センサーネットワーク内深層学習MicroDeepの実装と評価

福島 悠太[1,a)]　山口 弘純[1,b)]　東野 輝夫[1,c)]

概要：マイコンの高機能化・省電力化およびセンサーの小型化にともない，各センサーが一定のデータ処理能力と通信能力を備え，センシングデータを無線でリアルタイム集約する無線センサーネットワーク（Wireless Sensor Network, WSN）の現実性が高まりつつある．しかし，WSN においては，現場に設置される WSN からの継続的なセンシングデータを集約するための回線保持の必要性やクラウドにデータを集約することによるプライバシーの懸念などの課題がある．これに対し，我々の研究グループでは，従来クラウドで行っていたデータ学習や異常検出，判定などのタスク処理をセンサーネットワークのデータ処理機能にオフロードし，センサーネットワーク内（網内）でそれらを効率よく行うための自律的な知能センサーネットワークの新しい分散実行アーキテクチャを提案し，畳込みニューラルネットワーク（CNN）を網内実行するためのプロトコル設計を行ってきている．本論文ではその分散実行プロトコルを Intel Edison からなるセンサーネットワーク上に実装し，実行負荷ならびに通信容量に関する評価を行ったため，それについて報告する．歩行者の通常行動ならびに異常行動（転倒）に対しアナログ人感センサーアレイにより生成されるデータを用いて網内学習を実施した結果，Edison は最大でもアイドル時（0.046Ah）の 2 倍未満の電流量（0.080Ah）で動作した．また，学習時の通信による消費電力への影響は微量であることを確認した．

1. はじめに

分散コンピューティングのフレームワークは，サーバーやクライアントの通信性能や可用性，性能差などに応じ，クライアント指向やサーバー指向へと常に変遷しているが，近年では，クラウドサービスの信頼性と可用性向上により，データ処理をクラウドで集約して行う傾向がある．IoT に関するサービスでは，このクラウドヘビーコンピューティングに基づいており，それによって多くのツールが利用可能となっている．一方，マイクロソフトや Google TensorFlow [1,2] などによるいくつかの IoT ツールは，機械学習等により訓練された判定関数などを IoT デバイスに導入できるエッジヘビーコンピューティング機能をサポートしつつある．これにより，全センシングデータをクラウドサーバーに送信するための通信リソースを確保する必要がなくなり，プライバシーデータをローカルで処理することも可能となる．

そういった既存ツールおよび既存アプローチのほとんどは，学習済みの判定機能の一部または全部をクラウドからエッジデバイスに移行し，それ以後の検知や判定をエッジサイドで行うことで，クラウドへのデータ量を削減することが目的である．したがって，学習段階では依然として，学習機能を有するクラウドサーバーあるいはホームゲートウェイなどにすべてのデータを集約する必要がある．一方，現在 IoT デバイスと呼ばれるセンサーノードは，メモリ量や処理能力という点での性能向上が著しく，センシングやそのデータ送信のみならず，機械学習においても一部のタスクを実行することが可能であると考えられる．現状の無線センサーネットワーク（WSN）は，経路制御を含むデータ集約プラットフォームとみなされることが多いが，これらのノードをシームレスに連携させることで，より多くのメモリと処理能力が利用可能となり，センシング・学習・フィードバックのプロセスのほとんどがセンサーネットワーク内で完結するような，知的な局所型データ処理プラットフォームを形成できる．

これに対し，我々の研究グループでは，WSN において深層学習およびそれを用いたデータ処理を行うための新しい手法を提案している [3–5]．同手法では，WSN の各センサーノードから連続的に生成されるデータを 2 次元や 3 次元の地理的データ (例えば，温度データ等) として扱い，畳み込みニューラルネットワーク (CNN) の各順伝播ユニット（畳み込み処理（フィルタ）やプーリング，全結合など）

[1] 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka, Japan
[a)] y-fukushima@ist.osaka-u.ac.jp
[b)] h-yamagu@ist.osaka-u.ac.jp
[c)] higashino@ist.osaka-u.ac.jp

を，WSN 内のノード（センサーノード）のいずれかに割り当てる．また，逆伝播時のパラメータ更新処理を分散型で行う仕組みを開発することで，ユニットのパラメータ更新を分散環境で実現する．

同手法においては，通信量やノード処理負荷の偏りをなるべく少なくするユニット割り当て方法ならびにプロトコルの実現指針を示しており，シミュレーションにおいて，各ノードの最大処理負荷や最大通信量を抑制しながら分散環境で CNN を実現できることがわかっている [5]．しかし，現実のセンサーノードに対するコード実装はまだなされておらず，CNN の分散実行ならびに近隣ノードとの通信によるノードの処理負荷や通信量の実ノードでの評価は行われていなかった．我々の目標は，それらのセンサーノードを将来的に省電力化し，エナジーハーベストなどで動作させることで，ニアゼロエネルギーでの知的センサーネットワークを実現することである．そのためには，実ノードにおける分散学習やデータ交換がどの程度のオーバヘッドで実施され，負荷分散が実ノードからなるセンサーネットワーク上でも実現されていることを検証する必要がある．

本論文では，文献 [5] の分散 CNN を Intel Edison に実装し，その処理負荷ならびに通信量などを評価したため，それについて報告する．実装した分散 CNN の性能検証のため，実環境で取得したセンシングデータを用いた実験を行った．具体的には，6×6 の薄型でエネルギー効率の高いモーションセンサーアレイを構築し，その前を通過する通常歩行（正常状態）ならびにその前での転倒動作（異常状態）を識別する分類器を CNN で構築するシナリオを対象とし，36 台の Edison を用いて分散学習を行う．1 台の Edison がモーションセンサー 1 つのデータを保持し，各 Edison が CNN におけるユニットの役割を果たすことで，CNN の処理を 36 台の Edison に分散させる．このもとで，この分散型による学習精度への影響の評価を行った．まず，学習時間短縮のため，分散型学習プロセスを 1 台に導入し，分散学習の精度評価を行った．その結果，提案手法（分散型 CNN）の学習精度は 89.7275%，データを WSN で集約して学習を行う方式（通常型 CNN）では 91.875%となり，通常の CNN と遜色ない学習精度を達成できることが確認された．また，分散型学習の際の処理負荷ならびに通信量評価のため，36 台の Edison を用いて分散学習を行い，そのときの各 Edison の処理負荷および通信量の評価を行った．具体的には，USB 電流チェッカーを用いて積算電流量を測定し，同時に送受信データ量を測定した．その結果，各 Edison の学習により増加した積算電流量は最大でも 0.034Ah であり，学習を行っていない場合の積算電流量（0.046Ah）よりも小さく，通常の 2 倍未満の消費電力で分散学習を行うことができることが確認された．また，通信量については，割り当てられているユニットによって異なるものの，いずれの場合も通信量増加に対する積算電流量はほとんど変わらないことがわかった．

