Chainer2MPSGraph:高速深層学習 モバイルアプリ作成のためのモデルコンバータ

泉 裕貴[1,a)]　堀田 大地[1,b)]　丹野 良介[1,2,c)]　柳井 啓司[1,d)]

概要

近年のモバイル端末の性能向上により, モバイル端末上で深層学習を用いた画像認識や画像変換を行うことが可能となってきた. しかし, 最近の深く複雑なネットワークを実装するためには, 長文のコードを書く必要があり手間がかかる. そこで本研究では, 高速実行可能な深層学習アプリの実装を容易にするコンバータを作成し, デモによってその有用性を実証する.

1. はじめに

iPhone を始めとするモバイル端末の性能は年々向上し高い演算能力を持つようになったため, 学習済みモデルを用いた深層学習の画像認識や画像変換をモバイル端末上で行うことが可能となってきた. しかし, 最新のネットワークは枝分かれのあるものやネットワークが深くパラメータ数が非常に多いものとなっており, 今後更に複雑なネットワークが出現することが予想される. このようなネットワークをモバイル端末上で実装するためには長文且つ難解なコードを書く必要があるために人手で書くことは困難である.

本研究では, 計算機上で既存の深層学習フレームワーク Chainer を用いて学習した際に生成されるモデルファイルから, ネットワーク部分の Swift 言語のコードとパラメータファイルを自動生成することができるコンバータ Chainer2MPSGraph を作成した. このコンバータを用いて iOS 端末上に高速実行可能な深層学習アプリを実装し, デモによってその有用性を実証する.

2. 関連研究

丹野らは, 学習済みモデルファイルを iOS 上で動作可能にする深層学習モデルコンバータ, Caffe2C(Chainer2C) を開発した [4]. これは, Caffe(Chainer) で学習した後に生成されるパラメータファイルから, モバイル端末でも実行可能な C 言語のコードを自動生成するものである. この方法では, 全ての学習パラメータが定数配列として C 言語コードに組み込まれ, 学習済みモデルファイルを実行するために必要なファイルを全て含んだ C 言語コードを生成する. また, 高速化の工夫として im2col 操作により畳み込み演算を行列積で計算できるようにし, 高速行列演算ライブラリである BLAS を利用することで SIMD(NEON) 命令, マルチスレッド処理を行っている. しかし, Caffe2C(Chainer2C) は iOS と Android 両対応であるため GPU は使用していない.

そこで本研究では, 高度なスキルを必要としないモバイル深層学習アプリの作成を可能とするコンバータの作成を目指す. また, iOS に特化することにより CPU よりも演算処理能力の優れた GPU を利用し, iOS 上での演算の高速化を図る.

3. 手法

本研究で作成したコンバータ Chainer2MPSGraph は, 学習済みモデルファイルを用いてアプリ作成に必要なネットワークコードとパラメータファイルを生成する. 学習済みモデルファイルには, 深層学習フレームワークの Chainer によって学習したものを用い, ネットワークコードは GPU による演算の高速処理に加え, 更なる効率化を図るためにグラフ構造を利用した MPSNNGraph API によるコードを生成する. 以下にネットワークコードとパラメータファイルの生成する流れを示す.

(1) 作りたい深層学習アプリの学習済みモデルを用意
(2) モデルファイルからネットワーク情報を取得
(3) (2) の情報から MPSNNGraph API を用いたコードを生成
(4) (2) の情報からパラメータファイルを生成

3.1 MPSNNGraph

本研究で作成したコンバータ Chainer2MPSGraph が生成するネットワークのコードは, Metal Performance Shaders(MPS) フレームワークの一部である MPSNN-

[1] 電気通信大学 大学院情報理工学研究科 情報学専攻
[2] 現在, NTT コミュニケーションズ株式会社勤務.
[a)] izumi-y@mm.inf.uec.ac.jp
[b)] horita-d@mm.inf.uec.ac.jp
[c)] tanno-r@mm.inf.uec.ac.jp
[d)] yanai@cs.uec.ac.jp

Graph API を利用している．MPS フレームワークは，GPU 上で画像処理や行列演算などを実行させる計算カーネルのライブラリであり，MPS を使用することで深層学習の GPU 実行を実現している．MPSNNGraph は，ニューラルネットワーク中の画像とフィルタノードが最適化されたグラフ表現である．グラフを構築した時に解析され，どのノードが入力に必要であり，どのノードが出力として生成されるかが決まる．これにより，出力として生成されるノードを計算するために必要でないノードは無視し，いくつかのノードはパフォーマンス向上のために他のノードと内部的に連結したグラフとなり，より一層の高速化が見込める．

