

一般物体検出 YOLO の分散深層学習による性能評価

西川 由理^{1,2,a)} 佐藤 仁¹ 小澤 順¹

概要：近年，画像分類タスクを中心に，大規模並列分散環境下での分散深層学習による学習高速化のノウハウが，蓄積されつつある．しかし，その他のタスクにおいて，分散深層学習の性能を評価した事例はまだ多くない．本稿では，産業技術総合研究所の AI 橋渡しクラウド ABCI を用いて一般物体検出手法 YOLOv2 の分散深層学習を行い，その学習効果と台数効果を述べる．Goyal らの提案する linear scaling rate 法を参考に，ABCI の最大 32 ノード 128GPU を用い，2 種類の学習データセットで評価した結果，ミニバッチサイズと並列数の積が総データサイズの約 7%以下である範囲において，Goyal らの手法が効果的であることが分かった．

Performance Evaluation of Object Detection Algorithm YOLO using Distributed Deep Learning

YURI NISHIKAWA^{1,2,a)} HITOSHI SATO¹ JUN OZAWA¹

1. はじめに

分散深層学習は，複数の計算機を利用し，並列分散処理により深層学習を行うことで，高速に学習モデルを生成できることが期待されている．特に近年，ImageNet のデータセットをはじめとする画像分類タスクを中心に，大規模並列分散環境下での分散深層学習の取り組みが進み，大幅な性能向上に関する事例も報告されている [1][2][3]．分散深層学習は一般に，並列数の増加に伴いバッチサイズが増大することによる汎化性能の低下が課題と言われてきたが [4]，Goyal らの提案する学習率をバッチサイズに比例させる linear scaling rate 法が，汎化性能の低下を解決する手法として，有望視されている [5]．しかしながら，これらの手法は現状，ImageNet や MNIST など，画像分類タスクに適用し評価される場合が主であった．

一方，深層学習ベースの一般物体検出手法の一つとして，YOLO [6][7] が提案されている．YOLO では単一の CNN で分類タスクと領域推定を行い，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．

そこで本研究では，1 枚の画像から複数物体の位置と大きさを検出する一般物体検出 YOLO において，linear scaling rate 法の有効性，また ImageNet 等と比べ小規模なデータセットを用いてどの程度のバッチサイズまで当手法が適用可能かを計算機実験で検証する．本稿の YOLO version2 (以後 YOLOv2) を適用する学習用のデータセットとして，Pascal VOC (学習データ数約 16,000，分類数 20) と，独自データセット (学習データ数約 5,000，分類数 5) の 2 種類を用いる．計算機環境としては，産業技術総合研究所の AI 橋渡しクラウド ABCI[8][9] を用いる．また，YOLOv2 の分散深層学習を ABCI で実施し，並列化とモデルの推定精度を確認するため，GPU の台数を変化させたときの，損失関数の推移を調べる．また，並列化による学習時間を測定し，さらに，実行時の計算コストに関しても試算する．

2. 一般物体検出 YOLO

2.1 YOLO の概要と損失関数

YOLO[6] は深層学習ベースの一般物体検出アルゴリズムの一つである．R-CNN 等の物体検出アルゴリズムでは，物体の分類と領域推定 (region proposal) とが個別に行われており，特に後者で長い処理時間を要していた．一方 YOLO は，単一の CNN で分類と領域推定を同時に行う．このた

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

め、他手法と同等の精度でありながら、コンシューマ向け GPU でリアルタイムに検出が行える．なお、2017 年にはネットワーク構造を変更した YOLOv2[7] が、また 2018 年にはさらに精度向上を図った YOLOv3[10] が公開されている．本研究では、2017 年版の YOLOv2 を用いる．

YOLOv2 では、入力画像を $S \times S$ のグリッドに区切り、グリッド毎に、ある一定のアスペクト比を持つ B 個の矩形領域 (anchor box) の中心座標 (x, y) および幅と高さのスケール (w, h) 、そして矩形内に物体が存在する確率 (confidence) を予測する．さらに、各矩形に何らかの物体が存在するとき、その物体がどのクラスに属するを示す事後確率も予測する．クラスの予測には、VGG ベースの Darknet19 という独自の識別モデルを用いる．