2. 関連研究と位置づけ

近年，深層学習は多層ニューラルネットワークを学習する方法として幅広く研究されており [6]．音声認識 [7,8]，物体認識 [9,10]，画像検索 [11,12]，および強化学習 [13,14] など様々なデータ解析において大きな成果が得られている．一般的に，深層学習はデータ量が増加するほど高い精度を得ることができるが，画像などの高精細データにおける訓練では数千万のパラメータと数十億の訓練データが必要となる．したがって，それだけの膨大なデータ量により訓練する場合にはデータ量に応じた処理コストが要求される．そこで，訓練にかかる処理負荷の軽減を目指した分散実行手法が提案され，また，それらの手法を容易に扱うことを可能にする Theano [15]，Torch [16]，cuda-convnet and cuda-convnet2 [17]，Decaf [18]，Overfeat [19]，Caffe [20] など多くのフレームワークが開発されている．

文献 [21] では，大規模な深層学習のモデル学習のために，数千のコアおよび数千の計算スレッドを有する GPU と分散システムを用いる手法を提案している．GPU は数千の ALU コアを搭載しているため数値演算能力が優れる一方，メモリ制限が課題となる．この問題を解決するため，マルチ GPU を用いたデータ並列化，モデル並列化，およびデータモデル並列化からなる分散システムを導入している．データ並列化は，各 GPU が同じモデルと異なるデータセットで学習を実行し，その後異なる GPU から学習したパラメータ勾配を同期する手法である．モデル並列化は，大規模なモデルを分割し，分割したモデルを各 GPU で担当し，各 GPU が同じデータセットを異なるモデルで学習する手法である．しかしデータ並列化では，計算ノードが多数ある場合にはスムーズに学習を行うため学習率を下げる必要があり，モデル並列化では，モデル間の通信コストが問題となる．これに対し，データモデル並列化は，全結合層と畳み込み層の特性から，畳み込み層をデータ並列化し，全結合層をモデル並列化することで，大規模な深層学習のモデルより高速な学習を実現している．他に深層学習の並列化として，文献 [22] では 16,000 個の CPU を用いることで数十億のパラメータを持つ大規模な深層学習モデルでの学習精度を向上させる手法が提案されている．また文献 [23] では，6,000 万のパラメータと 65 万のニューロンを持つ大規模な畳み込みニューラルネットワークにおいて，GPU を用いることによる学習高速化が報告されている．

文献 [24] では，大規模な深層学習のモデルを学習するための手法として，SINGA という一般的な分散型の深層学習のプラットフォームを提案している．SINGA は畳み込みニューラルネットワーク（CNN），ボルツマンマシン（RBM），および再帰ニューラルネットワーク（RNN）の一般的な深層学習モデルをサポートする．SINGA は，パ

ラメータ勾配を計算する TrainOneBatch と順伝播を行う NeuralNet からなる Worker ユニットと，結果を集約する Stub ユニット，パラメータを更新する Server ユニットの 3 つのコンポーネントを提供し，これらのユニットをユーザーが設定することで分散深層学習を行える．Worker の NeuralNet では CNN や RBN，RNN のニューラルネットワークを設定でき，大規模なモデルでは，次の 4 つの並列化手法（(1) 全てのレイヤーを異なるサブセットに分割，(2) 1 つのレイヤーをバッチ次元によってサブレイヤーに分割，(3) 1 つのレイヤーを特徴次元によってサブレイヤーに分割，(4) (1)〜(3) の手法の組み合わせ）が提供される [25]．また，TrainOneBatch では，パラメータ勾配を計算するために順伝播型ニューラルネットワークや RNN などで用いられるバックプロパゲーションとエネルギーモデルで用いられる対比分散アルゴリズムが提供されている．文献 [26] では，並列確率勾配降下アルゴリズムが提案されている．

SINGA では Worker と Server を用いた多様な同期および非同期のフレームワークを提供でき，例えば Sandblaster [22] や AllReduce [27] といった同期フレームワークや Downpour [22,28] や Distributed Hogwild [29] といった非同期フレームワークが知られている．同期フレームワークは，分散により 1 回の学習速度を早くすることができ，非同期フレームワークは収束率を高めることができる．学習速度と収束率はトレードオフの関係であり，異なるフレームワークを組み合わせて収束率と学習速度における最適性を得ることができる．

様々な分散学習のオープンソースパッケージも開発されている．All of Distributed Machine Learning Toolkit (DMTK) [30] や Distributed TensorFlow [1]，ChainerMN [31] はデータ並列化によって分散学習を実行する．データの並列化は，分割されたデータセットを各 GPU 上でミニバッチとして処理することで学習速度を向上させるが，全てのパラメータ勾配を集約してパラメータを更新する必要がある．すなわち，データの並列化には，低遅延で高品質なネットワークが必要となる．

近年では，モバイル端末による深層学習に関する研究もなされている．MoDNN [32] は，事前に学習した CNN モデルをモバイル端末上に分割することによって，CNN の高速化を行う手法を述べている．DeepX [33] や DeepMon [34] では，複数のコプロセッサで計算処理を分割することによって，モバイル端末上での CNN の実行が可能であることを示している．これらのアプローチでは，事前に学習した CNN をモバイル端末上で実行することが可能であるが，モバイル端末上で学習を実行する手法については触れられていない．

