MCUへのAI実装に向けたAI開発環境とモデル圧縮技術

AI Development Environment and Compression Technology for Realization of AI on MCU

村瀬　輝高 MURASE Terutaka　　アジャイ カドルカル Ajay KADOLKAR　　久本　康司 HISAMOTO Koji

IoT（Internet of Things）デバイスの普及に伴い，エッジコンピューティングの重要性が高まる中で，低消費電力・低コスト・小型設計を特徴とするMCU（マイクロコントローラーユニット）は，IoTシステムの中核を担うプラットフォームとして注目されている。

東芝デバイス&ストレージ（株）は，MCUにAIを実装する有用性に着目し，OSS（オープンソースソフトウェア）を活用したMCU向けAI開発環境を構築している。AIのモデルは，重み係数の数が膨大になるため，学習可能なテンソル列（TT：Tensor Train）分解[1]ネットワークに基づく圧縮技術を開発・導入して，モデルの推論精度を維持しながら，モデルサイズと計算負荷の削減に取り組んでいる。これにより，エッジデバイス上でのリアルタイムAI推論の実用化を促進する有効なアプローチを実現する。

With edge computing becoming more important due to advancements in Internet of Things (IoT) devices, demand is increasing for compact, low power consumption, and low cost microcontroller units (MCUs), which are a key platform of IoT systems.

Toshiba Electronic Devices & Storage Corporation has built an artificial intelligence (AI) development environment focusing on the usability of AI on MCU, which is capable thanks to open-source software (OSS). We have also developed and introduced new compression technology based on the trainable tensor-train decomposition network (TTD-Net) to reduce both the size of AI models including a large number of weight coefficients and the computational load while maintaining inference accuracy. Our approach contributes to making practical use of real-time AI inference on edge devices.

1. まえがき

近年，IoTデバイスの普及に伴い，現場側のIoTデバイスで処理を行うエッジコンピューティングの重要性が急速に高まっている。IoTデバイスには，センサーやアクチュエーターを通じて物理環境と直接やり取りし，リアルタイムで意思決定を行う能力が求められる。このような背景の中，MCUは，低消費電力・低コスト・小型設計といった特性を備え，IoTシステムの中核を担うプラットフォームとして注目されている。

一方，AI技術の進展により，これまでクラウドサーバーで実行されていたAIアルゴリズムを，MCU上で実行することが可能になりつつある（ここでは，AIは，識別・検出・予測を目的とした深層学習を想定している）。このMCUの活用により，IoTデバイスは高度な機能を持つようになり，ユーザーインターフェースの向上や新たなアプリケーションの創出が期待される。

IoTデバイスの多くは，コストや消費電力の制約が厳しい環境で動作するため，MCUのようなプラットフォームが必要不可欠である。また，MCUは，汎用性があり，特定用途向けのカスタマイズが容易なため，様々なアプリケーションに適応可能である。更に，MCUにAIを実装することは，以下のような具体的な利点をもたらす。

⑴　レイテンシー（遅延）の低減　　クラウドサーバーとの間でデータを送受信する時間を削減し，リアルタイム性が求められるアプリケーションで確実な応答を実現する。

⑵　プライバシーとセキュリティー　　ローカル（デバイス側）で実行するため，機密情報の共有を回避し，ユーザーのプライバシーを保護できる。

⑶　電力効率　　クラウドサーバーとのデータ送受信時の電力消費を削減し，より効率的なデバイス運用を可能にする。

しかし，AIのモデル（入力から出力への複雑な関係を表現する多層ニューラルネットワーク）は，重み係数の数が膨大になるため，MCUへの実装には，モデルの圧縮が重要となる。

これに対して，東芝デバイス&ストレージ（株）は，MCU

向けAI開発環境の構築を進め，TT分解に基づく圧縮技術を開発した。この技術を導入することで，AIモデルの推論精度（以下，精度と略記）を維持しながら，モデルサイズと計算負荷の削減に取り組んでいる。