矩形領域、物体の存在確率、クラスの事後確率の予測を、YOLOv2 では一つの損失関数 (loss function) に統合している．矩形領域の中心座標と大きさに関する損失関数を $Loss_{box}$ 、存在確率に関するものを $Loss_{conf}$ 、クラスの事後確率のものを $Loss_p$ としたとき、統合した損失関数を 1 式に示す．

$$Loss = Loss_{box} + Loss_{conf} + Loss_p \quad (1)$$

また、それぞれの損失関数は次式で定義される．

$$Loss_{box} = \lambda_{coord} \sum_i \sum_j^B \mathbf{1}_{ij}^{obj} (x_{ij}^{pred} - x_{ij}^{truth})^2 + (y_{ij}^{pred} - y_{ij}^{truth})^2 + (w_{ij}^{pred} - w_{ij}^{truth})^2 + (h_{ij}^{pred} - h_{ij}^{truth})^2 \quad (2)$$

$$Loss_{conf} = \lambda_{conf}^{obj} \sum_i \sum_j^B \mathbf{1}_{ij}^{obj} \{conf_{ij}^{pred} - iou(box_{ij}^{pred}, box_{ij}^{truth})\}^2 + \lambda_{conf}^{no-obj} \sum_i \sum_j^B \mathbf{1}_{ij}^{no-obj} (conf_{ij}^{pred})^2 \quad (3)$$

$$Loss_p = \lambda_{prob} \sum_i \sum_j^B \mathbf{1}_{ij}^{obj} \{p_{ij}^{pred}(c) - p_{ij}^{truth}(c)\}^2 \quad (4)$$

ここで、添字 $pred, truth$ が付与された変数がそれぞれ予測値、正解を示す．また、 $\mathbf{1}_i^{obj}$ は、何らかの物体がセル i に存在するかどうかを示し (存在すれば 1)、 $\mathbf{1}_{ij}^{obj}$ は、セル i 内の j 番目の矩形候補が、その物体の予測の「担当」であることを示す． $\mathbf{1}_{ij}^{no-obj}$ は $\mathbf{1}_{ij}^{obj}$ における 1 と 0 とが反転する．また、 $iou(A, B)$ は、矩形 A と B の intersection-over-union を表し、 $p(c)$ はセル i 内の j 番目の矩形候補内に物体が存在するとき、それがクラス c である事後確率を表す．また、 $\lambda_{coord} = 1.0$ 、 $\lambda_{conf}^{obj} = 5.0$ 、 $\lambda_{conf}^{no-obj} = 1.0$ (ただし iou が閾値未満のとき)、 $\lambda_{prob} = 1.0$ である．

2.2 画像データを用いた YOLO の学習

YOLOv2 の学習手順を述べる．[7] によれば、まず、ImageNet の学習データを 224×224 の画像サイズに変換し、画像分類を行う Darknet19 モデルで、stochastic gradient descent により学習する．学習率は 0.1、polynomial rate decay は 10^{-4} 、weight decay は 0.0005、momentum は 0.9 に設定し、160 epoch 学習する．この際に、random crop, rotation, color shift (hue, saturation, exposure) 等の一般的なデータオーグメンテーションも行う．次に、画像サイズを 4 倍の 448×448 に設定し、学習率を 10^{-3} 、その他のハイパーパラメータは同一のまま、10epoch だけファインチューニングする．

次に、Pascal VOC などの物体検出に適用可能な学習モデルに変換するため、Darknet19 モデルにおける最後の畳み込み層を取り除き、3 層の 3×3 の畳み込み層 (フィルタ数は 1024) と、1 層の 1×1 の畳み込み層 (フィルタ数は検出用学習データのクラス数) を加えた、YOLOv2 モデルを作成する．そして、Darknet19 モデルによる ImageNet の学習で得られた結果のうち、最初の 18 層分の重みを抽出し、YOLOv2 モデルに適用する．これにより、物体検出用の学習データで、ファインチューニングが可能になる．なお [7] では、Pascal VOC により YOLOv2 モデルで学習するにあたり、epoch 数は 160、学習率は 10^{-3} とし、60epoch と 90epoch で学習率を $1/10$ にしている．その他のハイパーパラメータは Darknet19 と同様である．またこの際、Darknet19 と同様のデータオーグメンテーションを行うと共に、入力画像サイズを変更して学習する multi-scale training も行う．このとき、画像サイズの幅と高さは、それぞれグリッドの幅と高さの倍数になるようにする．