3. 提案手法の概要

以下では，我々が提唱している無線センサーネットワークにおける網内学習機構 MicroDeep についてその概要を述べる．

3.1 想定環境

MicroDeep では，ある程度のプロセシング能力と通信機能を搭載したセンサーノードが無線通信によって WSN を構成している状況を想定する．WSN 内のセンサーノード同士は OLSR などの適切な経路制御プロトコルにより互いに通信可能な状態であるとする．

MicroDeep では，入力層，T 層の隠れ層 $(1 \leq T)$，全結合層 f，出力層 o からなる一般的な CNN を対象とする．t 層目の隠れ層 $(1 \leq t \leq T)$ は畳み込み層 $c^{(t)}$ およびプーリング層 $p^{(t)}$ からなり，プーリング層は任意であるものとする．各層のユニットは XY 座標を用いて，ユニットの座標を非負の整数値で (x, y) のように表し，左下のユニットを原点 $(0, 0)$ とする．本稿で用いる CNN パラメータ等の表記を表 1 に示す．

MicroDeep では多地点センシングデータを画像のような 2 次元データとみなし，各データを有するセンサーノードが周辺のセンサーノードから畳み込みなどに必要なデータを受け取り CNN の順伝播を実現する．

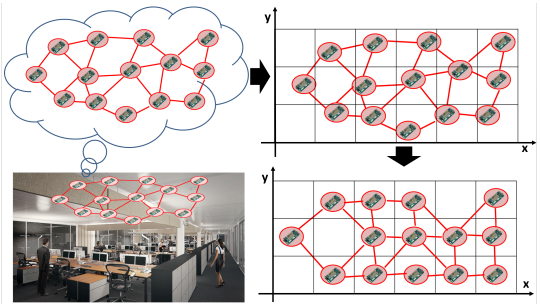
センサーノードのユニットの割り当てについて，本稿では，文献 [5] の手法を用いて，2 つのセンサーノード間の物理的距離を WSN におけるそれらの間の無線リンクに対応させることによって，センサーノードの座標を CNN の XY 座標に適合させる．具体的には，各センサーノードの位置情報に基づいて，CNN の XY 座標平面に割り当てる簡単な手法を用いて，(x, y) の CNN の各ユニットを (x, y) のセンサーノードに割り当てる．理想的には，$N \times M$ のアドレス空間に対し，$N \times M$ 個のセンサーノードがグリッド状に配置され，センサーノードが XY 座標平面に 1 対 1 のマッピングが可能であり、CNN のすべてのユニットをそれらのセンサーノードに割り当てることができる状態である．しかし，実環境においては，配置の制約などからグリッド状にセンサーノードを配置できない事があり，図 1(a) に示すように全ての座標平面にセンサーノードが割り当てられない場合が考えられる．このような場合，センサーノードが割り当てられなかった座標はセンサーノードの欠損として，データの欠けた面的なデータとして CNN のユニット処理を行う．各センサーノードが XY 座標平面に割り当てられると，図 1(b) に示すように，座標 (x, y) のセンサーノードは，CNN においてその座標系に割り当てられた各層のユニットの処理を行う．

3.2 分散実行プロトコル

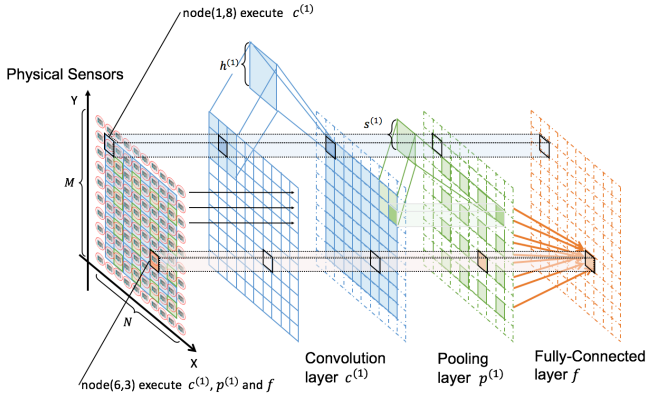
MicroDeep では，CNN の各ユニットをセンサーノードに対応づける．順伝播では．各センサーノードは WSN を介して，(入力層を含む) ユニットの出力データを交換し，

表 1　CNN のパラメータ表記

パラメータ	詳細
(N, M)	入力データサイズ
T	隠れ層の数
$c^{(t)}$	t 番目の畳み込み層
$p^{(t)}$	$c^{(t)}$ の後のプーリング層
$c_k^{(t)}(x, y)$	(x, y) 座標の k 番目のフィルタの $c^{(t)}$ のユニット
$p^{(t)}(x, y)$	(x, y) 座標の $p^{(t)}$ のユニット
$f(x, y)$	(x, y) 座標の全結合層のユニット
$o(x, y)$	(x, y) 座標の出力層のユニット
$K^{(t)}$	$c^{(t)}$ のフィルタ数
$h^{(t)}$	$c^{(t)}$ のフィルタサイズ
$s^{(t)}$	$p^{(t)}$ のプーリングサイズ



(a) WSN の座標系への割当

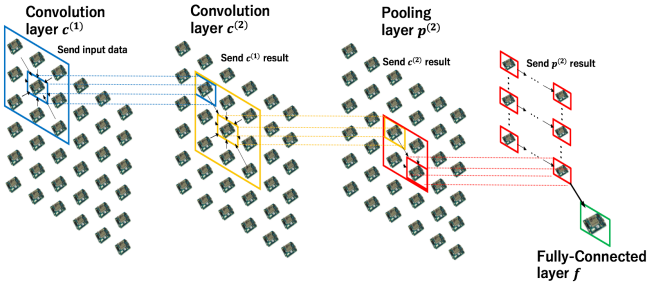


(b) CNN の座標系への割当

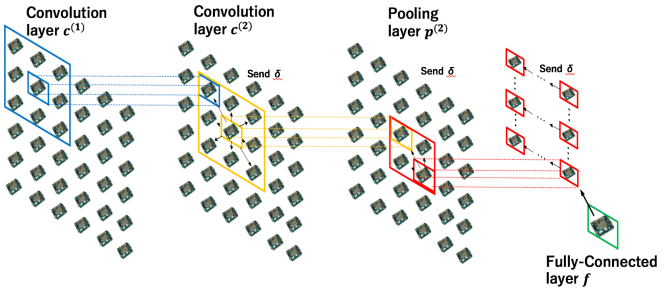
図 1　センサーノードのユニット割当

ユニットの処理を行う．出力層のユニットを割り当てられているセンサーノードは，出力データと正解データから逆伝播処理を行う．逆伝播では，各ユニットが前のユニットの伝播のみに基づいて分散実行し，重みを更新する．したがってユニット間での重み共有は行わず，その更新はユニット毎に独立して行われる．

