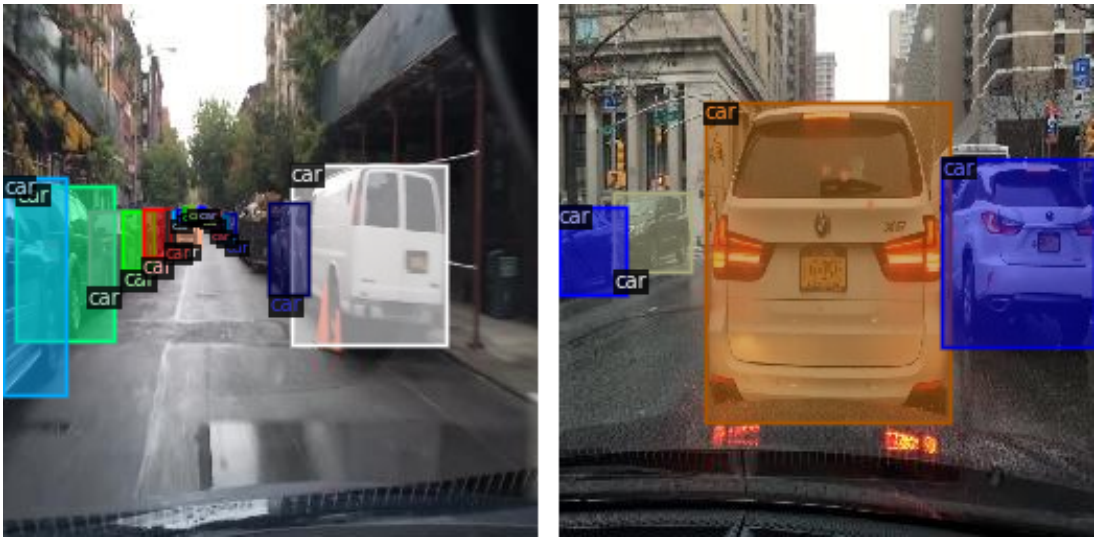

図 5.1: BDD100K の画像にアノテーションが示されている一例

下図 5.2 の抽象図は CyclaGAN のモデルの構造を示している。図では生成器 G に好天候のソース画像を入力し、同様に生成器 F には悪天候時のソース画像データを入力している。これら生成器 G と生成器 F がそれぞれターゲットの領域に適応した偽の悪天候画像 Y と偽の好天候画像 X を生成する。それら偽画像とソース画像を識別器に入力することで生成器と識別器の学習を行う。最終的に生成器 G が好天候画像から精巧な偽の悪天候画像を生成することを目指し、それによって生成される画像のデータセットをデータ拡張として利用する。

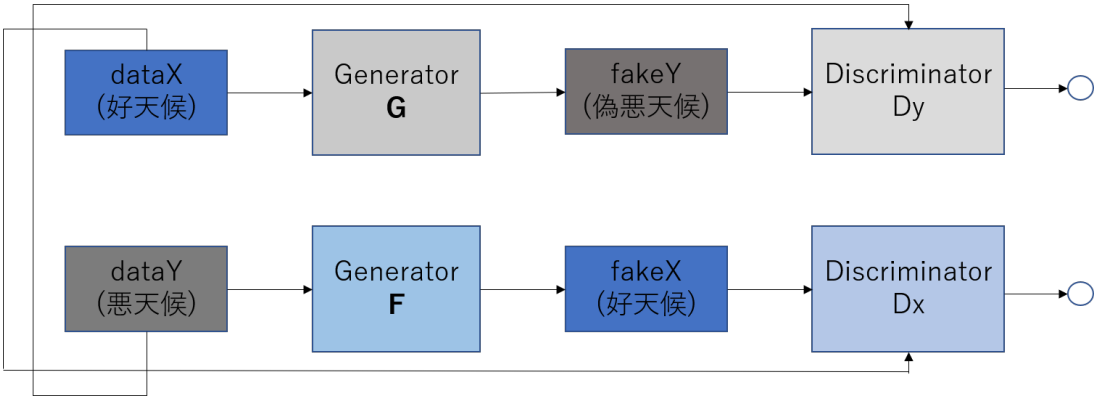

図 5.2: 本実験での CycleGAN の抽象図

CycleGAN のシステムに好天候時画像 2000 枚と悪天候時画像 2000 枚のデータセットを入力し生成器の学習を行った。生成器を用いて好天時の画像データセットから悪天候