3. 画像分類の分散深層学習における関連研究

一般に深層学習では、学習データを「ミニバッチ」と呼ばれる単位に分割し、この単位で学習モデルの更新を行う．分散深層学習ではプロセスごとにミニバッチ単位の学習データを割り当てるため、一度の並列計算における総バッチサイズ (以後、単にバッチサイズと表記する) は並列プロセス数 \times ミニバッチサイズで求まる．並列数を増やせば、バッチサイズも増え、学習に必要な計算時間が短縮される．一方バッチサイズの増加により、誤差関数の局所解に陥りやすく汎化性能が低下することが報告されており [4]、分散深層学習の課題とされてきた．

解決策として Goyal らが提示した linear scaling rate 法は [5]、バッチサイズが k 倍なら学習率も k 倍に設定することで汎化性能を保てるという経験則であり、120 万枚の ImageNet の学習データ、ResNet-50 モデルに対し、256 台の GPU、バッチサイズ 8,192 で学習を 1 時間で完了できることを示した．Linear scaling rate 法の妥当性は、その後 Smith らにより数理的にも示された [11][12]．また

表 1: 画像検出用の学習データセット

| データセット | クラス数 | 学習データ数 (画像数) |
|------------|------|--------------|
| Pascal VOC | 20 | 16,551 |
| Filmrole | 5 | 4,797 |

ImageNet の学習時間は, Goyal らの手法をベースに学習率を下げるタイミングの変更や勾配降下法の最適化関数を複数併用する等の工夫により 15 分 [3], 層ごとに異なる学習率を設定する Layer-wise Adaptive Rate Scaling (LARS) を適用し, 2,048 個の Xeon Phi CPU を用いて 20 分 [2], さらには LARS, AllReduce 処理の最適化, および FP32 と FP16 の併用により 65,536 のバッチサイズで 6.6 分 [1] と, その記録が更新されている. この他, 本稿でも用いる 5 クラスのデータセットによる, YOLO のクラス識別を行う Darknet19 モデルの学習でも, Goyal らの手法が有効であるとの知見が得られている [13].

4. YOLO の分散深層学習の手法

4.1 基本アプローチ

分散深層学習のためのハイパーパラメータを設定するにあたり, 前節で述べた, Goyal らの linear scaling rate 法を採用した [5]. 例えば, 分散深層学習を適用しない (1 ノードで順次実行する) 時のミニバッチサイズが n , 学習率が η であるとする. k 並列 (例えば k 台の GPU を用いる) で分散深層学習を行う場合, バッチサイズは kn となる. このとき, 学習率を $k\eta$ に設定する.

ただし, ネットワークの重みが大きく変化する学習開始直後は, 特に k の値が大きいき, 学習率も大きくなり, 汎化性能が悪化する. この問題を回避する上で, 最初の 1 epoch の学習率を η とし, 5 epoch 程度をかけて $k\eta$ に到達するよう線形に学習率を増加させることで汎化性能が改善することを, Goyal らが示している. 本稿では, この gradual warmup と呼ばれるテクニックも適用する.

その他のハイパーパラメータ (weight decay 等) は, [7] と同様とする. また, 学習率を 60, 90 epoch 目で $1/10$ にする設定も踏襲する. 学習を開始する時の initial learning rate は, 後述する Pascal VOC データセットのみ 10^{-4} , 独自データセットでは [7] と同じく 10^{-3} に設定する.

4.2 学習データ

本研究では, 画像検出用データセットとして, 表 1 に示すように Pascal Visual Object Classes (VOC)[14][15] と, 独自データセットの 2 種類を用いた. 前者は, コンピュータビジョン分野における画像検出・認識用のベンチマークデータの一つである. 1 枚の写真画像に対し, 人物, 犬, 車など 20 クラスの対象物が映っており, 正解データには, 対象物の位置と大きさを表す矩形領域と, そのクラス ID がアノテーションされている. 1 枚の画像中に含まれる矩形数

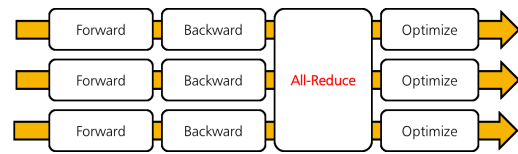


図 1: ChainerMN による分散深層学習

表 2: 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 |

は画像により異なる. なお Pascal VOC コンテストのデータセットは, 2005 年から 2012 年開催分が公開され [16], 本稿では, 2007 年データセットの画像から 5,011 枚, 2012 年から 11,540 枚の合計 16,551 枚を学習用データとする.

