Vol.2019-HPC-172 No.5
2019/12/18

人物検出における学習データ拡張のための CycleGANの分散深層学習

西川 由理[1,2,a)]　佐藤 仁[1]　小澤 順[1,2]

概要：
深層学習による人物検出では学習データの収集および教師ラベル作成に多くの時間と費用を要し，Generative Adversarial Networks (GANs) を用いた学習データ拡張による性能向上が期待されている．しかし GAN は一般に学習時間が長く，パラメータ設定も困難であることが知られており，効率の良い学習データ生成のために高速化が求められる．一方，大規模な分散深層学習は，画像認識でのパラメータ設定や最適化の汎用的な手法に関するノウハウが蓄積されているが，GAN で検討されている事例は非常に少ない．本研究では，GAN の一つである CycleGAN を用いた画像生成において分散深層学習を適用し，産業技術総合研究所の大規模 AI クラウド計算システム ABCI を用いた学習効果とスケーラビリティについて報告する．NVIDIA Tesla V100 を最大 512 台用いて評価した結果，4 台のときと比べて約 93.8 倍の高速化を確認した．また分散深層学習による生成画像を人物検出モデルの教師データに加えたところ，実画像のみで学習したモデルを超える検出精度が得られた．

キーワード：人物検出，学習データ拡張，CycleGAN, 分散深層学習，ABCI

1. はじめに

人物検出は，コンピュータービジョン分野における最重要タスクの一つである．近年，深層学習に基づく多種多様な手法により検出性能が飛躍的に向上しているが，実用化段階ではさらなる性能が求められる．しかし，そのためには，車載や防犯，マーケティング等の各応用先に対応する大規模な教師ありデータを用意する必要があり，データ収集や教師ラベル作成に多くの時間と費用を要する．

そのような中，深層学習を用いた生成モデルである Generative Adversarial Networks（GANs）による学習データ拡張が注目されている．GAN では，画像等を生成する Generator と呼ばれるデータ生成器と，生成データと実データの識別を行う Discriminator と呼ばれるデータ識別器の 2 つの深層学習モデルから成る．Discriminator は生成データを偽物として識別できるように，Generator はなるべく本物に近いデータを生成するように敵対的に学習を行うため，Discriminator が判別できないデータを生成可能な Generator を構築できれば，生成された画像等は本物に迫る品質となる．このことから，実環境を模した合成画像から実写レベルの画像を生成できれば，効率良く学習データ量を増やすことができると考えられる．

しかし GAN は，深層学習の学習パラメータである学習率やバッチサイズ等の初期値によっては学習が収束せず，高品質なデータが生成できないという課題がある．様々な学習安定化アルゴリズムが提案されているが，異なるデータセットでは効果的でない場合も多く，人物検出率向上に寄与するデータを生成するには性能チューニングのため試行回数を増やす必要がある．しかし，GAN は学習時間が長いため，効率化が求められる．一方，深層学習の高速化のために，大規模な計算資源を用いた分散深層学習に関する研究が広く行われ, 画像認識や画像検出タスクにおける汎化性能低下の解決手法やハイパーパラメータ設定のノウハウが蓄積されている．しかし，分散深層学習が GAN に適用された事例は非常に少なく，その有効性は明らかでない．

そこで我々は，GAN の一つである CycleGAN[1] を用いた画像生成に分散深層学習を適用する．計算機環境として，産業技術総合研究所の大規模 AI クラウドシステム ABCI[2][3] における NVIDIA Tesla V100 GPU を最大 512 台用い，GPU の台数を変化させたときのスケーラビリティを示す．また，CycleGAN による生成画像を，深層学習ベースの物体検出アルゴリズム YOLOv3[4] の教師データとして用い，人物検出性能向上の効果を示す．

[1] 国立研究開発法人 産業技術総合研究所
[2] パナソニック株式会社
[a)] nishikawa.yuri@aist.go.jp

2. 関連研究

機械学習のデータ拡張に GAN を応用する試みは様々に取り組まれており，本研究で着目する CycleGAN の適用事例もある [5]．特に車載画像への応用では，CycleGAN の派生アルゴリズムにより実写と CG 画像の変換を行った画像を生成し，それらを画像のピクセル単位で物体認識を行う「セマンティックセグメンテーション」[6] や，物体検出 [7] に適用する研究がある．ただし，人物検出に特化した試みは少ない．