時の偽画像データセットを生成する。この時好天候時画像での形状の一貫性が保たれたまま悪天候時のスタイルと変換されるため、もともとのアノテーションデータを偽の画像データセットへとそのまま対応させることができる。

生成した偽の画像および好天候時、悪天候時の真の画像を用いて複数のデータセットを作成した後、物体検出器の学習を行う。それらの物体検出器を後述するテストデータに対する mAP での評価を行い、偽の画像データの拡張によって精度が上がることを確認する。

評価指標である maP は COCO evaluater で提供されている COCO mAP の手法を利用する。COCO mAP では 101 点補間 AP が導入されており、IoU の閾値は複数利用して結果を出す。

第 6 章

実験

6.1 実験方法

CycleGAN によって学習を行うにあたり CycleGAN 論文の原著 [6] である Jun-Yan Zhu 氏が Github 上で CycleGAN のプログラムを公開している。以下はその URL である。

https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

これらのプログラムを github 上からローカルの環境へと clone することで CycleGAN を利用することができる。今回のプログラムを動かしたマシンの実行環境は次の通りである。

・Ubuntu 18.04.1

・python 3.6.9

・pytorch 1.6.0

・GPU GeForce RTX 2060

・CUDA 9.1.

学習用のデータセットとして悪天候時、好天候時の 2 種類のデータセットを用意し入出力を行った。BDD100K で提供されている画像のサイズは 1280*720 であるが、そのまま使用すると膨大な時間とコストがかかるためそれらの画像データとアノテーションデータのサイズを変換し CycleGAN に対する入力画像と出力画像のサイズをともに 256*256 とした。

その後 CycleGAN によって生成したデータセットを用いて複数の物体検出器の学習を行う。物体検出は GoogleColaboratory 上で detectron2 を用いて行った。物体検出器

のモデルには Faster R-CNN を利用した。それぞれの学習時のバッチサイズは 128、イテレーション数は 8000 とした。最終的に学習の完了した物体検出器をそれぞれ COCO evaluater を用いて COCO mAP による評価を行った。