後者は, ISSIA CNR が公開するイタリアのサッカー動画 [17] を静止画として切り出し, 人物の位置と大きさを表す矩形領域に対し, 選手, 審判, ゴールキーパーなど 5 つのクラス ID を, 独自でアノテーションしたものである. 1 枚の画像に含まれる矩形領域の数は, Pascal VOC 同様, 画像によって異なる. 学習には, 23,836 個の人物矩形情報を含む, 4,797 枚の画像を用いる. 以後, 本独自データセットを “Filmrole” と記載する.

4.3 分散深層学習フレームワーク

分散深層学習で学習モデルを生成するため, ChainerMN[3] を用いる. また [13] の先行研究と同様, データセットを分散して学習モデルを計算する「データ並列」の手法を採用する. その概要を図 1 に示す. 深層学習では, 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, 勾配を用いて学習モデルを更新する. ChainerMN は図 1 に示すように, 1) 2) を行った後, 複数の計算リソースで分散して計算した勾配から平均を求めて配り直す AllReduce 処理を行う.

5. 評価

5.1 実験環境

本稿における分散深層学習の評価は, 産総研の計算機環境である AI 橋渡しクラウド ABCI 上で行った. 表 2 に計算ノード 1 台のスペックを示す. 計算ノード間は, EDR Infiniband により, Full-bisection Fat Tree 構成で接続されている. 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 7.4, 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 v9.2.88.1, CuDNN v7.1.4 を使用し,

表 3: Pascal VOC の精度 (mAP)

| GPU 台数 | 1 | 16 | 32 | 64 | 128 |
|--------|------|------|------|------|------|
| mAP | 69.3 | 70.2 | 68.6 | 67.3 | 63.3 |

GPU 間の集団通信を行うため NCCL v2.2.13-1 と OpenMPI v2.1.3 を使用している。ChainerMN は v1.2.0 を用い、内部的に Chainer v4.3.0, ChainerCV 0.10.0, CuPy v4.3.0, mpi4py v3.0.0, Python v3.6.5 を使用した。なお、本研究の評価結果は、FP32 によるものである。

5.2 実験結果

5.2.1 バッチサイズと汎化性能

4.1 節で述べた Goyal らの手法を適用した時の、異なるバッチサイズについて、Pascal VOC と Filmrole データセットの epoch 数に対する 1 式で示した *Loss* の変化を、図 2, 図 3 に示す。また Pascal VOC の検証データを用いて計測した、画像検出精度を表す mAP (mean Average Precision) の比較結果を、表 3 に示す。mAP は値が大きいほど良好な結果を示す。ABCI は 1 ノードあたり 4 台の GPU を搭載し、例えば 8 ノードでは 32 台の GPU が使用できる。GPU 数を以後 k で表記する。本研究では、OpenMPI と ChainerMN を用い、GPU 数と等しい k 個のプロセスを生成し、分散処理深層学習を行う。

また 4.1 節で述べたように、ミニバッチサイズが n , k 台の GPU で学習すると、バッチサイズは kn となる。図 2 と図 3 では、 $n = 16$, $k = 16, 32, 64, 128$ に設定している。グラフの青線は、非分散時の結果である。学習率は η で表す。Epoch 数はともに 100 とした。

図 2 が示すように、Pascal VOC では、Goyal らの手法により、いずれも非分散時に沿った学習誤差曲線が得られた。しかしながら、表 3 を見ると、非分散時の mAP が 69.3% に対し 128 台の GPU を用いたときは 63.3% と、汎化性能が劣ることが分かった。このことは、図 2a-図 2c では分散時の学習誤差曲線（オレンジ色）が非分散時の曲線（青色）に数回交差しているのに対し、図 2d では一度も交差していないことから示唆される。なお、64 台、128 台の GPU を用いたときのバッチサイズはそれぞれ 1,024, 2,048 であり、学習データ数の約 6.2%, 12.4% に相当する。