次に，分散深層学習の関連研究について述べる．2017 年に Goyal らが提示した linear scaling 法 [8] は，バッチサイズと学習率を比例させることで汎化性能を保てるという経験則であり，同手法を基本とした改良により，ImageNet による画像認識モデルの学習時間の記録更新が続いている [9][10][11]．2019 年 4 月には，確率勾配法と Layer-wise Adaptive Rate Scaling (LARS) の組み合わせによる学習率の最適化により，2,048 台の NVIDIA Tesla V100 GPU を用い，学習時間 74.7 秒，精度 75.08%での訓練の成功事例が報告された [12]．さらに物体検出の分散深層学習においては，YOLO version 3（以後 YOLOv3）[4] に対して，Goyal らの手法の適用が良好なスケーラビリティを示すとの知見も発表されている [13]．しかし，GAN の分散深層学習の有効性については明らかではない．

最後に，GAN の並列処理の関連研究を紹介する．主に広域分散環境を利用したパラメータサーバモデルベースの深層学習手法において，連合学習（Federated Learning[14]）が注目されている．同手法は Google が発表した機械学習を非集中的な環境で行うための技術であり，クラウド環境の親モデルをローカル環境にコピーし，ローカル側で取得したデータをクラウドサーバに移行することなく学習を行い，得られた勾配情報のみをクラウドサーバに送信する．パラメータサーバモデルに基づく GAN の並列分散処理手法として，Discriminator を各ワーカーに分散する MD-GAN[15] が挙げられるが，前述のとおり広域分散環境を想定したものであり，本研究とは立場が異なる．また，複数の Generator や Discriminator を用いるなど，GAN の学習をマルチタスク，マルチプロセスで行う [16][17] 等の研究もあるが，いずれも GAN の学習の早期収束を図ることを目的としており，大規模分散環境を想定していないという点で，本研究と目的が異なる．

3. CycleGAN の分散深層学習による人物検出向け学習データ拡張

CycleGAN は，あるドメインの画像を別ドメインに変換する技術である．画像を別の画像に変換する image-to-image translation は広く研究されているが，変換元と変換先で対（ペア）になるデータセットを用意する必要があり，多くの実応用において，そのようなデータが収集できないことが課題であった．CycleGAN は，対のデータセットを用意する必要がなく，各ドメインの画像のみで学習できることから，広く注目されている．また前章でも述べたように，GAN による人物検出向け学習データ拡張に関する試みは少ないが，一般に人物検出は異なる服装や姿勢，年齢，照明条件などの実写の収集が困難であることから，本研究では，CycleGAN をデータ拡張に応用する．なお，CycleGAN の手法は付録 A.1 にて紹介する．

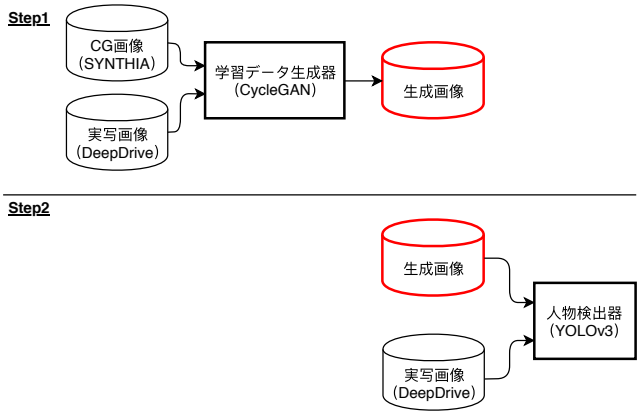

図 1　本研究における学習データ拡張

3.1 CycleGAN による人物検出向け学習データ生成

学習手順を**図 1** に示す．Step1 では，車載カメラから撮影された実写画像データセットと，車載画像を模した CG 画像（synthetic images）のデータセットとを用い，後者を CycleGAN によって人物検知の性能向上に寄与する「実写風の CG」に変換（生成）する．その後，Step2 にて，生成画像を実写画像データに加えて，人物検出器を学習することで，人物検出性能の向上を図る．

本研究で用いるデータセットの例を**図 2** に示す．実写画像データセットとしては，Berkley DeepDrive[18]（以後 DeepDrive）を用いる．シリコンバレーおよびニューヨーク等の都心エリアを中心に，異なる時間帯，気象条件および地域区分（住宅街，商業エリア等）で撮影され，人物や車など 10 種類のカテゴリについてバウンディングボックスによるアノテーションが付与されている．CG 画像データセットには，仮想都市の 3D モデルから作成された The SYNTHIA-CVPR'16 Dataset[19]（以後 SYNTHIA）を用いる．本データも異なる時間帯と気象条件を模した画像と，13 種類のカテゴリのセグメンテーションデータが提供されている．