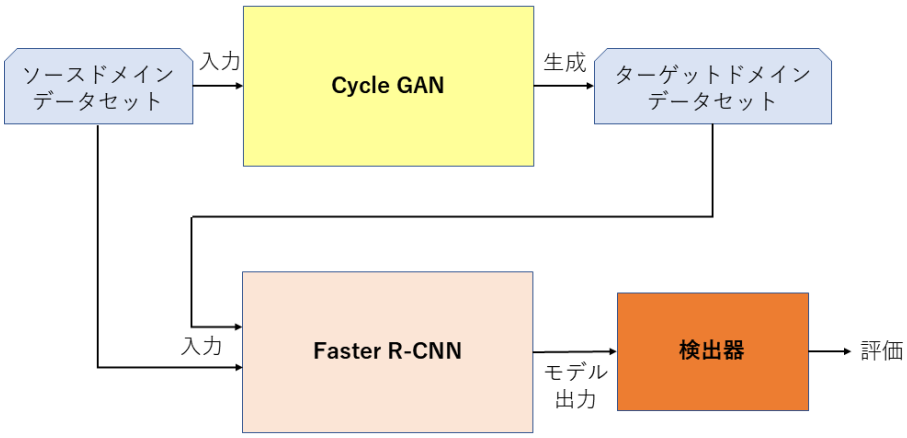

図 6.1: 実験の全体的な抽象図

6.2　実験データ

実験データとして BDD100K の画像データをアノテーションデータを使用する。BDD100K のアノテーションデータは独自の形式フォーマットで形成されている json ファイルで構成されている。今回は detectron2 および COCO evaluater を使用する都合上そのままのデータを使用するのではなくそれらを COCO フォーマット形式に整形してからしようする。使用した COCO フォーマットは次図 6.2 のような構成になっている。

```
{
  {
  "categories": [
                    {"supercategory": "none", "id": 1, "name": "car"}
                  ],
  “images”: [
                  {“file_name”: “(ファイル名 1 )", "height": 256, "width": 256, "id": 1},
                  {“file_name”: “(ファイル名 2 )", "height": 256, "width": 256, "id": 2},
                  {“file_name”: “(ファイル名 3 )", "height": 256, "width": 256, "id": 3},
                   ……
                  {“file_name”: “(ファイル名n)", "height": 256, "width": 256, "id": n}
              ]
  "annotations": [
                      {
                        "image_id": 1,
                        "bbox": [x1, y1, x2, y2],
                        "category_id": 1,
                        “id”:1,
                      },
                      {
                        "image_id": 2,
                        "bbox": [x1, y1, x2, y2],
                        "category_id": 1,
                        “id”:2,
                      },
                       ……
                      {
                        "image_id": n,
                        "bbox": [x1, y1, x2, y2],
                        "category_id": 1,
                        “id”:m,
                      }
}
```

図 6.2: coco フォーマットの形式

この形式では主に”categories”、”images”、”annotations”の三部で構成されている。”categories”では検出するクラスラベルの指定を行っており、今回の実験では車のみの検出を行うため”car”のみの指定となっている。”images”では画像の名前と id の関連づけがされている。”annotations”ではそれぞれの画像に写っている物体のバウンディングボックスとラベルのアノテーションが示されている。

画像データセットから好天時領域の画像 2000 枚と悪天候時領域の画像 2000 枚を入力する。ここで BDD100K の画像データはサイズが 1280*720 であるが、このまま CycleGAN に入力して学習をすると時間がかかりすぎてしまうため、画像のサイズを 256*256 へとリサイズする。また出力される画像サイズも 256*256 である。CycleGAN

で生成する偽のデータ数はオプションで 200 枚に設定しておく。

物体検出器で学習をするとき、データセットを次のように用意する。

1. 真の好天候時の画像データセット (1000 枚)：clearday

2. 偽の悪天候時の画像データセット (1000 枚)：fake_rainday

3. 真の好天候時の画像データセット∪偽の悪天候時の画像データセット (1000+1000 枚)：clearday ∪ fake_rainday

4. 真の悪天候時の画像データセット (1000 枚)：rainday

5. 真の好天候時の画像データセット∪真の悪天候時の画像データセット (1000 枚)：clearday ∪ rainday

またこの 5 つのデータセットに対して、以下の 2 つのテストデータ条件下で評価を行う。

A. テスト用悪天候時画像データセット∪テスト用好天候時画像データセット (1000+1000 枚)

B. テスト用悪天候時画像データセット (1000 枚)

2 つのテストデータ条件下で評価をする理由は、それぞれ「悪天候時と好天候時が発生する通常の環境下での物体検出」と「悪天候時のみの環境下での物体検出」という想定を行うためである。1-5 のデータセットによる物体検出器の学習をそれぞれ 10 回ずつ行い、最終的に A,B でのそれらの評価値の平均の値を算出する。

