Vol.2018-ARC-230 No.30
Vol.2018-SLDM-183 No.30
Vol.2018-EMB-47 No.30
2018/3/8

組込み機器に向けた Caffe の性能評価

平森将裕[†1]　攝津敦[†1]

概要：ディープラーニングは画像処理や音声処理，自然言語処理等に広く利用されているが，高い演算性能と大容量のメモリが必要となるため，リソース制限が厳しい組込み機器では性能が課題となる．そこで今回，組込み機器におけるディープラーニングの推論性能の改善指針を得るべく，汎用 PC と組込み機器である Raspberry Pi 3，R-Car スタータキット Premier の 3 つの機器上で，ディープラーニングのフレームワークである Caffe の推論処理におけるリソース使用量を測定し，比較・評価を行った．

本評価では，メモリ帯域，メモリ使用量，CPU の使用コア数，GPU の観点で Caffe の推論処理を評価した．その結果，組込み機器においては，モデルのコンパクト化が重要であり，大規模なモデルにおいてはメモリ帯域が推論時間に影響を与えることが分かった．一方，小規模なモデルにおいては CPU より GPU の方が推論時間が長く，この原因の調査については今後の課題である．

キーワード：ディープラーニング，ニューラルネットワーク，画像認識，Caffe，GPU，Raspberry Pi，R-Car

Performance Evaluation of Deep Learning Framework “Caffe” for Embedded Devices

MASAHIRO HIRAMORI[†1]　ATSUSHI SETTSU[†1]

Abstract: Since deep learining has been used for a wide variety of tasks such as image recogniton, speech recognition, and natural language processing, it remains a challenge to use them in resource-constrained devices with limited computational power and memory capacity. In this paper, we present our attempt to evaluate the performance of deep learning framework “Caffe” on embedded devices. We focus on evaluating the inference time and memory consumption of Caffe with four popular types of network models on three different types of hardware platforms. According to our results, we find that large scale models is much slower than small scale models in view of computational power differences. And inference time of large scale models on CPU is affected by memory bandwidth. It indicates that compacting network models is a key factor for embedded devices. However, inference time of small scale models on GPU is slower than that of CPU. Our evaluation can be a reference to improving performance of deep learning on embedded devices.

Keywords: Deep Learning, Neural Network, Image Recognition, Caffe, GPU, Raspberry Pi, R-Car

1. はじめに

ディープラーニングは画像処理から音声処理，自然言語処理等の分野において，驚異的な性能を実現している．例えば画像処理においては，1000 クラスの物体画像識別コンペティションである ILSVRC2012 では，ディープラーニングを用いたシステムにより，従来の特徴量ベースの手法と比べて認識エラー率が 10 ポイント以上減少した[1]. また，音声処理においては，電話会話音声の認識において，従来は 30%程度であった認識エラー率が，ディープラーニングを用いることで 20%以下の認識エラー率を実現した[2].

しかし，ディープラーニングの計算量は，LeNet[3]では 34.1 万回，AlexNet[1]では 7 億 2400 万回，GoogLeNet[4]では 1430 億回の積和演算が必要であり，膨大な計算量が必要となる．また，ネットワークの重みパラメータは，LeNet では 6 万個，AlexNet では 6100 万個，GoogLeNet では 700 万個必要なため，膨大なメモリ量が必要となる[5]．そのため，リソース制限の厳しい組込み機器で利用するには性能が課題となる．

そこで今回は，ディープラーニングフレームワークの一つである Caffe の推論処理におけるリソース使用量を測定し，評価・比較を行う．

Caffe **エラー！参照元が見つかりません。**は Berkeley Vision and Learning Center(BVLC)が中心となって開発を進めているディープラーニングフレームワークである．C++で実装されており，既存の C++プログラムとの統合が可能である．Python や MATLAB とのインタフェースも用意されているため，手軽に使うことが可能である．マルチスレッド処理と GPU に対応しているため，高速な処理が可能である．モデルの定義は設定ファイルに記述するため，再利用が容易である．また，Model Zoo から学習済みモデルを取得できるため，手軽に試すことができる．

2. 関連研究について

ディープラーニングフレームワークの性能評価として，いくつかの先行研究がある．例えば，Soheil Bahrampour ら[7]は汎用 PC 上でディープラーニングフレームワークの性

†1 三菱電機株式会社　情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation

Vol.2018-ARC-230 No.30
Vol.2018-SLDM-183 No.30
Vol.2018-EMB-47 No.30
2018/3/8

能評価を行っており, Pengfei Xu ら[8]は汎用 PC 上の Docker コンテナでのディープラーニングフレームワークの性能評価を行っている.