一方 Filmrole では、図 3a, 図 3b において、非分散時と近い学習誤差曲線が得られたが、図 3c で汎化性能が大きく低下し、図 3d では誤差が収束しなかった。また、16GPU を用いた図 3a と、32 台の GPU を用いた図 3b を比較すると、後者では分散時の曲線が非分散時の曲線と一度も交差せず、汎化性能が劣ることが分かった。GPU の台数が 16, 32 のとき、バッチサイズはそれぞれ 256, 512 であり、それぞれ学習データ数の約 5.3%, 10.7% に相当する。以上から、Goyal らの手法のみを用いて Pascal VOC, Filmrole

表 4: Pascal VOC の学習時間とコスト

| GPU 台数 | 1 | 16 | 32 | 64 | 128 |
|----------|--------|--------------|-------|-------|-------|
| 実行時間 [秒] | 33,124 | 3,224 | 1,747 | 987 | 597 |
| コスト | 9.20c | 3.58c | 3.88c | 4.39c | 5.30c |

表 5: Filmrole の学習時間とコスト

| GPU 台数 | 1 | 16 | 32 |
|----------|--------|--------------|-------|
| 実行時間 [秒] | 11,971 | 1,743 | 932 |
| コスト | 3.33c | 1.94c | 2.07c |

データセットにより YOLOv2 モデルで分散深層学習を行う場合、バッチサイズの限界が 5~10% の間にあることが示唆された。なお、120 万の ImageNet の学習データに対し、バッチサイズ 65,536 ので分散深層学習の成功が報告されている [1]。この時の学習データ数に対するバッチサイズの割合は、5.5% である。

5.3 台数効果

次に、図 2, 図 3 を取得したときの、実行時間を計測し、台数効果を評価した。結果を図 4 に示す。横軸は GPU の台数、縦軸は 1 台の GPU を用いた際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。Pascal VOC では GPU が 128 台のとき、Filmrole は 64 台と 128 台のときに、それぞれ汎化性能が低下していることから、それらに関する結果は破線で示した。概ねスケーラブルな処理性能を示すことを確認し、特に Pascal VOC では 64 台の GPU で約 33.6 倍の性能を達成した。

5.4 実行時のコスト

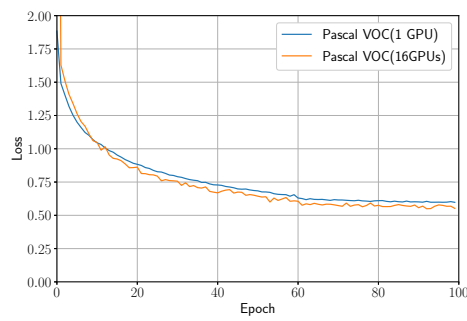
最後に、分散深層学習のコストについて述べる。表 4, 表 5 に、Pascal VOC と Filmrole の 100epoch 分の学習時間とその費用コストを示す。ミニバッチサイズは $n = 16$ とする。費用の計算は、課金単位が 1 ノード 4GPU 単位であるとし、コスト=実行時間/3,600×ノード数× c として求める。 c は 1 時間あたりの 1 ノードの利用金額を表す。

表 4, 表 5 が示すように、どちらの学習データにおいても、非分散時と比べ、分散深層学習を行った方が低コストであることが分かった。分散深層学習を行うことで、学習が短時間で完了し、そのことのコストメリットが高いことが示された。

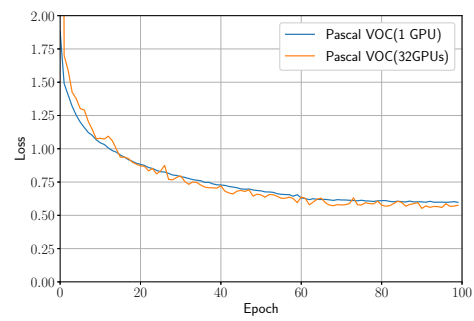
さらに、ノード数 (GPU 数) を増やした時のコストも、学習時間のスケーラビリティに対して緩やかに増えることが分かった。例えば GPU 数を 16 台から 128 台に増やして Pascal VOC の学習を行ったとき、学習時間は約 5.4 倍の高速化となるが、費用は約 1.5 倍であった。