6.3　実験結果

CycleGAN によって偽データが生成された。次の図はうまく生成ができた画像の例である。図上部の好天時の画像をもとに悪天候時領域の画像を生成している。

しかしすべての画像がうまく変換できたわけではなく、下記の画像のように目的とは遠い結果の画像が生成される場合もあった。

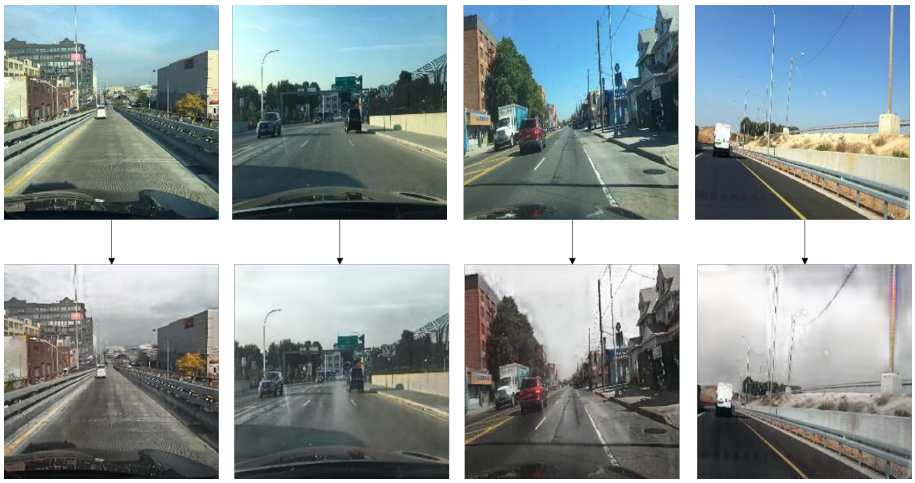

図 6.3: CycleGAN によって生成した画像

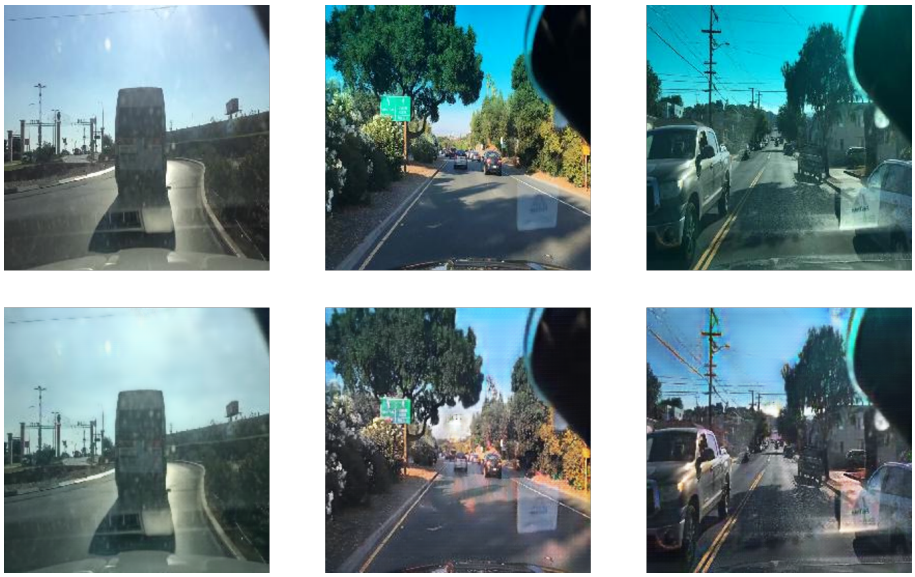

図 6.4: CycleGAN によってうまく生成できなかった画像

それぞれの物体検出器における各テストデータ下での評価値の平均は上記の通りである。

rainday test	AP	AP50	AP75	APs	APm	APl
rainday	34.5652	58.8103	35.638	26.5633	73.2255	74.9533
clearday	32.341	55.7713	33.2901	24.5773	70.0056	75.0323
fake_rainday	33.1949	57.6023	34.1313	25.4497	70.6998	74.3687
clearday ∪ rainday	35.2087	59.6696	36.6032	27.1416	73.7686	78.2359
clearday ∪ fake_rainday	33.6476	57.4971	34.87	25.7221	71.6435	77.2486

clearday ∪ rainday test	AP	AP50	AP75	APs	APm	APl
rainday	31.897	54.5022	32.7796	24.7064	73.4686	74.959
clearday	32.0764	55.1994	32.8462	25.0145	72.7797	75.4834
fake_rainday	32.1893	55.6538	33.0023	25.1742	72.9644	74.6664
clearday ∪ rainday	33.4399	56.9645	34.5205	26.2236	74.9734	77.954
clearday ∪ fake_rainday	32.9069	56.2252	33.8395	25.7229	74.1256	76.6915