3. 研究目的

本研究の目的は，メモリ帯域，メモリ使用量，CPU の使用コア数，GPU の観点で Caffe の推論処理を評価することで，組込み機器におけるディープラーニングの推論性能の改善指針を得ることである.

4. 評価方法

本評価では, 汎用 PC と 2 つの組込み機器を対象とした. 消費電力やコストの観点から GPU が使えない場合を想定して Raspberry Pi 3 を選定し，GPU が使える場合を想定して R-Car H3 を選定した. また，評価用プログラムは既存の組込みシステムとの統合を考え，Python や MATLAB ではなく C++で作成した.

4.1 評価項目

4 つの異なる規模のモデルにおける推論処理のモデル読込み時間，モデル計算時間，メモリ使用量を比較することで, モデルの規模がリソース消費量に与える影響を調べる. なお，推論処理はモデル読込み，画像読込み，モデル計算の 3 つの処理から構成されるが，今回は Caffe のリソース消費量が測定対象のため，OpenCV の機能で行う画像読込みは測定しない.

4.2 評価対象

推論処理の評価に用いるネットワークモデルを表 1 に示す.

表 1 評価に用いるモデル

項目	LeNet	CIFAR-10	GoogLeNet	AlexNet
畳込み層の深さ	2	3	21	5
畳込み層の数	2	3	57	5
全結合層の数	3	2	1	3
重みパラメータ	431k	145k	7M	61M
モデルのファイルサイズ	1.7MB	583KB	53MB	243MB
積和演算回数	341k	124M	16G	7.3G
比較回数	974k	942k	160M	17.7M
加算回数	640	277k	8.54M	4.78M
除算回数	640	100	16.1M	9.55M
指数演算回数	640	100	8.04M	4.78M

小規模なモデルとして LeNet と CIFAR-10[9][10], 大規模なモデルとして GoogLeNet と AlexNet を用いる.

LeNet と CIFAR-10 に関しては，公式チュートリアル[11][12]を元に事前に学習させたモデルを使用する. GoogLeNet と AlexNet に関しては，BVLC が提供する事前学習済みモデル[13][14]を使用する.

4.3 評価環境

評価に用いるハードウェア環境とソフトウェア環境を表 2 に示す.

表 2 ソフトウェア・ハードウェア環境

機器	汎用 PC	Raspberry Pi 3	R-Car スタータキット Premier
CPU	Intel Core i5-6500	Broadcom BCM2837	Renesas R-Car H3
アーキテクチャ	Intel Skylake	ARM Cortex-A53	ARM Cortex-A57
コア数	4	4	4
周波数	3.2 GHz	1.2 GHz	1.5 GHz
主記憶	32GB	1GB	4GB
ストレージ	1TB SSD	32GB microSDHC,Class10	
OS	Debian 8 Jessie	Raspbian Jessie[15]	Yocto-Gen3-ADAS[16]
Linux カーネル	4.9.0-0.bpo.5-amd64	4.9.35-v7	4.9.0-yocto-standard
Caffe	e963ef4e		e0f77c3b
OpenCV	2.4.9.1+dfsg-1+deb8u1		2.4.11

4.4 測定方法

汎用 PC と Raspberry Pi 3 上では，1 コア使用と 4 コア使用でそれぞれ測定し，マルチコア化による処理速度の変化を調べる．R-Car H3 上では，CPU4 コア使用と GPU 使用でそれぞれ測定し，GPU 化の影響を調べる.

Raspberry Pi 3 上での使用コア数の変更には，Caffe が使用している行列演算ライブラリである OpenBLAS[17]の環境変数「OPENBLAS_NUM_THREADS」を書き換えることで実現する.

R-Car スタータキット Premier 上での CPU/GPU 使用の変更は，Caffe::set_mode 関数を評価用プログラムのソースコード内で用いて変更する.