図 2 に，畳み込み 2 層，プーリング 1 層の場合における CNN の順伝播と逆伝播の処理を示す，WSN 上でこの処理を行うための，(1) 順伝播におけるユニットの実行，(2) 逆伝播におけるユニット毎の重み更新，の各処理およびセンサーノードが割り当てられない座標が存在する場合の処理



(a) 順伝播



(b) 逆伝播

図 2　分散型 CNN の処理例

について次節以降で述べる．

3.2.1　順伝播処理

説明の簡単のため，本章では WSN 内の $N \times M$ 個のセンサーノードが $N \times M$ のアドレス空間に割り当てられたとする．センサーノードが欠損している場合については後述する．

順伝播処理では，センサーノード間の通信によって，通常の CNN と同様に行われる．具体的には，センサーノード (x, y) は，割り当てられた各層のユニット $c_k^{(t)}(x, y)$，$p^{(t)}(x, y)$，$f(x, y)$，$o(x, y)$ を実行する．つまり CNN において (x, y) に対応する全てのユニットは，センサーノード (x, y) で実行される．

畳み込み層のユニット $c_k^{(t)}(x, y)$ を実行するために，周囲のノードからデータを取得する際に，そのノードまでの距離を表したものであるオフセット O_x, O_y を導入する．オフセット O_x, O_y はプーリング処理を行う度に，以下の式によって加算される．

$$(O_x^{(t)}, O_y^{(t)}) = (O_x^{(t-1)} + s^{(t-1)}, O_y^{(t-1)} + s^{(t-1)})$$

ただし，前の層にプーリング層がなければ，$s^{(t-1)} = 0$ とする．畳み込み層のユニット $c_k^{(t)}(x, y)$ は，前の層のユニット $p^{(t-1)}(x + i \cdot O_x, y + j \cdot O_y)$ に対応するセンサーノードからデータを受信し，畳み込み処理を行う．　ただし，i, j は $\underline{\rho}_k^{(t)} \leq i, j \leq \overline{\rho}_k^{(t)}$ とし，$\underline{\rho}_k^{(t)}$，$\overline{\rho}_k^{(t)}$ は

$$\underline{\rho}_k^{(t)} = -\lfloor \frac{h^{(t)}}{2} \rfloor, \quad \overline{\rho}_k^{(t)} = \lfloor \frac{h^{(t)}}{2} \rfloor - d.$$

とする．このとき，フィルタサイズ $h^{(t)}$ が奇数のとき

$d=-1$，偶数の時 $d=0$ である．入力データとして，前の層にプーリング層が存在しない場合は，入力層および畳み込み層のデータを用いて計算される，

プーリング層のユニット $p^{(t)}(x,y)$ は，前の層のユニット $c_k^{(t)}(x+i\cdot O_x^{(t)}, y+j\cdot O_y^{(t)})$ に対応するセンサーノードからデータを受信し．プーリング処理を行う．ただし，$\underline{\rho}_k^{(t)}$，$\overline{\rho}_k^{(t)}$ の導出の際には，フィルタサイズ $h^{(t)}$ の代わりに，プーリングサイズ $s^{(t)}$ を用いる．

3.2.2 逆伝播処理

通常の逆伝播アルゴリズムでは，逆伝播によって求められた各層の全てのユニットの微分結果を用いて，その層におけるユニットの重みを更新する集中型の処理である．これを分散環境で模擬するため，ユニットが後の層のユニットから微分結果を取得し，その結果のみを用いて重みを更新する分散アルゴリズムを設計する．したがって MicroDeep では，各ユニットは他のユニットと重みを共有しない．

出力層のユニットが実行され，正解データを取得すると，微分 δ を計算し，その結果を前の層のユニットへ送信する．微分 δ を受け取ったユニットは，以下の式によって，前の層へ送信するための δ を求める．このとき．$w_i^{(t)}$ および $\delta_i^{(t)}$ は，$T-1$ 層での i 番目のユニットから T 層での j 番目のユニットへの重みおよび微分を表す．

$$\delta_i^{(t)} = \sum_j \delta_j^{(t+1)}(w_j^{(t+1)} f'(u_j^{(t)}))$$

そして各ユニットは，以下の式によって，他のユニットの微分結果を用いずパラメータ更新を行う．そのため，各ユニットで別々にパラメータが更新される．このとき，ϵ は学習率で $z^{(t)}$ は前の層の出力結果を表す．

$$w_i^{(t+1)} = w_i^{(t)} - \epsilon\delta_i^{(t)} z_i^{(t)}$$

3.2.3 ノード欠損時の処理

実環境では必ずしも全ての座標にセンサーノードが存在するとは限らない．例えば，センサーノードが 11 個しか配置されていない場合，4×3 の空間を網羅することはできない．そういったセンサーノードの欠損に対しても学習処理を可能にするための方法を述べる．

センサーノードがセンサーデータを生成するため，センサーノードの欠損は，CNN の入力データを失うことを意味する．そこで，欠損部分の入力データを必要とするユニットは，その部分を 0 で補間する．この補間による入力データは，畳み込み処理の線形結合性より無視される．また，センサーノードの欠損は入力データの欠損だけでなく，畳み込み層の出力結果を失うことでもある．そのため，畳み込み層の出力を必要とするユニットは，同様に 0 として補間する．この補間データは，次の層の畳み込み層およびプーリング層の処理によって無視される．次の層が畳み込み層の場合は，上述と同様に線形結合性により無視され，

