Vol.2019-OS-145 No.7
2019/2/28

組み込み機器適用のためのディープラーニングフレームワーク使用リソース調査と評価

出口昌弘[†1]　平森雅裕[†1]　水口武尚[†1]

概要：ディープラーニングが広く利用されはじめており，ディープラーニングを容易に実装するためのディープラーニングフレームワークがオープンソースで公開されている．従来，ディープラーニングフレームワークを用いたディープラーニングは PC 等の高性能なハードウェアで利用されていたが，近年，組み込み機器適用のための検討・開発が行われるようになってきており，組み込み機器向け CPU(SoC)に対応するフレームワークも出てきている．しかし，ディープラーニングの実現では動作させる環境に関する課題が数多く存在する．組み込み機器向けでは，メモリ使用量と，その見積もりに関する課題である．今回，組み込み機器へ学習・評価も含めた処理の適用を検討するため，CNN と手書き文字認識のデータセットである MNIST を用い上記課題の解決策について調査・評価した．この結果，演算の構成を変更することでメモリ使用量を約 11%へ削減し，組み込み機器へ適用できる可能性があることを確認した．また，CPU モードであれば PC 上でのメモリ使用量の結果が組み込み機器向けメモリ使用量の見積もりに活用可能であることを確認した．本調査・評価では，PC と組み込み機器向けハードウェアを想定した Raspberry Pi 3 model B を対象とし，ディープラーニングフレームワークは TensorFlow を用いた．

1. はじめに

近年，ディープラーニングを用いた推論性能が驚異的に向上しており，画像，音声，自然言語，センサデータ等を入力とした認識，識別，判断等処理において，ディープラーニングが広く利用されはじめている．また，このディープラーニングを容易に実装するための仕組みとしてディープラーニングフレームワークが開発され，オープンソースとして公開されてきている．主なディープラーニングフレームワークとして，TensorFlow[1]，Caffe[2]，Pytorch[3](Caffe2[4])，Chainer[5]，CNTK[6]，MXNet[7]がある．ディープラーニングの研究分野ではネットワーク構成や推論での認識率が話題になるが，ディープラーニングを動作させる環境に関する課題が数多く存在する[8].

PC と比較し，組み込み機器はリソースのうち特に搭載可能なメモリ量が限られているという，特有の課題を持つ．このため，本稿では組み込み機器への適用する場合のメモリに関する課題について検討する．

一つは，学習を含めたディープラーニングの実現に関する課題である．組み込み機器は利用可能なメモリが限られており，限られたメモリ範囲内での実現性検討が必要となる．一部のディープラーニングフレームワークでは組み込み機器を考慮した拡張を行っている．例えば，TensorFlow は，学習済みモデルを用いて組み込み機器上で推論が可能な TensorFlow Lite[9]を提供している．しかし，表 1 に示すように現時点では機能やターゲットに制限があるという課題がある．

もう一つは，メモリ使用量の見積もりに関する課題である．表 2 に示すとおり組み込み機器の開発ではハードウェアの開発とソフトウェアの開発が同時に進行することが多く，ソフトウェアの開発・検証段階において利用可能なハードウェアが存在しない場合があることである．このため，開発する機能に対してメモリ使用量の検討が難しい．また，組み込み機器向けのハードウェアは通常すべてのデバイスが基盤に直付けされるため，PC とは異なりメモリ量の変更が難しいという課題もある．このため，メモリ使用量の見積もり誤りは開発そのもののリスクにつながる．ディープラーニングは非常に多くのメモリを使用する場合があるため，正しく見積もることが重要になる．

表 1　TensorFlow と TensorFlow Lite の比較

	TensorFlow	TensorFlow Lite
学習	○	×
推論	○	○
ターゲット	Linux, macOS, Windows, Raspberry Pi	Android, iOS

表 2　PC 向けと組み込み機器向けの開発の比較

	PC 向け開発	組み込み機器向け開発
ハードウェア	あり	開発初期にはないことが多い
メモリ量の変更	可能	難しい

2. 関連研究

ディープラーニングフレームワークを用いた評価や省リソースに関して，いくつかの先行研究がある．例えば，Soheil Bahrampour ら[10]は汎用 PC 上でディープラーニングフレームワークを評価している．X. Zhang ら[11]は汎用 PC と組み込み機器環境で処理時間，最大メモリ使用量，消費電力を評価している．C. Meng ら[12]は GPU のメモリ量

†1 三菱電機株式会社　情報技術総合研究所

制約に対しスワップアウト/インを用いてホストとなる汎用 PC のメモリを活用すること，および，メモリを使用するシーケンスを最適化することで，省メモリ化策を検討している．