処理時間の測定には，C++の標準ライブラリ std::chrono[18]を使用して 11 回測定し, 1 回目の測定を除く 10 回の平均値を算出する．1 回目の測定を除いた理由は，重みパラメータファイルがキャッシュされた状態で測定することでディスク I/O 性能の影響を減らすためである.

メモリ使用量の測定には，Linux が提供する proc ファイルシステム[19]の最大物理メモリサイズを示す VmHWM を用いてプロセスのメモリ使用量を測定する.

Vol.2018-ARC-230 No.30
Vol.2018-SLDM-183 No.30
Vol.2018-EMB-47 No.30
2018/3/8

5. 結果

本章は，本評価の結果について述べる．5.1 項にて推論時間, 5.2 項にてメモリ使用量の評価結果をそれぞれ示す．なお, Raspberry Pi 上では AlexNet がメモリ不足により動作しなかったため，表にはその旨を記載している．

5.1 推論時間

モデル読込み時間の測定結果を表 3 に示す．

表 3 モデル読込み時間の測定結果(ms)

対象機器	LeNet	CIFAR-10	GoogLeNet	AlexNet
汎用 PC (CPU/1 コア)	5.8	3.4	112.4	420.6
汎用 PC (CPU/4 コア)	6.2	3.5	117.5	425.9
Raspberry Pi 3 (CPU/1 コア)	59.2	28.0	972.9	メモリ不足
Raspberry Pi 3 (CPU/4 コア)	54.7	26.8	980.7	メモリ不足
R-Car (CPU/ 4 コア)	25.4	17.1	471.2	1171.4
R-Car (GPU)	79.7	69.4	727.6	1239.2

汎用 PC と Raspberry Pi 3 の結果を見ると，コア数が 1 コアから 4 コアに増えてもモデル読込み時間は変わらないことが分かった．R-Car の結果を見ると，CPU から GPU へ変えることでモデル読込み時間が 52.4～256.4ms 増えることが分かった．これについては考察で述べる．

モデル計算時間の測定結果を表 4 に示し，汎用 PC(CPU/1 コア)のモデル計算時間を 1 としたときの組込み機器におけるモデル計算時間を図 1 に示す．

表 4 モデル計算時間の測定結果(ms)

対象機器	LeNet	CIFAR-10	GoogLeNet	AlexNet
汎用 PC (CPU/1 コア)	3.2	4.9	567.1	213.2
汎用 PC (CPU/4 コア)	3.2	4.6	486.4	179.8
Raspberry Pi 3 (CPU/1 コア)	17.4	40.6	7198.4	メモリ不足
Raspberry Pi 3 (CPU/4 コア)	15.5	25.8	5359.9	メモリ不足
R-Car (CPU/ 4 コア)	15.3	15.4	2117.5	1428.6
R-Car (GPU)	52.0	31.7	1063.4	466.4

Raspberry Pi 3 の結果を見ると，1 コアから 4 コアに増やすと LeNet で 10.8%，CIFAR-10 で 36.5%，GoogLeNet で 25.5%のモデル計算時間が削減され，モデルごとに削減率が異なることが分かった．これについては考察で述べる．

R-Car の結果を見ると，CPU と GPU の比較では，LeNet と CIFAR-10 だと GPU の方が遅く，GoogLeNet と AlexNet だと CPU より GPU の方が速いことが分かった．これについては考察で述べる．