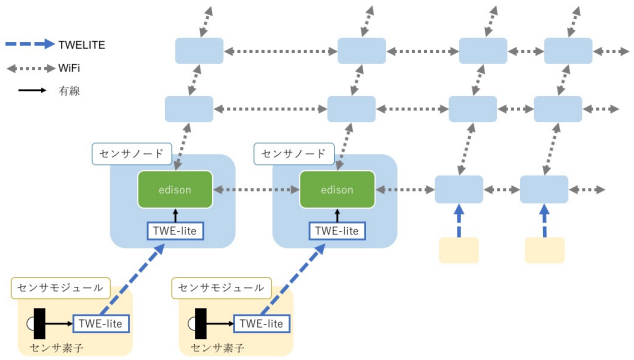


図 3　システムの概要

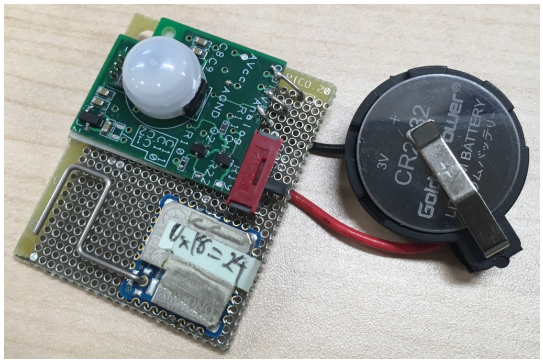


図 4　実装した焦電型赤外線センサーモジュール

プーリング層の場合は，活性化関数 ReLU により入力データが非負の値であり，それに MAX プーリング処理を行うため，0 は無視される．プーリング層におけるユニットの欠損も同様である．

全結合層および出力層の場合は，処理を 1 つのノードに集約して行うため，このセンサーノードの欠損は，順伝播および逆伝播処理を行うことができないということである．したがって，この場合には，代替ノードを設け，そのノードが処理を行うものとする．

4. MicroDeep の実装

4.1 システムの概要

MicroDeep を実現するシステムの概要を図 3 に示す．本システムはセンサー部の自由な空間配置を実現するため，センサー部をセンサーノードとは独立にセンサーモジュールとして実現し，1 対 1 対応するセンサーノードとは IEEE802.15.4 準拠の通信で接続する．各センサーノードはセンサーモジュールで収集されたデータを用いて，他のノードとの通信により分散学習を実行する．CNN の分散実行処理を行うセンサーノードとして Intel Edison を使用し，グリッド状に配置する．

4.2 センサーモジュールの実装

センサーモジュールとして，検知範囲内の熱源の変化をアナログ電圧として出力する焦電型の赤外線センサー素子

図 5　センサーモジュールアレイ

図 6　実装したセンサノード

を準備した．この焦電型赤外線センサー素子は，検知範囲の熱源の増減に反応する特性を有する．センサーの検知範囲内の熱源配置に変化がない場合，センサー素子の出力電圧は一定であるが，検知範囲内を人が通過すると出力電圧は大きく変化し，時間経過とともに定常状態の出力電圧へと振動しながら収束する．使用した赤外線センサ素子をセンサーモジュール化したものを図 4 に示す．この焦電型赤外線センサーを，2.4GHz 帯無線（IEEE802.15.4）を使用した無線マイコン TWELITE に接続し，赤外線センサー素子がアナログ出力する 0-2.4V の範囲の電圧を．同様に TWILITE を接続したセンサーノード（Intel Edison）に向けて毎秒約 5 回送信する．なお，センサーモジュールは無線送信部を含めて 3.3V のボタン電池 1 個で駆動し，大きさは縦 4.2cm，横 3.5cm である．

本研究では，センサーノードと同数である 36 個の赤外線センサーモジュールを作成し，図 5 に示すように横 30cm，縦 20cm の間隔で縦横 6 個ずつグリッド状に配置しセンサーアレイを構築した．なお，後述する実験では，センサーモジュールによって収集したデータを予めセンサーノードに与えたうえで実行処理を行っている．

4.3　センサーノードの実装

センサーノードには Intel Edison を使用し，以下で述べる機能を実装している．Intel Edison の CPU は 500MHz デュアルコア，MCU はインテル Quark プロセッサー・マイクロコントローラー 100MHz，RAM は 1GBytes，内部ストレージは 4GBytes であり，デュアルバンド Wi-Fi と Bluetooth4.0 を搭載する．Breakout Board を介した Edison の GPIO と TWELITE のチップを接続したセンサーノードを図 6 に示す．

4.3.1　センサー間ネットワークの構築

センサーノード間の通信については，実装の簡便を考慮し，Wi-Fi AP を複数用意し，AP を介したインフラストラクチャーモードで実現している．なお，アドホックモードでのメッシュネットワークの構築も試行したが，安定性に課題を残したため，そのような方針としている．BATMAN プロトコル等の利用も可能であるが，ルーティングオーバヘッドが大きいこと，ならびに研究室レベルの大きさでは全ノードが 1 ホップ内に収まってしまい，AP を利用する場合と差異がないため，AP による接続としている．

Edison は格子状に配置し，位置に応じて IP アドレスを付与する．すなわち，今回は既にグリッド状になっている WSN を想定し，その位置情報に基づいて CNN のユニットが割り当てられていると仮定する．位置と IP アドレスが相互に変換可能な割り当て規則を採用することで，MicroDeep で必要な位置情報通信を疑似的に実現する．

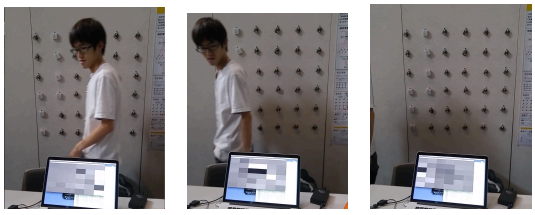
4.3.2　CNN ユニットの実装

各センサーノードは，順伝播における畳み込み層，プーリング層，全結合層のユニットを有し，データの送信ノードおよび受信ノードの位置情報から IP アドレスを求め，TCP ソケット通信でデータ交換を行う．逆伝播では，順伝播と逆のデータの流れで処理が行われるため，順伝播で作成したソケットを利用してデータの送受信を行う．逆伝播のユニットの処理が完了した後，畳み込み層ユニットが割り当てられているノードでは，受信した前の層のデータを用いてノード毎にパラメータの更新を行う．この際ノード間の通信は行われない．