第 7 章

考察

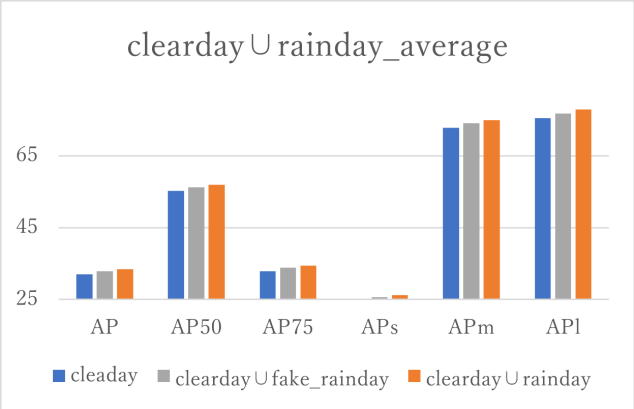

図 7.1: clearday ∪ rainday_test での平均値

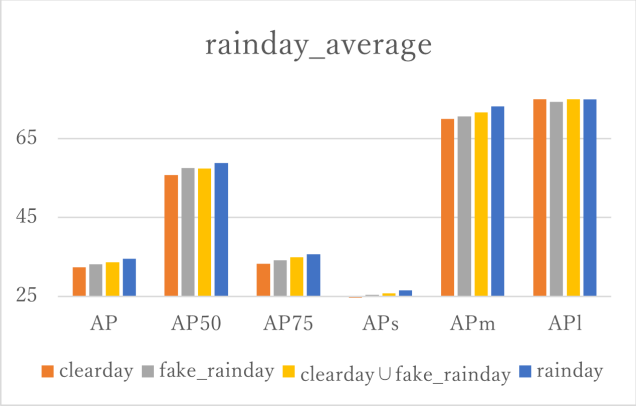

図 7.2: clearday ∪ fake_rainday_test での平均値

各テストデータで得られた評価値について、それぞれ重要と肝付が得られる項目について抜き出して図 8.1 と図 8.2 のようにグラフで表した。これらに対して考察を行う。

まずはじめに図 8.1 ではテストデータ $clearday \cup rainday$ に対する評価の平均値について考察を行う。この図のグラフでとりあげているデータセット $clearday$、$clearday \cup fake\_rainday$、$clearday \cup rainday$ で学習した物体検出器のものをそれぞれ比較する。通常テストデータと同じ構成である $clearday \cup rainday$ で学習した物体検出器が最も制度が高くなることが考えられる。また実験として $clearday$ に対して拡張を行った $clearday \cup fake\_rainday$ がより精度が高くなると考えられる。したがって $[clearday < clearday \cup fake\_rainday < clearday \cup rainday]$ となることが予測できる。実際の実験データのグラフを見てみると予測通りに評価値の大きさが並んだため、「悪天候時と好天候時が発生する通常の環境下での物体検出」において偽データによるデータ拡張には効果があると言える。

次に同様の観点で図 8.2 でのテストデータ rainday に対する評価の平均値について考察を行う。この図のグラフでとりあげているデータセット $clearday$、$fake\_rainday$、$clearday \cup fake\_rainday$、$rainday$ で学習した物体検出器のものをそれぞれ比較する。こちらでも通常テストデータと同じ構成である $rainday$ で学習した物体検出器が最も制度が高くなることが考えられる。また実験として $clearday$ に対して拡張を行った $clearday \cup fake\_rainday$ がより精度が高くなると考えられる。したがって $[clearday < clearday \cup fake\_rainday < fake\_rainday < rainday]$ または $[clearday < fake\_rainday < clearday \cup fake\_rainday < rainday]$ となることが予測できる。実際の実験データのグラフを見てみるとおおむね予測通りに評価値の大きさが並んだため、「悪天候時のみの特定の環境下での物体検出」において偽データによるデータ拡張には効果があると言える。

これらのことから悪天候時における物体検出の範囲において CycleGAN における偽データの画像生成とそれを用いたデータ拡張による物体検出は一定の効果が得られることが示されたと言える。

第 8 章

結論

本研究では CycleGAN によって好天候時のアノテーションデータを利用できる偽の悪天候時のデータセットの生成、またそれを用いて行った検出器の学習とその評価を行った。しかし今回の実験結果、特に rainday テストデータセットにおける評価の平均値は理想的な結果とはならなかった。まず APl の値が予想から全く外れていることに関しては COCO mAP の仕様から大きい物体を予測するためには今回のデータセットが不適切であったことが考えられる。次にグラフの AP、AP75、APs、APm では $[fake\_rainday < clearday \cup fake\_rainday]$ となったが、AP50 については $[clearday \cup fake\_rainday < fake\_rainday]$ となっている。これについてこのようになったことを説明できるデータがないため詳しい理由は不明であり純粋に学習時のデータの偏りの可能性もある。これに関しては 10 回の学習評価のセット回数を増やすことで解決できるかもしれない。しかしいずれにせよ fake_rainday を加えたデータセットが clearday 単体のものよりも効果があったことからやはり今回のデータ拡張には効果があることを示しているため、重要な問題ではないと考える。