なお，上記の学習データには人物が写っていない学習データも含まれるため，CycleGAN の学習にあたり，十分に大きな人物が含まれる画像を用いる．また本研究では，CG 人物の服装等の外観を実写風の色味やテクスチャに変換することを目的とし，両データセットの画像全体ではな

図 2 学習データセットの例（上段：Berkley DeepDrive，下段：SYNTHIA）

く，画像中の最も大きい人物のバウンディングボックスが画像の中心となるよう 256 × 256 ピクセルにトリミングした上で，学習器に与える．ハイパーパラメータは Zhu らの実装に準拠し，200 epochs の学習を行う．

3.2 CycleGAN の分散深層学習

CycleGAN の分散深層学習手法について述べる．Zhu らによるオリジナルの実装 [1] は，PyTorch により記述されている．PyTorch では，学習モデルに `nn.DataParallel` を適用した上で学習を実行すると，ノード内のマルチ GPU に対し，分散処理が自動的に行われる．一般に学習過程では，1) 予測を行ってその誤差を計算（Forward 処理）し，2) 誤差を減らす方向の勾配を計算（Backward 処理）し，その勾配を用いて学習モデルを更新する．PyTorch では**図 3** に示すように，Forward 処理で 1 回の scatter と gather 処理，Backward 処理で 1 回の scatter 処理と reduce 処理が各々行われる．なお，プログラマが実装時に図 3 の内部挙動を意識する必要はなく，`nn.DataParallel` を適用し，プログラムの初期化時に GPU 数に比例するバッチ数を明示的に与えるのみでノード内のマルチ GPU を恩恵を受けることができ，Zhu らの実装にも同手法が適用されている．

一方，PyTorch でマルチノード・マルチ GPU 処理を行うには，大きく (1) `nn.DistributedDataParallel` の利用，または (2) horovod[21] の利用という選択肢がある．前者は，各プロセスの計算資源（GPU 等）への割り当てをユーザが明示的に記述でき，性能チューニングの余地があるといった特色があるが，大規模分散環境下での実績が少ない．一方，後者の horovod は，深層学習フレームワークを複数 GPU で分散処理させる OpenMPI ベースのアプリケーションで，PyTorch，Tensorflow, Keras，MXNet に対応している（2019 年 11 月現在）．MPI のプロセス ID が GPU の ID に対応するため実装がシンプルになるほか，大規模分散環境に適用された成功事例 [22] もあることから，本研究では (2) によってマルチノードの処理を行う．

実装方法としては，基本的に，各プロセス（GPU）がデータセットの一部を担当する．具体的には，総データ数を D，プロセス数を n としたとき，各プロセスに D/n のデータをランダムに割り当てて担当する．ただし，プロセス数が増加したときに，各プロセスが担当するデータ量が少なくなることで各々の学習に偏りが生じ，精度が低下することが懸念される．そこで，パラメタ $scale$ を導入し，各プロセスが $scale \times D/n$ のデータをランダムに割り当て複数プロセスでデータを重複して学習するようにする．本研究では，4.1 節にて後述のように 1 ノードあたり GPU 4 台の環境で評価を行うため，各 GPU に学習データセットの全てに相当するデータ量が割り当てられる場合の生成画像の品質，およびその生成画像を人物検出の学習データとして用いたときの性能をベースラインとして算出することを目的とし，$scale = 4$ と設定する．

各プロセスは，自身の担当するデータに対し，各々 Forward, Backward 処理を行った後，Generator と Discriminator の重みを更新する Optimizer 処理の際に，各プロセスの勾配を Allreduce 処理により共有する．この Allreduce 処理は，horovod の分散深層学習用クラスである `DistributedOptimizer` を用いて記述する．なお，学習率の設定は異なる GPU 台数に対して変更せず，A.1.2 で述べる手法を採用する．

3.3 人物検出器の学習

人物検出器の学習には，深層学習ベースの一般物体検出アルゴリズム YOLOv3[4] を適用する．本手法は単一の CNN で画像認識（分類）タスクと物体の領域推定を行い，高速かつ高精度な画像検出が可能であることから，広く応用されている．