この論文では，まず，MCUにAIを実装するためのAI開発環境，及び従来のモデル圧縮技術に関して述べる。更に，TT分解に基づくモデル圧縮技術について述べる。

2. MCUへのAI実装のための開発環境

2.1 MCU向けAI開発手順

MCU向けAIの開発は，一例として，**図1**のように，以下の手順で進められる。

⑴ 適用対象の特定，モデルの選定，及び関連データの収集　　人物検出などのAIの適用対象を特定する。また，モデルを選定し，必要な関連データを収集する。

⑵ モデルの学習　　TensorFlow™などのオープンソースフレームワークを用い，重み係数や演算が浮動小数点形式であるモデルを生成する。

⑶ モデルの圧縮　　学習済みのモデルを圧縮する。

⑷ int8（符号付き8ビット整数）形式への量子化変換　　学習済みのモデルを，TensorFlow™のモデル変換用ツールであるTFLiteConverterを使って，MCUで利用可能なTensorFlow™ Liteモデルに変換する。

⑸ C/C++コードへの変換　　TensorFlow™ Liteモデルを，実装時にコンパイルできるように，microTVMやTensorFlow™ Lite for Microcontrollers（TFLM）などのオープンソースツールを用いてC/C++コードに変換する。

⑹ クロスコンパイル　　C/C++コードを，gccなどのコンパイルのツールでクロスコンパイルし，ターゲットMCUに書き込む。

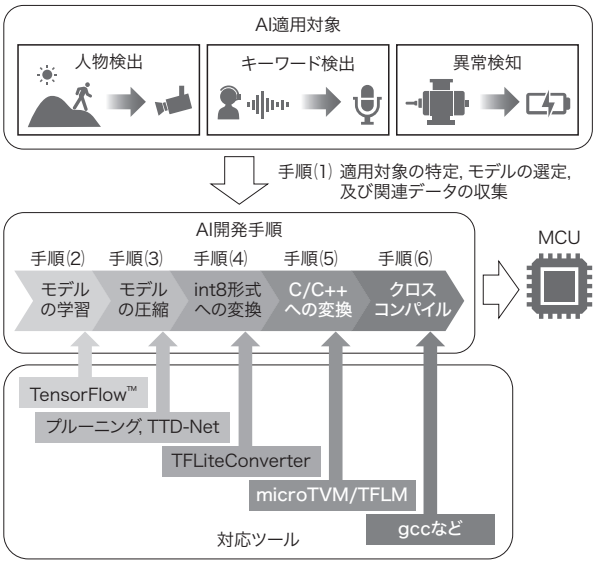

図1. MCU向けAI開発の手順と対応するツールの関係

各手順ではOSSツールを活用でき，当社のAI開発環境（ツールパッケージ）はこれらのツール利用手法などがまとめられている。

Relationship between AI development process and OSS tools in each step

2.2 MCU向けAI開発環境

MCU向けAI開発には，移植性，拡張性，効率性を支援する堅牢（けんろう）な環境が必要である。当社は，モデルの実装プロセスを統合し，OSSを活用して開発を支援するツールパッケージをAI開発環境としてサンプル提供する予定である。

この開発環境は，“学習”，“圧縮”，“コード変換”，“コンパイル”の四つの要素で構成され，用途に応じてアプリケーション開発ツールパッケージと圧縮ツールパッケージに分かれる。

2.2.1 アプリケーション開発ツールパッケージ

このパッケージは，MCU向けAI開発全般で使用するためのスクリプトファイルや，マニュアルを提供する。対象のサンプルアプリケーションと公開データベースを用い，マニュアルに従った操作により，モデルの学習からMCU上での推論プログラムの実行までについてAI開発を試行できる。

また，サンプルアプリケーションには，人物検出・キーワード検出・異常検知があり，ユーザーが自身の利用目的に近いパターンを選んでモデルを作成し，自身のモデルと置き換えて開発できるようになっている。

2.2.2 圧縮ツールパッケージ

このパッケージは，モデル圧縮に使用するスクリプトファイルやマニュアルを含み，リソース制約のある組み込みデバイスへモデルを展開するために用いる。ここでは，TensorFlow™により，モデルのメモリー使用量と計算負荷を削減しつつ，精度を維持することに重点を置く。モデルの圧縮とファインチューニング（既存の学習済みモデルを生かしながら，新しい学習データにあわせて一部の重み係数を微調整すること）を通じて，精度を維持しながらモデルサイズを縮小することで，組み込みデバイスでも高い性能を発揮する。

3. 従来のモデル圧縮技術

3.1 モデル圧縮の重要性

一般に，モデルは数百万の重み係数を持つことが多く，そのままMCUに組み込むのは難しい。メモリー使用量と計算能力に限りのあるMCUにモデルを展開するためには，圧縮は不可欠である。