3. 研究目的

本研究の目的は，CPU(SoC)の高性能化，低消費電力化により数年後は組み込み機器でもディープラーニングフレームワークを用いて学習や追加学習を実施することも考えられるため，組み込み機器へのディープラーニングフレームワークを適用することを想定してメモリ使用量を調査・評価し，組み込み機器への学習・評価も含めた処理の適用可能性と PC 上での動作との差異有無を確認することにある．

4. 調査と評価

4.1 調査・評価項目

ディープラーニングフレームワークとして組み込み機器レベルの CPU(SoC)に対応した TensorFlow を対象に，処理時間，メモリ使用量，認識率を調査することにより実現性を調査する．

4.2 調査・評価環境

本研究では組み込み機器向けであることを想定し，比較対象となる PC 環境と，組み込み機器向け CPU・メモリを搭載する汎用ボードとを調査・評価対象とした．PC は一般的なものを選択する．汎用ボードは入手性と TensorFlow が正式対応をアナウンスしている[13]組み込み機器レベルの CPU(SoC)を用いた Raspberry Pi 3 model B を選択する．それぞれの環境を表 3 に示す．TensorFlow は，pip(pip3)コマンドによりインストールされるバイナリイメージを用いる．

また，調査・評価では Raspberry Pi 3 model B のメモリ量を考慮し，CNN によるネットワークで MNIST のデータセットを用いた学習・評価・推論を行う．この中で，演算の構成を変更することによりメモリへの影響を調査する．調査・評価に用いたネットワークを表 4 に示す．構成を以下に示す．

- 畳み込み層演算パディング処理有・一括評価
  畳み込み層の演算である 2D コンボリューション演算，および，Max プーリング演算にてパディング処理を実施する．評価は 1 度に全データを用いて実施する．
- 畳み込み層演算パディング処理無・一括評価
  2D コンボリューション演算，および，Max プーリング演算にてパディング処理を実施しない．評価は 1 度に全データを用いて実施する．
- 畳み込み層演算パディング処理無・分割評価
  2D コンボリューション演算，および，Max プーリング演算にてパディング処理を実施しない．評価は 10 分割して実施し，計 10 回の平均値を最終の正解率とする．

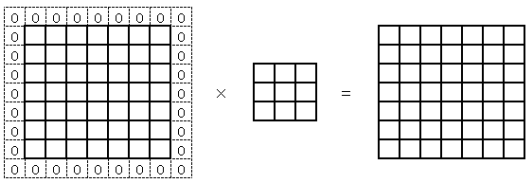
表 3　調査・評価環境

	PC	Raspberry Pi 3 model B
CPU(SoC)	Intel Core i5-6500	Broadcom BCM 2873
アーキテクチャ	Skylake	ARM Coretex-A53
コア数	4	4
周波数	3.2GHz	1.2GHz
メモリ	2GB	1GB
ストレージ	500GB HDD	Class10 32GB microSDHC
OS	Linux	Linux
Distribution	Ubuntu 18.04 LTS	Raspbian Stretch
Kernel	4.15.0-43-generic	4.14.79-v7+
Python	3.6.7	3.5
TensorFlow	1.11.0	1.11.0

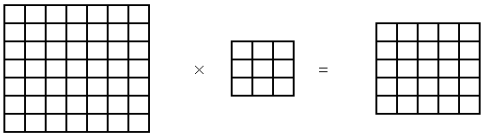
表 4　調査・評価に用いたネットワーク

使用モデル	CNN
使用データセット	MNIST
推論用データ	白色背景画像 10 枚
層構成	畳み込み層 2 層 全結合層 1 層 出力層 1 層
損失関数	交差エントロピー誤差
勾配の最適化手法	Adam
学習時のバッチサイズ	50
学習回数	2000
モード	CPU

畳み込み層の演算におけるパディング有無は，例えば 2D コンボリューションであれば TensorFlow が提供する API である tf.nn.conv2d[14]の第 4 引数にて設定する．Max プーリングパディング有無，同じく TensorFlow が提供する API である tf.nn.max_pool[15]の第 4 引数にて設定する．tf.nn.conf2d を代表例として，動作を図 1 に示す．パディング処理有の場合は padding 引数に SAME を指定し，畳み込み演算で入力と同じ出力サイズを得るため，周囲に 0 をパディングし演算する．演算量が増加するため，メモリ使用量に影響することが想定される．



a) padding="SAME"



b) padding="VALID"