YOLOv3 の学習においては，Redmon らが画像認識用データセット ImageNet によってプレトレーニングした

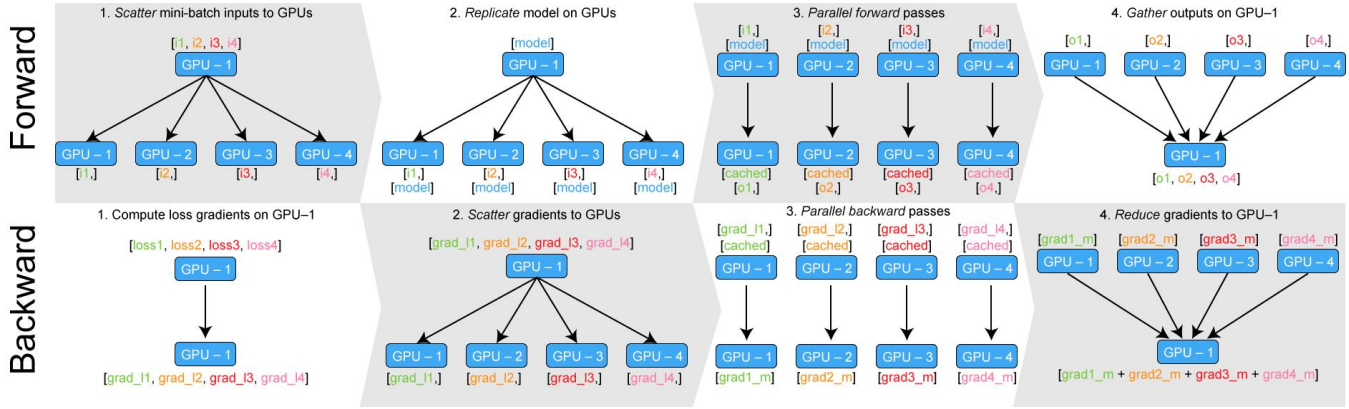

図 3　PyTorch の DataParallel の挙動 [20]

表 1　ABCI の計算ノードのスペック

CPU	Intel Xeon Gold 6148 (27.5M Cache, 2.40 GHz, 20 core) × 2
GPU	NVIDIA Tesla V100 SXM2 × 4
Mem	384 GiB
SSD	1.6TB NVMe SSD × 1

公開重みデータに基づき，DeepDrive の実写画像に SYNTHIA を元に生成した生成画像を加えた学習データセットでファインチューニングを行う．なお，CycleGAN と異なり，YOLOv3 では背景を含む画像全体に対し，人物のみを検出する 1 クラス検出器として学習を行う．DeepDrive および SYNTHIA データセットの画像サイズを揃えた上で YOLOv3 に与え，プログラム内部で 608 × 608 にリサイズする．また正解データには，DeepDrive は公開されている正解ラベルのうち，人物（”person”）とアノテーションされたデータを抽出して用いる．SYNTHIA は人物領域のバウンディングボックスを手動でアノテーションしたデータを用いる．基本的なハイパーパラメータは Redmon らの実装に従い，バッチサイズは 64，iterations（学習の繰り返し回数）は 5000 とする．

4. 評価

4.1 実験環境

分散深層学習の評価は，産業技術総合研究所の大規模 AI クラウドシステム ABCI 上で行った．**表 1** に計算ノード 1 台のスペックを示す．ABCI は 1 ノードあたり 4 台の GPU を搭載する．また計算ノード間は EDR Infiniband により，Full-bisection Fat Tree 構成で接続される．ただしラックを跨ぐ計算ノード間は Full-bisection の帯域が 1/3 となるように接続されている．

計算ノードの OS は CentOS 7.5, Linux のカーネルは v3.10.0 である．またソフトウェアについて，GPU に対しては，CUDA Toolkit v9.2.148.1，CuDNN v7.3.1 を使用し，GPU 間の集団通信を行うため NCCL v2.3.5-2 と OpenMPI v3.1.2 を使用した．Python およびライブラリは、Python v.3.5.2, PyTorch v1.0.0, horovod v0.15.2 を使用した．また分散処理機能を有効にするため，ソースからビルドした PyTorch を Singularity v.2.6.1 によってコンテナ化し，そのイメージを呼び出すことで学習および評価を実施した．学習は，ABCI の GPU 4 台，64 台，128 台，256 台，512 台でそれぞれ行った．