3.2 Chainer2MPSGraph

Chainer で使用する画像などの入力やネットワークの中間出力などのデータは，単なるベクトルではなく，計算グラフを保持する Variable で構成する必要がある．この Variable という構造体を使用することによってネットワークで順伝搬計算を一度行うことで，ネットワークの各ノードはチェーンで繋がり，逆伝搬計算をすることが可能になる．

本研究では，この機能を利用して，一度順伝搬計算を行うことでネットワークの各ノードをチェーンで繋げる．これにより，ネットワークの出力から属性 creator を得ることでネットワーク最後のレイヤーがどのようなレイヤーであるかを知ることができる．そのレイヤーの inputs[0] を得ることでそのレイヤーへの入力 (1 つ前のレイヤーの出力) を得ることができる．同様に creator を調べてその入力を辿ることを繰り返すことでネットワーク全てを遡る．この時，レイヤーが畳み込み層や全結合層などの場合，inputs のリストに重みとバイアスのパラメータを抱えており，inputs[1] に重みのパラメータ，inputs[2] にバイアスのパラメータが格納してある．これを図解したものを図 1 として以下に示す．

このようにネットワークの出力からネットワークの最初のレイヤーまでを辿り，ネットワークの各レイヤーの種類やパラメータなどの情報をまとめて 1 つのリストとして作成する．そのリストを用いて Swift 言語のネットワークのコードとパラメータのバイナリファイルを作成する．バイナリファイルは，各層の重みとバイアスをそれぞれ分けてファイル化する．

3.3 アプリ作成の流れ

本研究のコンバータ Chainer2MPSGraph を用いた深層学習モバイルアプリの作成は以下の手順で行う．

(1) 深層学習フレームワーク Chainer を用いてネットワークを学習し，モデルファイルを作成
(2) Chainer2MPSGraph を用いて学習済みのモデルファイルから Swift 言語のコードとパラメータファイルを生成
(3) iOS 深層学習アプリの GUI 部分のコードを作成

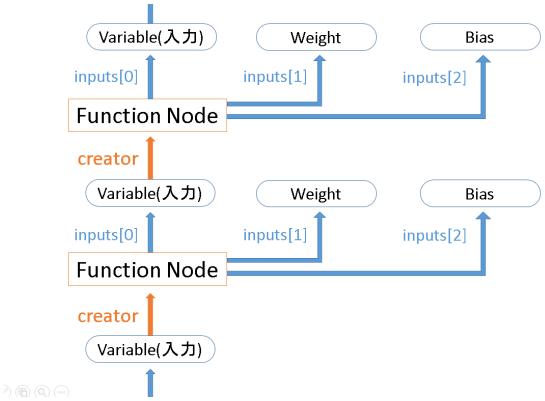

図 1 Chainer におけるネットワーク構造

(4) 生成した Swift 言語のコード，パラメータファイル，GUI のコードからアプリを作成

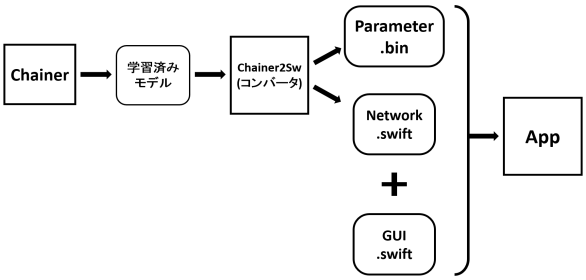

図 2 深層学習アプリの作成の流れ

4. 実験

Chainer2MPSGraph による実装の高速性を実証するために，演算処理を CPU のみで行う Chainer2C による実装と演算処理を GPU のみで行う Chainer2MPSGraph による実装を AlexNet の認識時間で比較を行った．また，演算処理に GPU を用いる実装方法における VGG16 の認識速度の比較も行った．この時，計測を 20 回行いその平均時間を認識時間とした．

4.1 結果

表 1 に Chainer2C と Chainer2MPSGraph との認識時間の比較を示す．演算処理を CPU のみで行った場合に比べて GPU のみで行った場合の方が約 3.7 倍の速度で実行可能という結果が得られた．また，表 2 に演算処理に GPU を用いる CoreML, MPS, Chainer2MPSGraph の認識時間の比較を示す．本研究のコンバータ Chainer2MPSGraph を利用して作成した MPSNNGraph を利用して実装した物体認識アプリが最も速い 108.99[msec] という結果となった．理由としては，CoreML がグラフ構造を利用していないためと考えられる．