図 1　TensorFlow tf.nn.conv2d

4.3 測定方法

MNIST を用いた学習・評価・推論を実行し，処理時間とメモリの使用量を定期的に取得する．また，評価での認識率を取得する．取得に使用したツールを表 5 に示す．処理時間の測定では Python の line_profiler を用いる. line_profiler は Python で使用するプロファイラであり，今回は，学習・評価･推論の各処理の処理時間算出に利用する. また, Linux の top コマンドは時系列のメモリ使用量を取得するため，1 秒ごとに起床させ情報を取得する．測定はそれぞれ 3 回取得し，処理時間については 3 回の平均値を，メモリ使用量についてはそのうち 1 回の結果を用いる．

表 5　使用したツール

処理時間	line_profiler
メモリ使用量	Linux top コマンド
認識率	TensorFlow API

5. 結果

処理時間，メモリ使用量，認識率について測定した結果を示す．なお，Raspberry Pi 3 model B 上では，「畳み込み層演算パディング有・一括評価」，および，「畳み込み層演算パディング無・一括評価」のパターンは評価時にメモリ不足により「std::bad_alloc」の例外が発生し処理が完遂しなかった．このため，結果は N/A と表記している．

5.1 処理時間

全体の処理時間の測定結果を表 6 に，それぞれの学習・評価･推論の処理時間の測定結果を表 7, 表 8, 表 9 示す．表 6，および，表 9 において「()」で示している割合は PC での同一内容の結果との比較である．全体の時間は学習・評価・推論処理以外の例えば画像の読み込み時間等を含むため，学習・評価・推論の各処理時間の和と全体の処理時間は一致しない．畳み込み層演算パディング無は，畳み込み層演算パディング有と比較して演算量が減少するため処理時間が減少し，今回の場合は半分以下になることとが判明した．また，PC と比較して Raspberry Pi model B は処理時間が約 20 倍増加することが判明した. 処理時間について，表 9 の通り，推論の処理時間は全体と比較して短いが，PC と比較して増加割合が多いことが判明した．

表 6　全体処理時間(単位：秒)

	PC	Raspberry Pi 3 model B
畳み込み層演算パディング有・一括評価	130.17	N/A
畳み込み層演算パディング無・一括評価	60.86	N/A
畳み込み層演算パディング無・分割評価	60.62	1148.51 (1895%)

表 7　畳み込み層演算パディング有・一括評価での各処理時間(単位：秒)

	PC	Raspberry Pi 3 model B
学習	125.55	N/A
評価	3.47	N/A
推論	0.12	N/A
全体	130.17	N/A

表 8　畳み込み層演算パディング無・一括評価での各処理時間(単位：秒)

	PC	Raspberry Pi 3 model B
学習	58.15	N/A
評価	1.62	N/A
推論	0.10	N/A
全体	60.86	N/A

表 9　畳み込み層演算パディング無・分割評価での各処理時間(単位：秒)

	PC	Raspberry Pi 3 model B
学習	57.89	1093.43 (1889%)
評価	1.65	32.16 (1949%)
推論	0.09	10.11 (11233%)
全体	60.62	1148.52 (1895%)

5.2 メモリ使用量

最大メモリ使用量の測定結果を表 10 に，PC での時系列メモリ使用量を図 2 に示す．表 10 において，「()」で示している割合は PC の「畳み込み層演算パディング有・一括評価」との比較であり，「[]」で示している割合は PC の「畳み込み層演算パディング無・分割評価」との比較である．処理時間の測定結果および本測定結果より，メモリは常に最大メモリ使用量付近の量を使用しているのではなく，推論時に多くのメモリを使用することが判明した．一般的な畳み込み層演算パディング有と比較して畳み込み層演算パディング無は演算量が減少するため，最大メモリ使用量が減少することが判明した．また，評価を分割で実施することにより，さらに最大メモリ使用量が大きく減少することが判明した．PC と比較して Raspberry Pi model B は最大メモリ使用量が約 10%減少することが判明した．

表 10　最大メモリ使用量(単位：MB)

	PC	Raspberry Pi 3 model B
畳み込み層演算パディング有・一括評価	5610.50	N/A
畳み込み層演算パディング無・一括評価	2305.02 (41.08%)	N/A
畳み込み層演算パディング無・分割評価	620.57 (11.06%)	573.70 [92.45%]