4.3.3　欠損プログラム

ノード間通信が不安定な場合やノードが動作停止した場合などはノード欠損として扱う．そのようなノード欠損は，ノード間データ送受信ソケットの監視および一定時間のタイムアウト処理により判断する．具体的には，ノードへの接続要求，データ送受信待ちにおいてタイムアウト時間を設定し，タイムアウトとみなした場合は以降その通信先のノードは欠損として扱う．なお，タイムアウト時間は各層の実行時間を考慮し，層毎に決定する．

今，入力層にセンサーデータ入力があった時刻を便宜上，時刻 0 とする．第 n 層の順伝播におけるタイムアウトの時刻を F_n，逆伝播におけるタイムアウト時刻を B_n とする．また，初期接続要求のタイムアウト時間を t_c，各層における順伝播処理時間を t_f，逆伝播処理時間を t_b とし，最後の層を L とする．このとき，以下のようにタイムアウト時刻を定義する．



(a) (b) (c)

図 7 センサーからの入力データの視覚化

$$
\begin{aligned}
F_1 &= t_c + t_f \\
F_n &= \sum_{i=0}^{n-1} F_i + t_f \\
B_L &= \sum_{i=0}^{L} F_i + t_f + t_b \\
B_n &= \sum_{i=n+1}^{L} B_i + t_b
\end{aligned}
$$

ソケットについては読み込みおよび書き込みができなくなった場合を切断とみなし，ノード欠損として扱う．これにより欠損とみなされたノードにはデータを送信せず，そのノードから受信するはずのデータは 0 として扱う．

5. MicroDeep を用いた異常歩行検知と評価

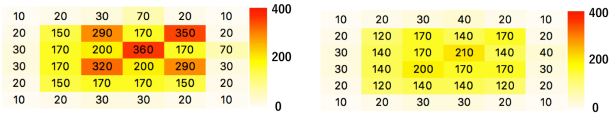
実装した MicroDeep を用いて以下の性能評価実験を行った．

5.1 データセット

図 5 のセンサーアレイを用いて壁に設置し，5 人の被験者にその前を歩行または転倒してもらい，計 55 回の歩行サンプルを取得した．センサーアレイの前を人が通過した際の出力電圧の変化を PC に表示させた様子を図 7 に示す．55 回中 23 回は転倒動作であり，それらには「異常」のラベルを，通常の通過動作には「正常」のラベルを付与した．取得したデータは毎秒約 5 フレームの 6×6 ピクセルの動画とみなすことができる．今回は，2 秒程度で人の通過を検知できることから 10 フレームを 1 データとしている．また，スライディングウィンドウを用い，1 フレームごとにスライドさせたデータを構成し，$6 \times 6 \times 10$ のデータを計 1600 収集し，CNN への入力データとした，うち 1070 個は「正常」ラベル，530 個は「異常」ラベルであった．

5.2 学習精度とデータ通信量の評価

まず，評価の効率化のため，実装した MicroDeep を 1 ノードに集約し，仮想的な分散環境において学習精度を評価した．本評価では，畳み込み層 1 層，プーリング層 1 層，全結合層 1 層，出力層 1 層の CNN を用いた．CNN のハイパーパラメータはそれぞれ以下のように設定した．MicroDeep による分散学習の場合は，フィルタ数：$K^{(1)} = 5$，フィルタサイズ：$h^{(1)} = 3$，プーリングサイズ：$s^{(1)} = 2$，パディングサイズ：0，全結合のユニット数：70，バッチサイズ：4，学習回数：10 とした，10 回ずつ学習と評価を行った結果を平均化したものを学習精度とした．なお，学習には時間を要するため，学習精度の評価時にはデータ送受信を省略するため，前述のように 1 台に分散プロセスを導入し，仮想的に分散学習を行っている．また，データ通信量は 1 回の送受信データを 1 として評価を行った．



(a) 最適なパラメータ選択の場合 (b) 制約の中でのパラメータ選択の場合

図 8 各センサーノードのデータ通信量

その結果，最適なハイパーパラメータを設定した場合において，学習精度は 91.875%，各センサーノードの通信量は図 8(a) で示され，最大通信量は 360 となった．一方，提案手法において，学習精度は 89.7275%，各センサーノードの通信量は図 8(b) で示され，最大通信量は 210 となった．したがって，提案手法の性能としては，通常の最適化された CNN と比較して，およそ 2%の学習精度の低下がみられたが，データ通信量はおよそ 40%削減された．

また，通常の CNN において提案手法と同じハイパーパラメータを設定した場合，学習精度は 90.4775%となり，提案手法における重み共有を行わないパラメータ更新でも，精度の差は 1%未満であり，通常の CNN と同程度の学習精度を達成可能であることが確認できた．

提案手法の学習精度は，依然として高い精度を達成しており，分散学習を実行するにあたって，十分な精度だといえる．

5.3 センサー NW のノードの欠損による影響

一部のセンサーが欠損している状態での評価についても行った．6×6 のセンサーノードのうちランダムに 1 割 (3 個) および 2 割 (7 個) のセンサーノードが欠損している場合について評価を行った．その結果，1 割のノード欠損では 89.7075%，2 割のノード欠損では 89.03%となった．この結果から，センサノードの 2 割が欠損したとしても，センサーネットワークの性能を維持できることが確認された．

また，プーリング層の処理を行っている 4 つのノードのうち 3 つが欠損した場合には 81.125%となった．これは重要な特徴が欠損しすぎることによって，大幅な学習精度の低下となった．しかし，このような欠損の状況は非常に稀であり，また，このような最悪な状況でも学習精度はある

表 2　各ノードの積算電流量

各ノード	学習時	通常時	増加量
入力層のノード	0.063Ah	0.046Ah	0.017Ah
畳み込みノード	0.078Ah	0.046Ah	0.032Ah
プーリングノード	0.079Ah	0.046Ah	0.033Ah
全結合ノード	0.080Ah	0.046Ah	0.034Ah