3.2　従来の圧縮手法

従来の圧縮手法としては，プルーニング（Pruning）と知識蒸留（Knowledge Distillation）がある。

プルーニングは，不要な重み係数などを削除して，モデルのサイズと複雑さを削減する技術であり，特に組み込みシステムにおいて，効率性の向上と電力消費の低減に寄与する。個々の重み係数を削除してスパース（疎）な行列を作成する非構造化プルーニングと，重要度に基づき，フィルター，チャネル，又は層全体を削除する構造化プルーニングがある。構造化プルーニングはモデルの規則性を保ち，ハードウェアの負担を減らせるが，精度を維持するためにはファインチューニングが必要である。

また，知識蒸留は，大規模で複雑なモデル（教師モデル）の内容を，小さなモデル（生徒モデル）で学習することで，より小さく効率的なモデルを構築する手法である。

4. TT分解に基づくモデル圧縮技術

学習可能なTT分解ネットワークであるTTD-Net（Trainable Tensor Train Decomposition Network）を用いたモデル圧縮技術を，更なるモデルサイズ削減を目指して開発している。

4.1　TT分解ネットワーク

TT分解は，大規模な重み係数テンソルを一連の小さなTTに分解することで次元を削減する手法である。この手法は，重み係数の拡張性と，モデルの表現力を維持できるという利点がある。一方で，重み係数テンソルの再構成が不完全であることによる精度の劣化や，モデル構造に適応しないままTT分解を適用した場合，実行する浮動小数点演算の総量を示す指標であるFLOPs（Floating-Point Operations）の削減効果が限定的になる，という問題点がある。

そこで，この問題点を克服するTTD-Netを提案する。

4.2　TT分解の数式

d次元テンソル$\boldsymbol{A}$が与えられたとき，TT分解は，式(1)のように，それを低ランクのTTコアの連鎖に分解する[2]。

$$\boldsymbol{A}_{\{i_1,i_2,.....,i_d\}} = \boldsymbol{G}_1[i_1] \times \boldsymbol{G}_2[i_2] \times ..... \times \boldsymbol{G}_d[i_d] \tag{1}$$

$\boldsymbol{A}$ ：d次元テンソル（$\boldsymbol{A} \in \mathbb{R}^{(n_1 \times n_2 \times ... \times n_d)}$）

$\boldsymbol{G}_j$：次元$\mathbb{R}^{(r_{(j-1)} \times n_j \times r_j)}$のTTコア

（$r_0=1$，及び$r_d=1$

$r_1, r_2, ..., r_{(d-1)}$：TTランクであり，

圧縮率を制御するパラメーター）

このように分解した複数のTTコアに対して，モデル圧縮のための処理を施していく。

4.3　TTD-Netフレームワーク

TTD-Netは，体系的な層単位の圧縮パイプラインとして機能し，事前に学習済みの畳み込みニューラルネットワーク（CNN：Convolution Neural Network）を，低メモリー消費かつ高速推論に最適化された圧縮モデルへと変換する。このフレームワークにより，重み係数の数の大幅な削減，計算の高速化，そして最小限の精度低下の実現が可能となり，エッジデバイス上でのリアルタイムAI推論に最適なモデルが作成できる。

4.3.1　畳み込み層のTT分解

標準的な畳み込み層の重み係数は，式(2)のような4次元テンソル$\boldsymbol{M}$に格納されている。

$$\boldsymbol{M} \in \mathbb{R}^{(N \times C \times H \times W)} \tag{2}$$

$\boldsymbol{M}$：畳み込み層の重み係数（4次元テンソル）

N：出力特徴マップ（フィルター）の数

C：入力チャネルの数

H, W：カーネルの空間的なサイズ（高さと幅）

TTD-Netでは，この$\boldsymbol{M}$を適切な高次元形式に再構成し，式(3)のように四つのTTコアに分解する。

$$\boldsymbol{M} \rightarrow (\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3, \boldsymbol{A}_4) \tag{3}$$

それぞれのTTコアは，元の重み係数テンソルと異なる構成であるが，構造的要素は保持する（**図2**）。

4.3.2　選択的TTコア保持と再構成

TTD-Netでは，四つのTTコア全てを使用するのではなく，最初の二つのTTコア（$\boldsymbol{A}_1$と$\boldsymbol{A}_2$）だけを選択・保持し，$\boldsymbol{A}_3$と$\boldsymbol{A}_4$は破棄する。近似された重み係数テンソル$\hat{\boldsymbol{M}}$は，式(4)のように再構成される（図2）。

$$\hat{\boldsymbol{M}} = \boldsymbol{A}_1[N, r_1] \times \boldsymbol{A}_2[r_1, C, H, W] \tag{4}$$