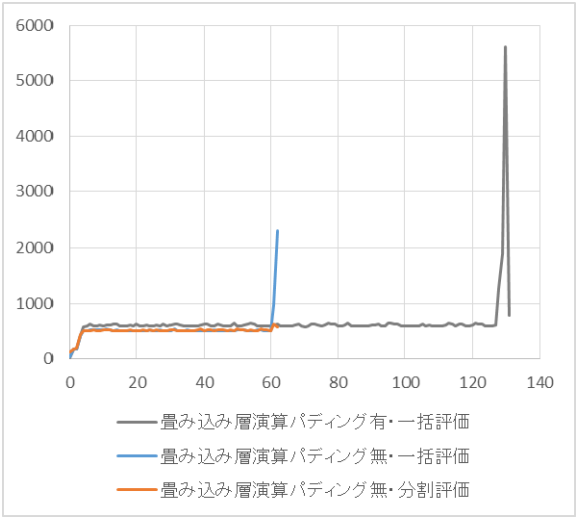
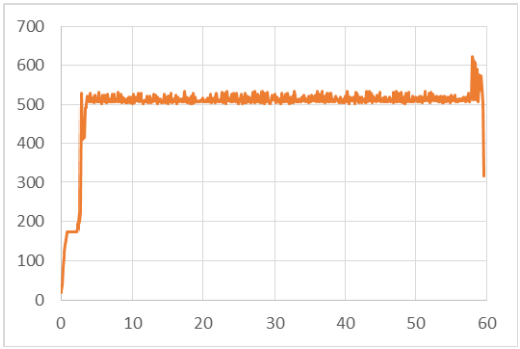
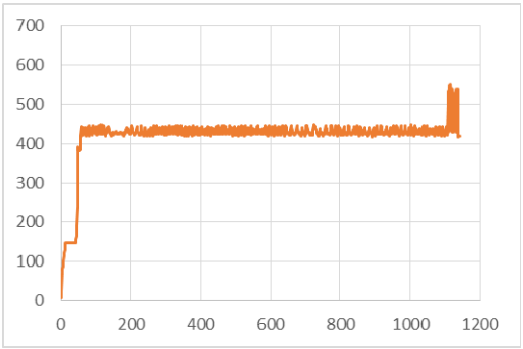


図 2　PC での時系列メモリ使用量
(縦軸：メモリ使用量(単位：MB)，横軸：処理時間(単位：秒))

次に，PC と Raspberry Pi model B とで時系列のメモリ使用量を比較する．PC での時系列のメモリ使用量を詳細に測定するため，Linux の top コマンドを 0.1 秒ごとに起床させ情報を取得し，Raspberry Pi 3 model B でのメモリ使用量と比較した．結果を図 3 に示す．前述の通り PC と比較して Raspberry Pi model B は最大メモリ使用量が約 10%減少する他に，時系列でのメモリ使用量もすべてにおいて約 10%減少しているが，学習・評価・推論におけるメモリ使用量の振る舞いは PC と Raspberry Pi model B で大きな差はないことが判明した．



a) PC



b) Raspberry Pi 3 model B

図 3　畳み込み層演算パディング無・分割評価での時系列メモリメモリ使用量
(縦軸：メモリ使用量(単位：MB)，横軸：処理時間(単位：秒))

5.3 正解率

各構成にて正解率が大きく変化したかの確認を目的に，評価にて出力された正解率の平均値を表 11 に示す．それぞれにおいて大きな差異はないことが判明した．

表 11　正解率

	PC	Raspberry Pi 3 model B
畳み込み層演算パディング有・一括評価	0.9768	N/A
畳み込み層演算パディング無・一括評価	0.9769 (0.01%)	N/A
畳み込み層演算パディング無・分割評価	0.9781 (0.13%)	0.9749 (-0.19%)

6. 考察

6.1 メモリ使用量について

畳み込み演算のパディング処理を目的に合わせ必要に応じて変更することにより最大のメモリ使用量を 58.92%減少させ，さらに評価を分割することにより上記とあわせて88.94%減少させることができた．前者は演算量の減少に伴い演算に用いられるメモリ使用量もあわせて減少したものと考える．また，評価は分割することにより 1 回あたりのメモリ使用量が減少したためだと考える．

GPU を使用しない場合，メモリ使用の振る舞いは PC と Raspberry Pi 3 model B で大きく差はないため, PC での評価結果が組み込み機器向けでの検討に活用可能である．しかし，メモリ使用量は双方で異なる結果となった．これは，CPU アーキテクチャや利用可能な命令セットに依存している可能性がある．バイナリ版の TensorFlow は AVX のみに対応しており，AVX 非対応やその他命令に対応した環境で調査・評価するためには，TensorFlow のソースコードからビルドする必要がある．