程度維持できているといえる．

5.4　Edison による消費電力と実行時間の評価

図 9 のように実際に，36 台の Edison に分散型深層学習を実装し，学習を行った際の消費電力と実行時間についての評価を行った．消費電力については，36 台のうち同層のユニットが割り当てられているノードの消費電力の差は微量であると考え，割り当てられているユニットごとに，入力層のノード，畳み込みノード，プーリングノード，全結合ノードの 4 種類に分割して評価を行った．消費電力の評価方法については，USB 電流チェッカーを用いて，学習を初めてから終わるまでの積算電流量を測定する．その学習にかかった積算電流量と，学習にかかった実行時間と同じ時間の何も実行しない場合の Edison の積算電流量を比較することで，分散学習による 1 つのノードにおける消費電力を求める．また，その時の送信データ量と受信データ量についても ifconfig コマンドを用いて同様の方法で評価を行った．5.2 節と同様のパラメータにおいて，分散学習を実行した．この時の様子を図 10 に示す．その結果，実行時間は 30:33 となり，積算電流量，送信データ量および受信データ量は表 2，3，4 のようになった．送信データおよび受信データ量の結果から，仮想的にデータ通信量の評価を行った図 8 と比較しても，同様の比率であり，5.2 節の仮想的な評価によって，データ通信量を評価することができるといえ，提案手法によってデータ通信量を削減することができることが分かる．

入力層のノードは，データの送信のみを行い，計算処理を行わないため，他の Edison と同期すること無く送信データを送るとプログラムが終了する．一方，他のノードは，データの送受信の後，計算処理を行うため，他のノードの処理を待つ必要がある．そのため，学習が終わるまでプログラムを動かし続ける必要がある．そのため，入力層のノードのみ積算電力量が低い結果となったと考えられる．また，各ノードのデータの送受信量については，層が深いノードほど処理が多いため，送受信データ量が大きい結果となった．これら 2 つの結果から，送受信データ量の増加による積算電力量への影響は軽微であることが分かる．したがって，分散学習において，センサーノード間の通信量を許容することができると考えられる．

次に学習において実行時間の要因と考えられるパラメータのうちフィルタ数，学習回数，ユニット数およびバッチサイズを変更した時の実行時間について評価を行った．な

図 9　Edison による評価実験の様子

(a) 実行初期　(b) 実行途中　(c) 実行終了時

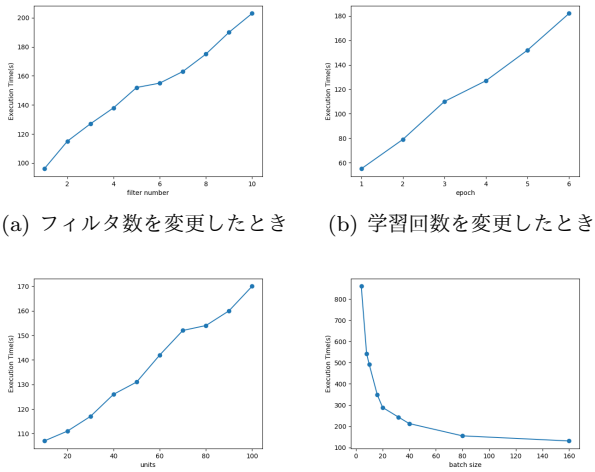
図 10　分散学習時の Tester

表 3　各ノードの送信バイト数

各ノード	学習時	通常時	学習時-通常時
入力層のノード	1.2MiB	249.8KiB	950.2KiB
畳み込みノード	8.3MiB	250.0KiB	8.1MiB
プーリングノード	15.7MiB	246KiB	15.5MiB
全結合ノード	20.2MiB	236.3KiB	20.0MiB

表 4　各ノードの受信バイト数

各ノード	学習時	通常時	学習時-通常時
入力層のノード	569.6KiB	423.4KiB	146.2KiB
畳み込みノード	7.5MiB	408.7KiB	7.1MiB
プーリングノード	14.6MiB	420.8KiB	14.2MiB
全結合ノード	16.6MiB	393.5KiB	16.2MiB



(a) フィルタ数を変更したとき　(b) 学習回数を変更したとき

(c) ユニット数を変更したとき (d) バッチサイズを変更したとき (仮)

図 11　パラメータを変更したときの実行時間の変化

お，ノード数の関係上，層数とフィルタサイズの変更はできなかったため，それらは同じとしている．フィルタ数 5，学習回数 5，ユニット数 50，バッチサイズ 160 に固定したとき，それぞれのパラメータを変更した時の実行時間を図 11 に示す．図 11 の結果から，実行時間の要因となるパラメータの中で．フィルタ数，学習回数，ユニット数に関しては，実行時間とほぼ比例関係にあり，バッチサイズに関しては，バッチサイズを増やすごとに徐々に実行時間の短縮率が低下することが確認された．

6. おわりに

本研究では，無線センサーネットワーク内で深層学習を実行する MicroDeep の実装と評価について述べた．MicroDeep では，CNN のユニットを無線センサーノードに割り当て，センサーノード間の通信によって CNN の学習を分散実行する．モーションセンサーアレイを用いて，その前を通過する際の通常歩行および転倒動作を検知するシナリオを対象とし，この分散実行プロトコルによる学習精度への影響を，通常の学習プロトコルと比較することで評価を行った．また，実際に分散学習プロトコルを Intel Edison からなるセンサーネットワーク上に実装し，Edison の処理負荷および通信量に関する評価を行った．その結果，分散型による学習精度への影響およびデータ通信による消費電力の増加は軽微であり，分散学習時の消費電力はアイドル時の 2 倍未満であることが確認された．

なお，今回は，センサーモジュールによって予め収集したデータを用いた分散学習を行ったが，今後はセンサーモジュールで収集したデータをセンサーノードで逐次学習するシステムの実現を目指していく．また，欠損ノードの早期発見機構なども実装し，実環境での実行に耐えうるシステム実装を行っていく予定である．最終的には，この分散学習システムをエナジーハーベスティングなセンサーに導入し，WSN 内でエネルギー効率の高い深層学習を実現していきたいと考えている．

謝辞

本研究成果の一部は，国立研究開発法人情報通信研究機構（NICT）の委託研究「未来を創る新たなネットワーク基盤技術に関する研究開発」ならびに JSPS 科研費 15K12019 の助成を受けたものです．

参考文献

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M. et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems, *arXiv preprint arXiv:1603.04467* (2016).