5. アプリケーション

Chainer2MPSGraph の有用性を実証するために，画像変

表 1　AlexNet による物体認識の認識時間 [msec]

実装方法	平均認識時間
Chainer2C[9](CPU)	134.9
Chainer2MPSGraph(GPU)	**36.4**

表 2　VGG16 による認識時間の比較 [msec]

実装方法	平均認識時間
CoreML	144.87
MPS	155.21
Chainer2MPSGraph	**108.99**

換タスクから MultiStyleTransfer [5], 画像生成タスクから Conditional Cycle GAN [8] を iOS 端末上に実装し, 高速実行可能であることを示す.

5.1　MultiStyleTransfer [5]

ある画像のスタイルを他の画像に転写するものとして Gatys らの Imagestyle transfer using convolutional neural networks [1] がある. これは, 1 回の変換を行う毎に多大な時間を必要としたため, 事前に特定のスタイルを学習しておくことで, リアルタイムな画像変換を可能にした手法が提案された [2]. MultiStyleTransfer は, [2] の学習に用いた単一のスタイルでしか変換出来ない部分を拡張し, 複数のスタイル画像を任意の重みで合成し画像変換を行うことを可能にした.

図 3　MultiStryleTransfer による変換例 ([6] より引用)

5.2　Conditional Cycle GAN [8]

異なる 2 種類のドメイン変換を行う手法として CycleGAN [7] がある. この手法では, ドメインの 1 : 1 対応の変換しか行うことが出来ず, k 個のドメインがあった場合には k(k-1) 個の Generator を作成し学習を行わなければならない. そこで, 異なる複数のドメインに変換することが可能な Conditional Cycle GAN [8] が提案された. これは, [7] に Classification を補助タスクとして追加した ACGAN [3] を組み合わせることで実現している.

6.　おわりに

深層学習フレームワークの Chainer を用いて計算機上で深層学習の学習を行い, 生成されたモデルファイルからネットワーク部分の Swift 言語のコードとパラメータを格納したバイナリファイルを生成するコンバータ Chainer2MPSGraph を作成した. これにより, iOS 端末上に容易に深層学習アプリを実装することを可能にした. また, MPSNNGraph を用いて演算処理を全て GPU で行うことで, 演算処理を CPU で行う場合と比較して AlexNet において約 3.7 倍高速化を実現し, VGG16 において 108.98[msec] という高速実行可能ということを実験から示し, Chainer2MPSGraph の有用性を実証した.

図 4　Conditional Cycle GAN による生成例 ([8] より引用)

謝辞:　本研究は JSPS 科研費 17H01745/15H05915/17H05972/17H06026/17H06100 の助成を受けたものです.

参考文献

[1] Gatys, L. A., Ecker, A. S. and Bethge, M.: Image style transfer using convolutional neural networks, *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pp. 2414–2423 (2016).

[2] Johnson, J., Alahi, A. and Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution, *European Conference on Computer Vision*, pp. 694–711 (2016).

[3] Odena, A., Olah, C. and Shlens, J.: Conditional image synthesis with auxiliary classifier gans, *arXiv preprint arXiv:1610.09585* (2016).

[4] Tanno, R. and Yanai, K.: Caffe2C: A Framework for Easy Implementation of CNN-based Mobile Applications, *Proc. of International Workshop On Mobile Ubiquitous Systems, Infras- tructures, Communications, And AppLications (MUSICAL 2016)* (2016).

[5] Tanno, R., Matsuo, S., Shimoda, W. and Yanai, K.: DeepStyleCam: A Real-Time Style Transfer App on iOS, *International Conference on Multimedia Modeling*, Springer, pp. 446–449 (2017).

[6] Yanai, K. and Tanno, R.: Conditional Fast Style Transfer Network, *Proc. of ACM International Conference on Multimedia Retrieval (ICMR)* (2017).

[7] Zhu, J.-Y., Park, T., Isola, P. and Efros, A. A.: Unpaired image-to-image translation using cycle-consistent adversarial networks, *arXiv preprint arXiv:1703.10593* (2017).

[8] 堀田大地，成冨志優，丹野良介，下田　和，柳井啓司：大量の Twitter 画像を用いた Conditional Cycle GAN による食事写真カテゴリ変換，人工知能学会全国大会 (2018).

[9] 丹野良介，柳井啓司：CNN を用いた高速モバイル画像認識エンジンの自動生成フレームワーク，画像ラボ 4 月号 (2017).