ここで，r_1は最初の分解ステージ後におけるTTのランクである。この処理で，重み係数の数と乗算の複雑さが大幅に削減されるが，誤差（精度低下）が増える。

4.3.3　ファインチューニングによる誤差補償

破棄されたTTコアによる精度低下を補うために，保持されたTTコア$\boldsymbol{A}_1$と$\boldsymbol{A}_2$は，学習可能な重み係数をファインチューニングによって調整する。$\boldsymbol{M}$を教師モデル，$\hat{\boldsymbol{M}}$を生徒モデルとし，MSE（平均二乗誤差）に基づきファインチューニングした$\hat{\boldsymbol{M}}^{\text{ft}}$とすることにより，特徴表現が元の形に近い状態まで復元され，圧縮されたモデルが元のモデルに近い精度となる（図2）。

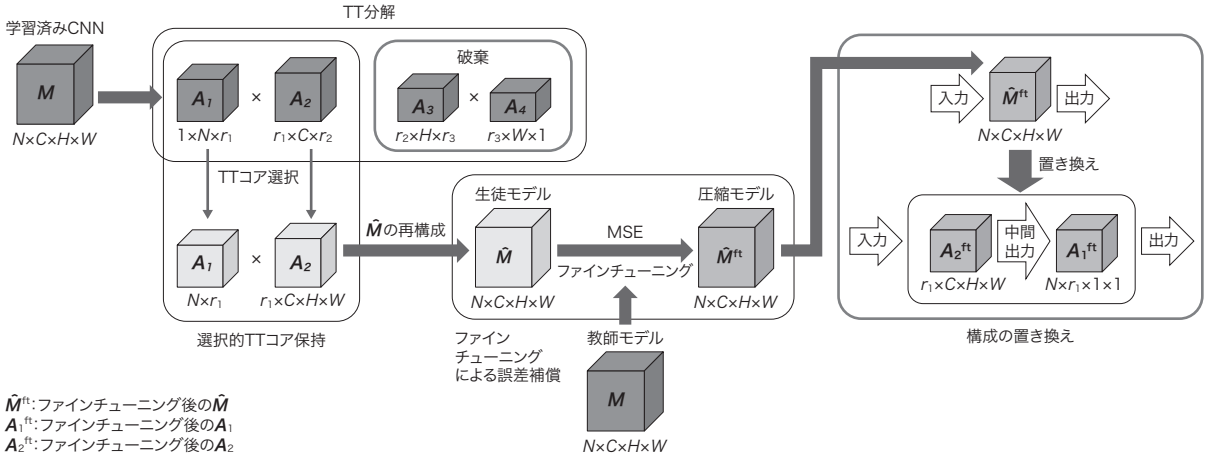

図2. TTD-Netのフレームワーク

学習済みCNNをTT分解後，重要度の高いTTコアを抽出し，ファインチューニングにより精度を復元して，圧縮したモデルを作成する。

TTD-Net framework

4.3.4 モデル構成の置き換え

ファインチューニング後，A_1は低次元の畳み込みカーネルA_1^{ft}として，A_2はカーネルサイズが1×1の畳み込みA_2^{ft}として実装される。この置き換えにより，重み係数が削減されることで演算量が減り，推論演算処理速度が向上する。この処理は，複数ある畳み込み層の全てに対して繰り返され，圧縮されたCNNが得られる。

4.4 評価結果

著名なニューラルネットワーク構造を実装したモデル定義の一つであるResNet-18アーキテクチャーに対して，検証用の画像データであるCIFAR-10データセットを用い，TTD-Netを評価した。ここでは，圧縮係数αの値を変えて，メモリーサイズの圧縮率，計算効率（FLOPsの削減率），精度とのトレードオフを検証した（**表1**，**図3**）。

αを増加させることで，圧縮率と計算効率は一貫して向上するが，一方で，精度が低下する。したがって，ターゲットとなる組み込みシステムの制約に合わせてαを選択する必要がある。

4.5 結論

TTD-Netを用いた圧縮は，TT分解，選択的TTコア保持と再構成，ファインチューニングによる誤差補償を統合することで，圧縮率，計算効率，精度維持のバランスを実現できる。