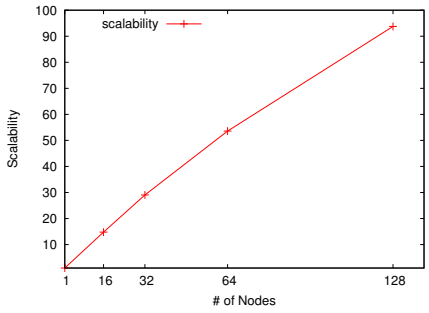

図 4　スケーラビリティ

CycleGAN の学習では，人物が大きく映る画像を DeepDrive から 4822 枚，SYNTHIA から 3163 枚抽出した．一方，YOLOv3 の学習では，実写データの入手が困難なシーンを生成画像によりオーグメンテーションするケースを想定して，生成画像をより多く用いて学習することとし，DeepDrive から 500 枚，SYNTHIA からの生成画像を 1,017 枚用いた．

4.2 スケーラビリティ

図 1 の Step 1 に示した，DeepDrive と SYNTHIA を用いた CycleGAN の分散深層学習を行ったときの実行時間から求めた台数効果を評価した．**図 4** に示すように，概ねスケーラブルな処理性能が確認された．なお横軸はノード数，縦軸は 1 ノード（GPU 4 台）の実行時間を基準としたときのスケーラビリティを表す．1 ノードでの計算時間は約 68.77 時間（約 2 日と 20 時間 46 分）であったのに対し，128 ノード（GPU 512 台）では約 0.73 時間（約 44 分）となり，約 93.8 倍高速となった．

4.3 生成画像を用いた人物検出精度

次に，図 1 の Step 2 に示した YOLOv3 の学習による，人物検出の性能を評価した．結果を**表 2** に示す．表における矢印↑は値が大きいほど良いことを意味し，↓はその逆である．なお，学習時には DeepDrive と SYNTHIA（による生成画像）を加えて学習したが，検出性能の評価では DeepDrive のみを用いた．また表には，学習過程において最も高い mAP を示した時点での学習モデルによる評価結果を掲載した．学習は下記条件を比較した．

(a) DeepDrive 500 枚
(b) (a) + SYNTHIA 1017 枚 (変換前の CG 画像)
(c) 同上 (GPU 4 台で学習した CycleGAN による生成画像)
(d) 同上 (GPU 64 台　〃)
(e) 同上（GPU 128 台　〃）
(f) 同上（GPU 256 台　〃）
(g) 同上（GPU 512 台　〃）

ベースライン (a) および CG 画像を加えた (b) と比べ，CycleGAN による生成画像を与えた (c)-(g) では，いずれも F1 および mAP が向上した．特に mAP は，(a) が 34.3% に対し，(c)-(g) のいずれの GPU 台数の場合も 7%以上向上した．さらに，各 mAP の値には，大きな差はないことも確認された．このことから，分散深層学習によって高速に生成した画像が，人物検出モデルの性能向上に寄与することが示された．

(f) では TP（true positive: 真の正解）と FP（false positive: 誤検出）の値が他と比べ減少し，FN（false negative : 見逃し）の値が増加している．YOLOv3 は学習過程において，誤検知に対して厳しい学習と，見逃しに対して厳しい学習を行うことがあると考えられ，(f) のケースでは前者が獲得された時の mAP が最も高かったことによる．次点の mAP は 41.12%であり，このとき TP=4866, FP=4741, FN=5907 であった．

なお今回は，DeepDrive，SYNTHIA ともに比較的少ない枚数で YOLOv3 の学習を行った．今後は双方の学習データ数を増やした場合にも同様の結果が得られるか否かを検証する予定である．

4.4 CycleGAN の分散深層学習による生成画像とその品質評価

最後に，生成した画像の品質評価について述べる．ABCI の GPU 4 台, 128 台，512 台を用いた分散深層学習により生成した画像の例を，**図 5** に示す．いずれの場合も，学習が進むに従い，学習初期で見られたモアレが目立たなくなり，色味や服装のテクスチャが実写データに近づく様子が見られたが，ただし，200 epoch 後に生成された画像は同一ではなく，肉眼でも色味の違いが識別できる程度に異なることが分かった．

指標として，(1) で定義される Fréchet Inception Distance[23]（FID）を用い，異なる GPU 台数で学習した時の推移を比較した．FID は，2 つの画像集合 A_1, A_2 の分布間の距離を表す指標であり，μ_1, μ_2 がそれぞれの平均ベクトル，Σ_1, Σ_2 が共分散行列を表す．A_1 を実画像，A_2 を生成画像の集合と見なしたとき，一般に値が小さいほど GAN の学習結果が良好（実画像に似ている）とされる．

$$|\mu_1 - \mu_2|^2 + \mathrm{tr}\left(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{\frac{1}{2}}\right) \quad (1)$$