6. まとめ

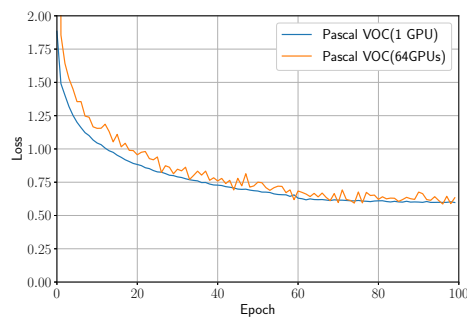
本稿では、産業技術総合研究所の AI 橋渡しクラウド



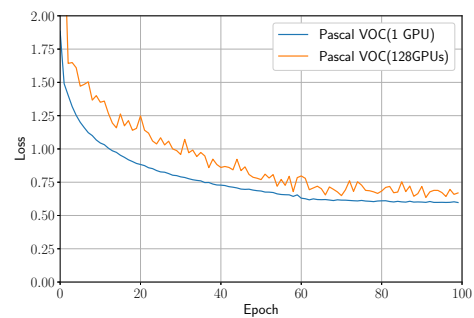
(a) 4 ノード, $k = 16, \eta = 0.0016$



(b) 8 ノード, $k = 32, \eta = 0.0032$

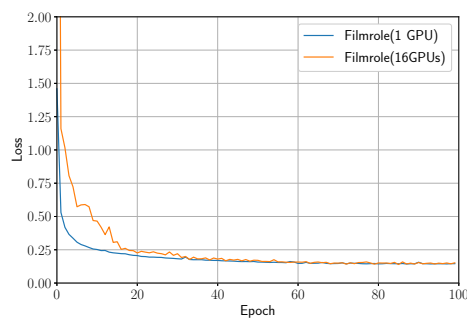


(c) 16 ノード, $k = 64, \eta = 0.0064$

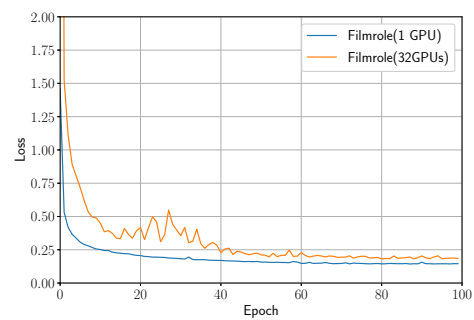


(d) 32 ノード, $k = 128, \eta = 0.0128$

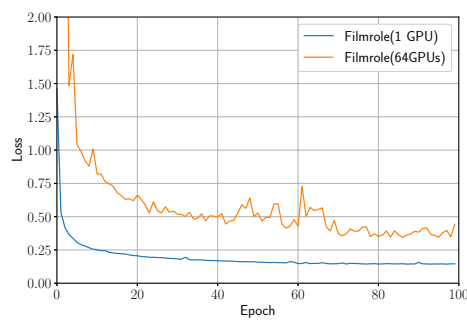
図 2: Pascal VOC の学習誤差曲線 ($n = 16$)



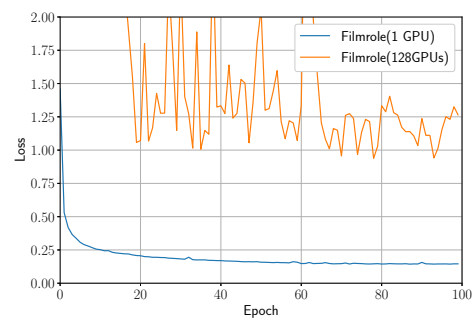
(a) 4 ノード, $k = 16, \eta = 0.016$



(b) 8 ノード, $k = 32, \eta = 0.032$



(c) 16 ノード, $k = 64, \eta = 0.064$



(d) 32 ノード, $k = 128, \eta = 0.128$

図 3: Filmrole の学習誤差曲線 ($n = 16$)