[2] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al.: TensorFlow: A system for large-scale machine learning, *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, pp. 265–283 (2016).

[3] 福島悠太，三浦太樹，濱谷尚志，山口弘純，東野輝夫：無線センサーネットワークにおける深層学習の分散実行に関する検討，情報処理学会第 170 回 DPS 研究会技術報告，pp. 1–8 (2017).

[4] 福島悠太，三浦太樹，濱谷尚志，山口弘純，東野輝夫：センサーネットワークにおける分散型深層学習の設計と実装，技術報告 (2017).

[5] Fukushima, Y., Miura, D., Hamatani, T., Yamaguchi, H. and Higashino, T.: MicroDeep: In-network Deep Learning by Micro-sensor Coordination for Pervasive Computing, *Proceedings of the 4th IEEE International Conference on Smart Computing (IEEE SMARTCOMP2018)*, pp. 1–8 (2018).

[6] LeCun, Y., Bengio, Y. and Hinton, G.: Deep learning, *Nature*, Vol. 521, No. 7553, pp. 436–444 (2015).

[7] Abdel-Hamid, O., r. Mohamed, A., Jiang, H. and Penn, G.: Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition, *Proceedings of 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 4277–4280 (online), DOI: 10.1109/ICASSP.2012.6288864 (2012).

[8] Sainath, T. N., Kingsbury, B., r. Mohamed, A., Dahl, G. E., Saon, G., Soltau, H., Beran, T., Aravkin, A. Y. and Ramabhadran, B.: Improvements to Deep Convolutional Neural Networks for LVCSR, *Proceedings of 2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, IEEE, pp. 315–320 (online), DOI: 10.1109/ASRU.2013.6707749 (2013).

[9] Scherer, D., Müller, A. and Behnke, S.: Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition, *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, ICANN'10, Berlin, Heidelberg, Springer-Verlag, pp. 92–101 (online), available from ⟨http://dl.acm.org/citation.cfm?id=1886436.1886447⟩ (2010).

[10] Huang, F. J. and LeCun, Y.: Large-scale Learning with SVM and Convolutional for Generic Object Categorization, *Proceedings of 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 1, pp. 284–291 (online), DOI: 10.1109/CVPR.2006.164 (2006).

[11] Babenko, A., Slesarev, A., Chigorin, A. and Lempitsky, V.: Neural codes for image retrieval, *Proceedings of European conference on computer vision*, Springer, pp. 584–599 (2014).

[12] Ge, T., Ke, Q. and Sun, J.: Sparse-Coded Features for Image Retrieval., *BMVC* (2013).

[13] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M.: Playing atari with deep reinforcement learning, *arXiv preprint arXiv:1312.5602* (2013).

[14] Wang, X.: Deep Reinforcement Learning (2016).

[15] : `http://deeplearning.stanford.edu/wiki/index.php/Neural_Networks`.

[16] Tu, J. V.: Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes, *Journal of clinical epidemiology*, Vol. 49, No. 11, pp. 1225–1231 (1996).

[17] : `https://code.google.com/p/cuda-convnet2/`.

[18] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. and Darrell, T.: DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition., *Icml*, Vol. 32, pp. 647–655 (2014).

[19] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. and LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks, *arXiv preprint arXiv:1312.6229* (2013).

[20] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T.: Caffe: Convolutional architecture for fast feature embedding, *Proceedings of the 22nd ACM international conference on Multimedia*, ACM, pp. 675–678 (2014).

[21] Buyya, R., Calheiros, R. N. and Dastjerdi, A. V.: *Big Data: Principles and Paradigms*, Morgan Kaufmann (2016).

[22] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V. et al.: Large scale distributed deep networks, *Advances in neural information processing systems*, pp. 1223–1231 (2012).

[23] Schmidhuber, J.: Deep learning in neural networks: An overview, *Neural networks*, Vol. 61, pp. 85–117 (2015).

[24] Ooi, B. C., Tan, K.-L., Wang, S., Wang, W., Cai, Q., Chen, G., Gao, J., Luo, Z., Tung, A. K., Wang, Y. et al.: SINGA: A distributed deep learning platform, *Proceedings of the 23rd ACM international conference on Multimedia*, ACM, pp. 685–688 (2015).

[25] Wang, W., Chen, G., Dinh, A. T. T., Gao, J., Ooi, B. C., Tan, K.-L. and Wang, S.: SINGA: Putting deep learning in the hands of multimedia users, *Proceedings of the 23rd ACM international conference on Multimedia*, ACM, pp. 25–34 (2015).

[26] Zinkevich, M., Weimer, M., Li, L. and Smola, A. J.: Parallelized stochastic gradient descent, *Advances in neural information processing systems*, pp. 2595–2603 (2010).

[27] Wu, R., Yan, S., Shan, Y., Dang, Q. and Sun, G.: Deep image: Scaling up image recognition, *arXiv preprint arXiv:1501.02876*, Vol. 7, No. 8 (2015).

[28] Agarwal, A. and Duchi, J. C.: Distributed delayed stochastic optimization, *Advances in Neural Information Processing Systems*, pp. 873–881 (2011).

[29] Recht, B., Re, C., Wright, S. and Niu, F.: Hogwild: A lock-free approach to parallelizing stochastic gradient descent, *Advances in Neural Information Processing Systems*, pp. 693–701 (2011).

[30] : `https://github.com/Microsoft/DMTK`.

[31] : `https://github.com/chainer/chainermn`.

[32] Mao, J., Chen, X., Nixon, K. W., Krieger, C. and Chen, Y.: MoDNN: Local distributed mobile computing system for Deep Neural Network, *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, pp. 1396–1401 (2017).

[33] Lane, N. D., Bhattacharya, S., Georgiev, P., Forlivesi, C., Jiao, L., Qendro, L. and Kawsar, F.: Deepx: A software accelerator for low-power deep learning inference on mobile devices, *Proceedings of the 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, IEEE, pp. 1–12 (2016).

[34] Huynh, L. N., Lee, Y. and Balan, R. K.: DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications, *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, ACM, pp. 82–95 (2017).