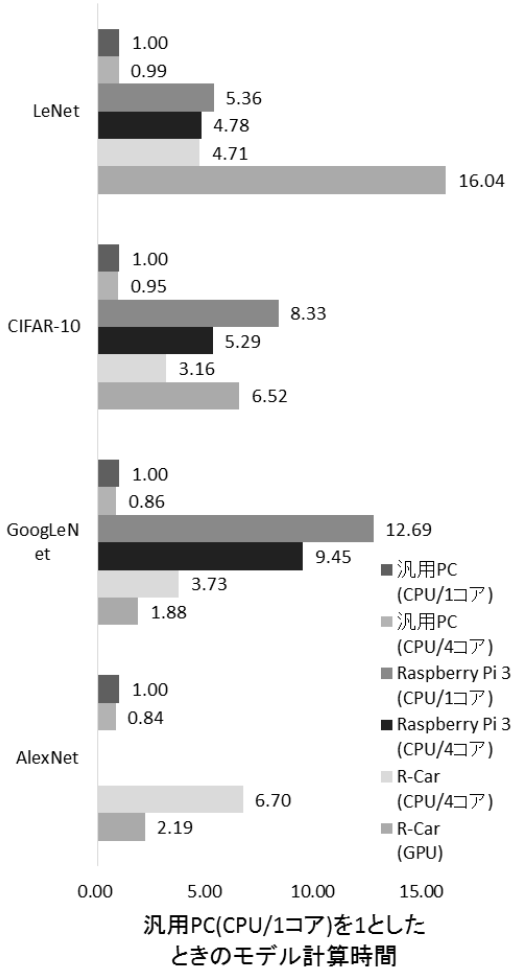

図 1 汎用 PC と組込み機器のモデル計算時間の比較

5.2 メモリ使用量

表 5 にメモリ使用量の測定結果を示す．

汎用 PC に比べて Raspberry Pi 3 の方がメモリ使用量が約 4.5MB 少ないことが分かった．これは，x86 と ARM のアーキテクチャの違いであると考える．AlexNet は汎用 PC 上で 980MB のメモリを使用しているため，Raspberry Pi 3 上では約 975MB 使用すると考えられる．Linux の free コマンドで Raspbery Pi 3 の起動時の空きメモリ容量を調べると 804MB であった．したがって，AlexNet はメモリ不足で動作しなかったと推測できる．

Raspberry Pi 3 の結果を見ると，コア数が 1 コアから 4 コアに増えてもメモリ使用量は大きく変わらないことが分か

Vol.2018-ARC-230 No.30
Vol.2018-SLDM-183 No.30
Vol.2018-EMB-47 No.30
2018/3/8

った．モデルデータはコアで共通の情報であるからだと考える．

R-Car の結果を見ると，CPU と GPU の比較では，GPU の方がメモリを多く使用することが分かった. これは, GPU のコンテキスト等のデバイス情報をメインメモリに確保するためであると考える．

AlexNet は GoogLeNet に比べて重みパラメータの数は 8.7 倍であるにもかかわらず，AlexNet は GoogLeNet の 2 倍のメモリ使用量となっており，重みパラメータの数に比例してメモリ使用量が増えるわけではないことが分かった．

表 5　メモリ使用量の測定結果(KB)

対象機器	LeNet	CIFAR-10	GoogLeNet	AlexNet
汎用 PC (CPU/1 コア)	37,756	39,980	450,332	981,728
汎用 PC (CPU/4 コア)	37,772	39,744	450,584	982,228
Raspberry Pi 3 (CPU/1 コア)	32,676	34,200	445,140	メモリ不足
Raspberry Pi 3 (CPU/4 コア)	32,648	34,740	445,600	メモリ不足
R-Car (CPU/ 4 コア)	44,080	38,452	513,600	970,236
R-Car (GPU)	47,776	45,152	516,936	988,464

6. 考察

本章では，5 章で得られた結果の考察について述べる．6.1 節にて CPU における 1 コアと 4 コアの比較について述べ，6.2 節にて CPU と GPU の比較について述べる．

6.1 1 コア/4 コア比較

演算性能に対するメモリバンド幅を調べるために，汎用 PC，Raspberry Pi 3，R-Car における CPU の浮動小数点演算性能とメモリバンド幅を測定した．浮動小数点演算性能は UnixBench[20]で測定し，メモリバンド幅は STREAM[21]で測定した．測定結果を表 6 に示す．

表 4 と表 6 をもとに，汎用 PC(CPU/1 コア)と汎用 PC(CPU/4 コア)をそれぞれ 1 としたときのモデル計算時間と浮動小数点演算実行時間，メモリ転送時間の比較を表 7 と表 8 に示す．

表 7 で汎用 PC と Raspberry Pi 3 を比較すると，LeNet では演算性能の差だけモデル計算時間が増えているが，GoogLeNet ではメモリ帯域の差だけモデル計算時間が増えている．表 8 を見ると，4 コアの場合でも同様の結果である．しかし，R-Car の結果を見ると，GoogLeNet におけるモデル計算時間の増加は Raspberry Pi 3 ほど大きくない. これは, 浮動小数点演算性能に対するメモリ帯域が, Raspbery Pi 3 に比べて R-Car の方が高いためであると考える．したがって，CPU においては，小規模なモデルでは浮動小数点演算性能が，大規模なモデルではメモリ帯域がモデル計算時間に影響を与えるといえる．