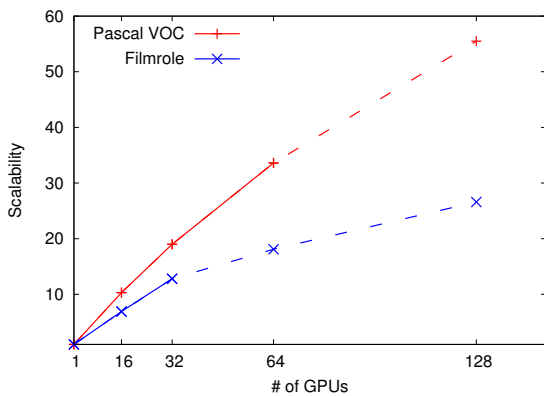


図 4: GPU 数に対するスケーラビリティ

ABCI を用い、一般物体検出手法 YOLOv2 の分散深層学習を行った。Goyal らが提案する、linear scaling rate 法と gradual warmup を適用し、非分散時と比較した学習効果と台数効果を評価した。評価には、Pascal VOC (学習データ数: 約 16,000) と独自データセット (学習データ数: 約 5,000) を用いた。その結果、ミニバッチサイズと並列数の積が総データサイズの 7% 以下の範囲において、非分散時に相当する汎化性能を達成した。Goyal らの手法は実装が容易でありながら、異なる学習モデル、また比較的小規模な学習データに対しても適用可能な、汎用性の高いものであることが示された。また、GPU 台数に対し、概ねスケーラブルな結果を得ることができ、Pascal VOC データセットを 64 台の GPU で学習させた結果、非分散時の 33.6 倍の高速化を確認した。さらには分散深層学習の実行時のコストを計測し、非分散時に対するコストメリットも確認した。

近年は、学習モデルの層毎に、異なる学習率を設定する LARS (Layer-wise Adaptive Rate Scaling) などのテクニックも提唱されており [2]、今後は、本研究で用いた学習モデルや学習データに対する有効性を検証したい。また本稿では FP32 の学習結果を示したが、今後、FP16 に対応した実装と評価も行いたい。

参考文献

- [1] Jia, X., Song, S., He, W., Wang, Y., Rong, H., Zhou, F., Xie, L., Guo, Z., Yang, Y., Yu, L., Chen, T., Hu, G., Shi, S. and Chu, X.: Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes, *CoRR*, Vol. abs/1807.11205 (online), available from <http://arxiv.org/abs/1807.11205> (2018).
- [2] You, Y., Zhang, Z., Hsieh, C.-J., Demmel, J. and Keutzer, K.: ImageNet Training in Minutes, *Proceedings of the 47th International Conference on Parallel Processing, ICPP 2018*, pp. 1:1–1:10 (online), DOI: 10.1145/3225058.3225069 (2018).
- [3] Akiba, T., Suzuki, S. and Fukuda, K.: Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15

- Minutes, *CoRR*, Vol. abs/1711.04325 (online), available from <http://arxiv.org/abs/1711.04325> (2017).
- [4] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. and Tang, P. T.: On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, *International Conference on Learning Representations*, (online), available from <https://openreview.net/forum?id=H1oyRlYgg> (2017).
- [5] 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 (online), available from <http://arxiv.org/abs/1706.02677> (2017).
- [6] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).
- [7] Redmon, J. and Farhadi, A.: YOLO9000: Better, Faster, Stronger, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525 (2017).
- [8] 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).
- [9] ABCI: <https://abci.ai/>.
- [10] Redmon, J. and Farhadi, A.: YOLOv3: An Incremental Improvement, *CoRR*, Vol. abs/1804.02767 (online), available from <http://arxiv.org/abs/1804.02767> (2018).
- [11] Smith, S. L., Kindermans, P.-J., Ying, C. and Le, Q. V.: Don't Decay the Learning Rate, Increase the Batch Size, *International Conference on Learning Representations*, (online), available from <https://openreview.net/forum?id=B1Yy1BxCZ> (2018).
- [12] Smith, S. L. and Le, Q. V.: A Bayesian Perspective on Generalization and Stochastic Gradient Descent, *International Conference on Learning Representations*, (online), available from <https://openreview.net/forum?id=BJij4yg0Z> (2018).
- [13] 佐藤 仁, 西川由理, 小澤 順: 多人数追跡のための分散深層学習による高精度な検出にむけて, *人工知能学会全国大会論文集*, Vol. JSAI2018 (2018).
- [14] Everingham, M., Gool, L., Williams, C. K., Winn, J. and Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge, *Int. J. Comput. Vision*, Vol. 88, No. 2, pp. 303–338 (online), DOI: 10.1007/s11263-009-0275-4 (2010).
- [15] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A.: The Pascal Visual Object Classes Challenge: A Retrospective, *International Journal of Computer Vision*, Vol. 111, No. 1, pp. 98–136 (2015).
- [16] The PASCAL Visual Object Classes Homepage: <http://host.robots.ox.ac.uk/pascal/VOC/>.
- [17] D'Orazio, T., Leo, M., Mosca, N., Spagnolo, P. and Mazzeo, P. L.: A Semi-automatic System for Ground Truth Generation of Soccer Video Sequences, *Proceedings of the 2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS '09*, pp. 559–564 (2009).