結果を**図 6** に示す．GPU 4 台（黒線）の FID の値が 200 epoch 時点で約 221.0 であったのに対し，分散深層学習による生成画像の値は GPU 256 台（緑線），512 台（赤線）でそれぞれ約 269.2，270.7 となった．

本研究の分散深層学習による生成画像の人物検出の学習データ拡張への応用時には，表 2 に示すように，品質の差の影響は見られなかった．さらに，図 5 に示すように，GPU の台数を増やしても，主観的な見た目は破綻していないことがわかる．FID は (1) に示すように，共分散行列の項を含み，FID の値実写画像と生成画像の分布の広がりが一致するほど FID の値は小さくなる．すなわち値が大きいということは，実写画像の集合と比べ，生成画像の方がばらつきの大きい画像を生成したことが示唆されるが，そのことによって実写による学習データとして不足していた画像を生成でき，人物検出に有効だった可能性があると考えられる．

一方，CycleGAN による写真や動画編集，アートワーク生成等，生成画像の見た目を重視する応用においては，本研究の分散深層学習手法に課題がある可能性がある．今後は，学習率や各プロセスに割り当てる学習データ量等，異なるハイパーパラメータによる調査を行う予定である．

5. まとめ

本稿では，人物検出の性能向上を目的として，GAN の一つである CycleGAN による学習データ拡張と，その効率化のための分散深層学習の手法と，人物検出性能と学習時間の評価について述べた．分散深層学習では，PyTorch のノード内分散処理機能と，horovod によるノード間分散処理機能を併用して実装を行った．産業技術総合研究所の大規模 AI クラウドシステム ABCI における NVIDIA Tesla V100 GPU を最大 512 台用いた CycleGAN の分散深層学習による生成画像を，実写の車載画像の教師データに加えて人物検出モデルの学習を行ったところ，いずれの GPU 台数による生成データでも物体検出の評価指標である mAP が 7%以上向上した．さらに，GPU 512 台による学習時間は，GPU 4 台のときと比較し約 93.8 倍の高速化を確認した．以上より，分散深層学習によって高速に生成した学習データが，人物検出モデルの性能向上に寄与することが示された．なお本研究では，合成（CG）画像としてオープン

表 2　CycleGAN の生成画像を学習データに用いた検出率

	TP↑	FP↓	FN↓	Precision↑	Recall↑	F1↑	mAP@0.50[%]↑
(a) DeepDrive のみ	3930	2920	6843	0.57	0.36	0.45	34.34
(b) (a) + CG	4086	2643	6687	0.61	0.38	0.47	39.45
(c) (a) + 4 GPU 生成画像	4813	4560	5960	0.51	0.45	0.48	41.39
(d) (a) + 64 GPU 生成画像	4403	2625	6370	0.63	0.41	0.49	41.84
(e) (a) + 128 GPU 生成画像	4718	3762	6055	0.56	0.44	0.49	42.03
(f) (a) + 256 GPU 生成画像	3996	1938	6777	0.67	0.37	0.48	41.40
(g) (a) + 512 GPU 生成画像	4604	3616	6169	0.56	0.43	0.48	41.53