表 6　浮動小数点演算性能とメモリバンド幅の測定結果

機器	浮動小数点演算性能 [MWIPS]	メモリコピー速度[MB]
汎用 PC (CPU/1 コア)	4267	18352.8
汎用 PC (CPU/4 コア)	18671	18537.8
Raspberry Pi 3 (CPU/1 コア)	737	2193.6
Raspberry Pi 3 (CPU/4 コア)	2944	2165.1
R-Car (CPU/4 コア)	2899	5258.8

表 7　汎用 PC(CPU/1 コア)を 1 としたときのモデル計算時間と浮動小数点演算時間，メモリ転送時間の比較

項目	LeNet	CIFAR-10	Goog LeNet	浮動小数点演算実行時間	メモリ転送時間
汎用 PC (CPU/1 コア)	1.0	1.0	1.0	1.00	1.00
Raspberry Pi 3 (CPU/1 コア)	5.4	8.3	12.7	5.79	8.56

表 8　汎用 PC(CPU/4 コア)を 1 としたときのモデル計算時間と浮動小数点演算時間，メモリ転送時間の比較

項目	LeNet	CIFAR-10	Goog LeNet	浮動小数点演算実行時間	メモリ転送時間
汎用 PC (CPU/4 コア)	1.0	1.0	1.0	1.00	1.00
Raspberry Pi 3 (CPU/4 コア)	4.8	5.6	11.0	6.34	8.56
R-Car (CPU/4 コア)	4.7	3.3	4.4	6.44	3.53

6.2 CPU/GPU 比較

6.2.1 モデル読込み時間

R-Car の CPU と GPU におけるモデル読込み時間の差について考察する．

モデル読込み時には，GPU を初期化し，モデルデータをメインメモリへ読込む．したがって，モデル読込み時間は

Vol.2018-ARC-230 No.30
Vol.2018-SLDM-183 No.30
Vol.2018-EMB-47 No.30
2018/3/8

モデル層数とモデルのファイルサイズに依存すると考えられる．

モデル読込み時間とモデル層数，モデルのファイルサイズを比較した結果を表 9 に示す．

表を見ると，モデル読込み時間はモデルのファイルサイズではなく，モデルの層数に依存していることが分かる．これは，Caffe では各層ごとに層の作成，パラメータの設定を行うことが理由である考える．

表 9　R-Car(GPU)におけるモデル読込み時間とモデルのファイルサイズ，モデルの層数の関係

項目	LeNet	CIFAR-10	GoogLeNet	AlexNet
R-Car(CPU/4 コア)との読込み時間の差	54.3	52.4	256.4	67.8
モデルのファイルサイズ	1.7MB	583KB	53MB	243MB
サイズ 1MB あたりの読込み時間の差(ms)	2.94	8.62	0.42	0.03
モデルの層の深さ	5	5	22	8
1 層あたりの読込み時間差	10.9	10.5	11.7	8.5

6.2.2　モデル計算時間

R-Car の CPU と GPU におけるモデル計算時間の差について考察する．

汎用 PC(CPU/4 コア)を 1 としたときのモデル計算時間と浮動小数点演算実行時間，メモリ転送時間の比較を表 10 に示す．

表 10　汎用 PC(CPU/4 コア)を 1 としたときのモデル計算時間と浮動小数点演算時間，メモリ転送時間の比較

項目	LeNet	CIFAR-10	GoogLeNet	AlexNet	演算時間	メモリ転送時間
汎用 PC (CPU/4 コア)	1.0	1.0	1.0	1.0	1.00	1.00
R-Car (CPU/4 コア)	4.7	3.3	4.4	7.9	6.44	3.53
R-Car (GPU)	16.2	6.8	2.2	2.6	-	

GoogLeNet や AlexNet のような大規模なモデルにおいては，CPU より GPU の方がモデル計算時間が短い結果となった．しかし，LeNet や CIFAR-10 のような小規模なモデルにおいては，CPU より GPU の方が遅く，かならずしも GPU を使うとモデル計算時間が短縮されるとは限らないことが分かった．これは，小規模なモデルにおいては GPU 実行によるモデル計算時間の短縮に比べて，GPU 実行のオーバーヘッドが大きくなることが原因であると推測する．しかし，この原因については今後，さらに調査する必要がある．