6.2 正解率について

畳み込み演算のパディングを実施しない場合，畳み込み演算にて画像端の情報がその他の個所の情報と比較して利用されないため，画像端の情報(画素)が考慮されづらい．例えば図 1における画像の 4 隅を用いた畳み込み演算について，パディング有の場合は 4 回の畳み込み演算で利用されるが，パディング無の場合は 1 回の畳み込み演算でのみ利用される．最大 9 回の畳み込み演算で利用されるため，4 隅の情報は次の層へ影響しにくくなる．

今回は MNIST を用いて評価を実施した．MNIST は手書き数字の認識であるため，画像の端に重要な情報が付与されることが少ない．このため，今回はパディング有と無で正解率に大きな差が発生しなかったと考える．

6.3 組み込み機器向けについて

以上より，組み込み機器で用いるにはメモリ使用量の軽減策が必要であり，パディング無や評価の分割により軽減が可能である．しかし，パディング無とする場合．実際の対象を考慮して決定する必要がある．また，GPU を使用しない CPU モードであれば組み込み機器向けのメモリ使用量検討に PC の結果を活用することが可能である．

7. おわりに

今回，組み込み機器へ学習・評価も含めた処理の適用可能性を検討するため，TensorFlow 上でディープラーニングモデルの CNN と手書き文字認識のデータセットである MNIST を用いてリソースのうちメモリについて使用量を調査し，

- 演算の構成を変更することによるメモリ使用量と認識率への影響
- PC 上でのメモリ使用量の結果が組み込み機器向けの検討で活用可能であるか

を調査・評価した．調査には，PC と組み込み機器向けハードウェアを想定した Raspberry Pi 3 model B を用いた．その結果，メモリ使用量は評価段階で最大値となり，評価の工夫によりその最大値を約 89%減少させることが可能であると判明した．また，PC 上と Raspberry Pi 3 model B 上ではメモリ使用量の絶対値は異なるが挙動は同一であるため，GPU を使用しない CPU モードであれば組み込み機器向けメモリ使用量の見積もりに PC の結果を活用することが可能であることが判明した．

PC と Raspberry Pi 3 model B で異なるメモリ使用量の調査は，TensorFlow をソースコードからビルドして挙動を把握する必要あり，この原因の調査は今後の課題である．また，今回の調査・評価では PC と Raspberry Pi 3 model B では同じ CPU コア数であったが, これが異なる場合についても調査し挙動を確認しておく必要がある．

今後は，メモリ以外の使用リソースについても調査・評価し，組み込み機器向けディープラーニングのリソース検討の仕組みを確立する．

参考文献

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu and Xiaoqiang Zheng,: TensorFlow: A System for Large-Scale Machine Learning, *Proc. of Operating Systems Design and Implementation*, pp. 265-283 (2016).

[2] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama and Trevor Darrell,: Caffe: Convolutional Architecture for Fast Feature Embedding, *Proc. of Proceedings of ACM Multimedia*, pp. 675–678 (2014).

[3] Facebook: PyTorch, https://pytorch.org/.

[4] Facebook: Caffe2, https://caffe2.ai/.

[5] Seiya Tokui, Kenta Oono, Shohei Hido and Justin Clayton,: Chainer: a next-generation open source framework for deep learning, *Proc. of Workshop on Machine Learning Systems in NIPS*, 2015.

[6] Microsoft: CNTK, https://www.microsoft.com/en-us/cognitive-toolkit/.

[7] Apache: MXNet, https://mxnet.apache.org/.

[8] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Franc¸ois Crespo and Dan Dennison,: Hidden Technical Debt in Machine Learning Systems, *Proc. of the 28th International Conference on Neural Information Processing Systems - Volume 2*, pp. 2503-2511 (2015).

[9] Google: TensorFlow Lite, https://www.tensorflow.org/lite/.

[10] Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott and Mohak Shah,: Comparative Study of Deep Learning Software Frameworks, https://arxiv.org/abs/1511.06435.

[11] Xingzhou Zhang, Yifan Wang and Weisong Shi,: pCAMP: Performance Comparison of Machine Learning Packages on the Edges, *Proc. of the USENIX Workshop on Hot Topics in Edge Computing*, 2018.
[12] Chen Meng, Minmin Sun, Jun Yang, Minghui Qiu and Yang Gu,: Training Deeper Models by GPU MemoryOptimization on TensorFlow, *Proc. of ML Systems Workshop in NIPS*, 2017.
[13] TensorFlow - Install TensorFlow, https://www.tensorflow.org/install/
[14] TensorFlow - API tf.nn.conv2d, https://www.tensorflow.org/api_docs/python/tf/nn/conv2d
[15] TensorFlow - API tf.nn.max_pool, https://www.tensorflow.org/api_docs/python/tf/nn/max_pool