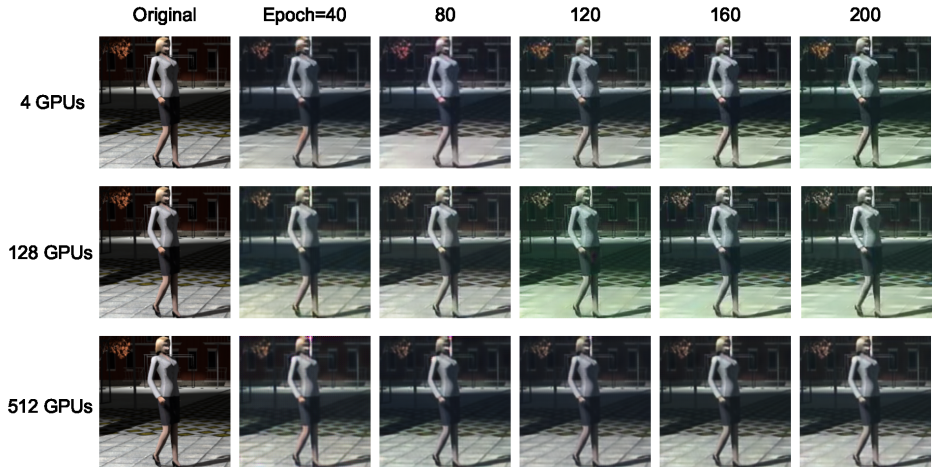

図 5　CycleGAN の分散深層学習による生成画像の例

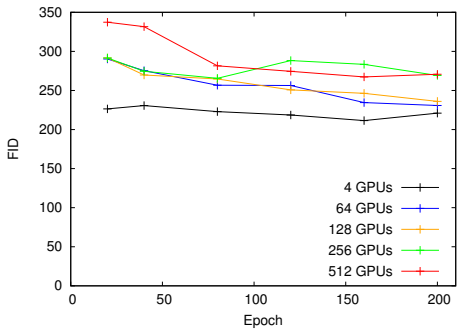

図 6　GPU 台数による FID の比較

データである SYNTHIA を用いたが，今後は，DeepDrive で高い人物検出性能が得られない撮影条件（人物の属性，姿勢，気象条件等）を分析し，それを模した独自の合成画像の作成および GAN を用いたデータ拡張によって，人物検出率の向上を図りたい．

謝辞　本研究を進めるにあたり，GAN による画像生成，学習データセット，人物検出手法に関する知見をはじめ様々なご教示を頂いたパナソニック株式会社の石井育規氏に深く感謝いたします．

参考文献

[1] Zhu, J.-Y., Park, T., Isola, P. and Efros, A. A.: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, *ICCV* (2017).
[2] 小川宏高，松岡　聡，佐藤　仁，高野了成，滝澤真一郎，谷村勇輔，三浦信一，関口智嗣：AI 橋渡しクラウド–AI Bridging Cloud Infrastructure (ABCI)–の構想，情報処理学会研究報告，Vol. 2017-HPC-160, No. 28, pp. 1–7 (2017).
[3] ABCI: https://abci.ai/.
[4] Redmon.J and Farhadi, A.: YOLOv3: An Incremental Improvement, *CoRR*, Vol. abs/1804.02767 (2018).
[5] 松岡拓未，日朝祐太，大竹義人，高尾正樹，高嶋和磨，菅野伸彦，佐藤嘉伸：CycleGAN を用いた CT-マルチパラメトリック MR 画像変換，*Medical Imaging Technology*, Vol. 37, No. 3, pp. 130–136 (2019).
[6] Choi, J., Kim, T. and Kim, C.: Self-Ensembling With GAN-Based Data Augmentation for Domain Adaptation in Semantic Segmentation, *ICCV* (2019).
[7] Huang, S.-W., Lin, C.-T., Chen, S.-P., Wu, Y.-Y., Hsu, P.-H. and Lai, S.-H.: AugGAN: Cross Domain Adaptation with GAN-Based Data Augmentation, *ECCV* (2018).
[8] Goyal, P., Dollár, P., Girshick, R. B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y. and He, K.: Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, *CoRR*, Vol. abs/1706.02677 (2017).
[9] Akiba, T., Suzuki, S. and Fukuda, K.: Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes, *CoRR*, Vol. abs/1711.04325 (2017).
[10] Mikami, H., Suganuma, H., U.-Chupala, P., Tanaka, Y. and Kageyama, Y.: ImageNet/ResNet-50 Training in 224 Seconds, *CoRR*, Vol. abs/1811.05233 (2018).
[11] Ying, C., Kumar, S., Chen, D., Wang, T. and Cheng, Y.: Image Classification at Supercomputer Scale, *CoRR*, Vol. abs/1811.06992 (2018).
[12] Yamazaki, M., Kasagi, A., Tabuchi, A., Honda, T., Miwa, M., Fukumoto, N., Tabaru, T., Ike, A. and Nakashima, K.: Yet Another Accelerated SGD: ResNet-