7.　おわりに

今回，ディープラーニングフレームワークである Caffe の推論処理における処理時間とメモリ使用量を測定し，評価・比較を行った．評価は，推論処理におけるモデル読込み時間およびモデル計算時間について，4 つのモデルを対象に行った．

モデル読込み時間は，使用 CPU コア数を 1 コアから 4 コアへ変化させても差は見られなかったが，CPU と GPU では差が見られた．これは，読込み時の GPU 初期化時間が原因だと考える．また，モデルの層数に比例してモデル読込み時間が長くなることが分かった．

モデル計算時間は，使用 CPU コア数を 1 コアから 4 コアへ変化させると，小規模なモデルでは浮動小数点演算性能，大規模なモデルではメモリ帯域の影響を受けることが分かった．しかし，小規模なモデルにおいては CPU より GPU の方がモデル計算に時間がかかることが分かった．この原因の調査については今後の課題である．

メモリ使用量は，コア数が 1 コアから 4 コアに増えても大きく変わらなかったが，CPU から GPU へ変更すると増加した．これは，GPU のデバイス情報をメモリ上に確保するためだと考える．また，重みパラメータの数に比例してメモリ使用量が増えるわけではないことが分かった．

本結果から，メモリ帯域の狭い組込み機器におけるディープラーニングでは，モデルのコンパクト化が重要であり，大規模なモデルを使用する場合には，メモリ帯域が広い SoC を選択することが重要であるといえる．

今後は，ディープラーニングに適した組込みシステムのアーキテクチャ設計と，組込みシステムに適したディープラーニングのアルゴリズムを開発する．

8.　参考文献

[1]Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, “Imagenet classification with deep convolutional neural networks”, Advances in Neural Information Processing Systems, 2012.

[2]F. Seide, Frank Seide, Gang Li and Dong Yu, “Conversational Speech Transcription Using Context-Dependent Deep Neural Networks”, INTERSPEECH, 2011.

[3]Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition”, Proc. IEEE 86(11):2278–2324, 1998.

[4]Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich, “Going Deeper With Convolutions”, CVPR, 2015.

[5]Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang and Joel Emer, ”Efficient

Vol.2018-ARC-230 No.30
Vol.2018-SLDM-183 No.30
Vol.2018-EMB-47 No.30
2018/3/8

Processing of Deep Neural Networks:A Tutorial and Survey”, CoRR, 2017.
[6]Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama and Trevor Darrell, “Caffe: Convolutional Architecture for Fast Feature Embedding”, CoRR, 2014.
[7]Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott and Mohak Shah, “Comparative Study of Deep Learning Software Frameworks”, CoRR, 2015.
[8]Pengfei Xu, Shaohuai Shi and Xiaowen Chu, “Performance Evaluation of Deep Learning Tools in Docker Containers”, CoRR, 2017.
[9]Caffe: https://github.com/BVLC/caffe/blob/master/examples/cifar10/cifar10_quick.prototxt
[10]CIFAR-10 and CIFAR-100 datasets: http://www.cs.toronto.edu/~kriz/cifar.html
[11]Caffe | LeNet MNIST Tutorial: http://caffe.berkeleyvision.org/gathered/examples/mnist.html
[12]Alex’s CIFAR-10 tutorial, Caffe style: http://caffe.berkeleyvision.org/gathered/examples/cifar10.html
[13]caffe/models/bvlc_googlenet: https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet
[14]caffe/models/bvlc_alexnet: https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet
[15]Raspberry Pi: https://www.raspberrypi.org/blog/raspbian-jessie-is-here/
[16]R-Car/Boards/Yocto-Gen3-ADAS - eLinux.org: https://elinux.org/R-Car/Boards/Yocto-Gen3-ADAS
[17]OpenBLAS: http://www.openblas.net/
[18]Date and time utilities - cppreference.com: http://en.cppreference.com/w/cpp/chrono
[19]proc(5) — Linux manual pages: http://manpages.courier-mta.org/htmlman5/proc.5.html
[20]UnixBench: https://github.com/kdlucas/byte-unixbench
[21]STREAM: http://www.cs.virginia.edu/stream/