このフレームワークは，AI適用対象の要件に応じて，精

表1. TTD-Netを使用したモデル圧縮の評価結果

Performance evaluation results of AI model compressed by TTD-Net

α（圧縮係数）	使用メモリーサイズ（Miバイト）	圧縮率*1（%）	FLOPs（百万）	FLOPs削減率*2（%）	精度（%）	精度差分（低下）*3（ポイント）
ベースモデル	42.70	–	556.65	–	95.38	–
4	12.46	29.18	161.34	71.02	94.95	0.43
6	8.30	19.44	105.90	80.98	94.45	0.93
8	6.32	14.80	84.79	84.77	93.96	1.42
10	5.07	11.87	71.76	87.11	93.50	1.88
12	4.23	9.91	64.18	88.47	92.94	2.44
14	3.67	8.59	59.33	89.34	92.33	3.05
16	3.30	7.73	56.86	89.79	92.31	3.07
18	2.65	6.21	52.21	90.62	91.19	4.19

Mi：メビ（2^{20}）

*1：圧縮モデルの使用メモリーサイズをベースモデルの使用メモリーサイズで除算し，割合表示とした値

*2：ベースモデルのFLOPsから圧縮モデルのFLOPsを差し引いた値を，ベースモデルのFLOPsで除算し，割合表示とした値

*3：ベースモデルの精度から圧縮モデルの精度を差し引いた値

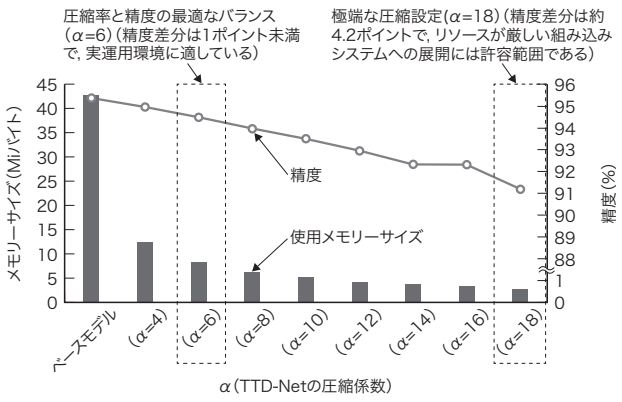

図3. TTD-Netのメモリーサイズ削減の評価結果

αの選定により，メモリーサイズと精度が変わるため，ターゲットの組み込みシステムに応じて，αを調整することが必要である。

Evaluation of relationship between TTD-Net compression coefficient and memory size

度低下を少なくすることを優先した圧縮を選択することや，精度低下を抑えつつ中程度の圧縮を選択することなどが可能であり，様々なAI適用対象に対して柔軟に対応可能なAI開発環境を提供できる。

5. あとがき

当社は，MCUへのAI実装の有用性に着目し，AI実装に必要な要素をAI開発環境として構築した。また，学習可能なTT分解ネットワークに基づくモデル圧縮技術を開発し，AI推論精度の維持と，モデルサイズ・計算負荷の削減を可能にした。更に，圧縮係数αの適切な選択によって，圧縮率・計算効率・精度を考慮した柔軟なAIモデルの開発が可能であることを確認した。

圧縮性・効率面・精度面において柔軟性を考慮したAI開発環境は，今後，増えていくMCUへのAI実装に不可欠であり，MCUの更なる利活用に貢献する。このようなMCUへのAI実装に備え，環境構築を進めていく。

文　献

(1) Gabor, M. et al. "Convolutional Neural Network Compression via Tensor-Train Decomposition on Permuted Weight Tensor with Automatic Rank Determination". ICCS2022. London, 2022-06, ICCS, p.654–667.

(2) Liu, X. et al. Tensor Decomposition for Model Reduction in Neural Networks: A Review. IEEE Circuits and Systems Magazine, 2023, **23**, 2, p.8–28.

村瀬　輝高　MURASE Terutaka
東芝デバイス&ストレージ(株)
半導体事業部 IC開発センター
Toshiba Electronic Devices & Storage Corp.

アジャイ カドルカル　Ajay KADOLKAR
東芝ソフトウェア・インド社
Toshiba Software India Pvt. Ltd.

久本　康司　HISAMOTO Koji
東芝デバイス&ストレージ(株)
半導体事業部 IC開発センター
Toshiba Electronic Devices & Storage Corp.