50 Training on ImageNet in 74.7 seconds, *CoRR*, Vol. abs/1903.12650 (2019).
[13] 西川由理, 佐藤仁, 小澤順: Darknet53 を用いた ImageNet による一般物体認識と YOLOv3 を用いた物体検出の分散深層学習, 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2019-HPC-168(24) (2019).
[14] Konecný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T. and Bacon, D.: Federated Learning: Strategies for Improving Communication Efficiency, *CoRR*, Vol. abs/1610.05492 (2016).
[15] Hardy, C., Merrer, E. L. and Sericola, B.: MD-GAN: Multi-Discriminator Generative Adversarial Networks for Distributed Datasets, *CoRR*, Vol. abs/1811.03850 (2018).
[16] Im, D. J., Ma, H., Kim, C. D. and Taylor, G. W.: Generative Adversarial Parallelization, *CoRR*, Vol. abs/1612.04021 (2016).
[17] Durugkar, I. P., Gemp, I. and Mahadevan, S.: Generative Multi-Adversarial Networks, *CoRR*, Vol. abs/1611.01673 (2016).
[18] Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V. and Darrell, T.: BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling, *CoRR*, Vol. abs/1805.04687 (2018).
[19] Ros, G., Sellart, L., Materzynska, J., Vazquez, D. and Lopez, A. M.: The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes, *CVPR* (2016).
[20] Wolf, T.: Training Neural Nets on Larger Batches: Practical Tips for 1-GPU, Multi-GPU & Distributed setups.
[21] Sergeev, A. and Balso, M. D.: Horovod: fast and easy distributed deep learning in TensorFlow, *arXiv preprint arXiv:1802.05799* (2018).
[22] Kurth, T., Treichler, S., Romero, J., Mudigonda, M., Luehr, N., Phillips, E., Mahesh, A., Matheson, M., Deslippe, J., Fatica, M., Prabhat and Houston, M.: Exascale Deep Learning for Climate Analytics, *SC '18*, pp. 51:1–51:12 (2018).
[23] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. and Hochreiter, S.: GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, *NIPS*, pp. 6626–6637 (2017).

付　　録

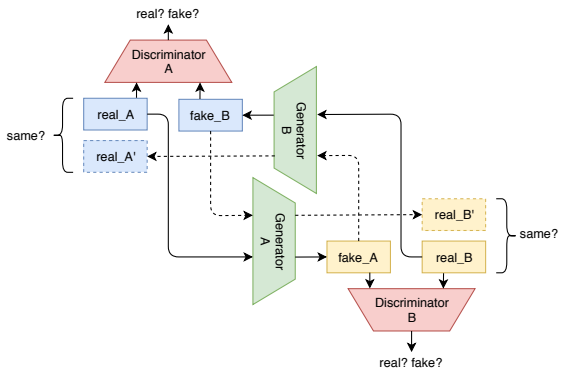

図 **A·1**　CycleGAN の学習モデル

A.1　CycleGAN を用いた画像生成

A.1.1　学習モデルと損失関数

学習モデルの概要を図 **A·1** に示す．図における長方形は画像，台形は学習モデル（サブモデル）を表す．CycleGAN は画像を生成する Generator とその画像が本物かどうかを見分ける Discriminator から成ることに加え，「別ドメインに変換した画像を元のドメインに復元できるように学習する」という特色がある．そのため，Generator が 2 つと Discriminator が 2 つの計 4 つのモデルから成り，全体は CycleGAN の名のとおり循環型のアーキテクチャとなる．

CycleGAN では Generator が生成した画像（fake）か本物（real）画像であるか否かを識別する Discriminator と Generator の学習が交互に行われる．Discriminator は生成画像と本物画像を fake と real に正しく識別できるように学習されるものであり，学習には，生成画像と本物画像をそれぞれ入力し，識別を行ったときの誤差が損失 (loss) として利用される．

Generator は 3 種類の損失関数を重み付けして学習する．Adversarial Loss は，生成画像を Discriminator に入力したとき，real として Discriminator が誤識別するように Generator を学習する損失である．GAN に欠かせない loss であり，Discriminator を騙すように Generator を学習することで，本物画像に近い画像を生成できる．一方，Cycle Consistency Loss は CycleGAN に特有の損失であり，生成画像を介して入力画像を再構成できるように Generator を学習する損失である．具体的には，図 A·1 における $real\_A$ と $real\_A'$, $real\_B$ と $real\_B'$ の分布を比較することで算出する．CycleGAN の学習においては，以上の Adversarial Loss と Cycle Consistency Loss が基本となるが，Zhu ら [1] はこれらに加え，過度な画像変換を抑制するために，「変換元の画像を Generator に入力した時はそのまま出力する」ように学習を行う際に生じる Identity Mapping Loss を導入し，変換元の画像の色味が変換先にも反映されるようにしている．

A.1.2　学習手順

Zhu らの実験では，学習率が最初の 100 epoch で初期値 0.0002 を維持するように，次の 100 epoch で線形に減衰するように設定され，計 200 epoch の学習が行われる．Discriminator の学習率は Generator の 1/2 とし，前者の学習が相対的に遅くなるように設定される．損失関数は，前節で述べたように 3 種類の loss の重み付け和からなり，Cycle Consistency Loss の重みは 10，Identity Mapping Loss は 5 が適用される．最適化は Adam solver を用い，バッチサイズは 1 を適用する．また各ドメインから数百から数千枚のデータが用いられ，それぞれの枚数は異なる．