謝辞

本論文の最後に、日頃から多くのご指導をしていただいた新納浩幸教授に心からの感謝をいたします。また研究をするにあたり助言をいただいた新納研究室の皆様に多くの感謝をいたします。

参考文献

[1] Alex Krizhevsky Ilya Sutskever Geoffrey E. Hinton"'ImageNet Classification with Deep Convolutional Neural Networks'(2012)

[2] Karen Simonyan and Andrew Zisserman"Very Deep Convolutional Networks for Large-Scale Image Recognition"(2015)

[3] Vinicius F. Arruda, Thiago M. Paixão, Rodrigo F. Berriel, Alberto F. De Souza, Claudine Badue, Nicu Sebe, Thiago Oliveira-Santos"Cross-Domain Car Detection Using Unsupervised Image-to-Image Translation: From Day to Night"(2019)

[4] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, Dilip Krishnan"Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks"(2017)

[5] Ian J. Goodfellow, Jean Pouget-Abadie , Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio"Generative Adversarial Nets"(2014)

[6] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros"Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks"(2017)

[7] Mehdi Mirza, Simon Osindero "'Conditional Generative Adversarial Nets'(2014)

[8] Alec Radford, Luke Metz, Soumith Chintala "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks"(2015)

[9] 'Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros'Image-to-Image Translation with Conditional Adversarial Networks"(2018)

[10] Olaf Ronneberger, Philipp Fischer, Thomas Brox"U-Net: Convolutional Networks for Biomedical Image Segmentation"(2015)

付録

A　プログラム

以下では実験中に使用した主要なプログラムを記す。これらのプログラムはGoogleColab 上で動作していることを確認している。

Googlecolab 上で detectron2 および主要な関連ライブラリのセットアップ

ソースコード A.1: xlmload

```
1  !python -m pip install pyyaml==5.1
2  import sys, os, distutils.core
3  !git clone 'https://github.com/facebookresearch/detectron2'
4  dist = distutils.core.run_setup("./detectron2/setup.py")
5  !python -m pip install {' '.join([f"'{x}'" for x in dist.
       install_requires])}
6  sys.path.insert(0, os.path.abspath('./detectron2'))
7  import torch, detectron2
8  !nvcc --version
9  TORCH_VERSION = ".".join(torch.__version__.split(".")[:2])
10 CUDA_VERSION = torch.__version__.split("+")[-1]
11 print("torch: ", TORCH_VERSION, "; cuda: ", CUDA_VERSION)
12 print("detectron2:", detectron2.__version__)
13
14 import detectron2
15 from detectron2.utils.logger import setup_logger
16 setup_logger()
17
18 import numpy as np
19 import os, json, cv2, random
20 from google.colab.patches import cv2_imshow
21
```

```
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor, DefaultTrainer
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer, ColorMode
from detectron2.data import MetadataCatalog, DatasetCatalog
from detectron2.data.datasets import register_coco_instances
```

COCO 形式のデータセットを Detectron2 のデータ形式に登録しデータローダーを行う。その後学習用パラメータのセッティングを行い学習を実行する。使用するデータセット毎にファイル名やパスの変更を行う。

ソースコード A.2: xlmload

```
register_coco_instances("images", {}, "./images/∪
    bdd100k_labels_images_cleardayrainday_coco_256.json", "./images")
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/
    faster_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("images",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/
    faster_rcnn_R_50_FPN_3x.yaml")
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.0004
cfg.SOLVER.MAX_ITER = (
    8000
)
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = (
    128
)
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1

os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)

trainer = DefaultTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

生成したモデルをレジスタに登録し COCOevaluater の COCOmAP によって評価を行う。

ソースコード A.3: xlmload

```
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.6
predictor = DefaultPredictor(cfg)

register_coco_instances("rainday_test", {}, "../test/rainday_test/
    images/bdd100k_labels_images_rainday_test_coco_256.json", "../test/
    rainday_test/images")
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader
evaluator = COCOEvaluator("rainday_test", output_dir="./output")
val_loader = build_detection_test_loader(cfg, "rainday_test")
print(inference_on_dataset(predictor.model, val_loader, evaluator))
```