ISSN 2186-5647

P-35

Puppet-CNNの高速化

日大生産工　○佐藤　遥香　　日大生産工　山内　ゆかり

1. まえがき

画像処理に用いられるConvolutional Neural Networks（CNN）は一般的に「train-and-use」方式を採用しており、ネットワーク構造が事前に定義され、学習後はカーネルパラメータが固定される。そのため入力データの複雑さは考慮されず、すべてのデータに対して同じ構造・パラメータが使用される。しかし、異なるデータは異なる複雑さを有するため、実際にはそれぞれのサンプルに応じた構造とカーネルで処理することが望ましい。また、CNNはネットワークを広く、深くするほど性能が向上する。しかしそれは同時にパラメータ数の爆発的増加を引き起こすため、モデルの保存や最適化が困難になるという欠点が存在している。CNNは人間の脳における情報処理を模倣して設計されているため、畳み込み層間で依存関係を持つべきである。しかし従来のCNNは層間の依存関係を無視し、各カーネルを独立に扱う設計がほとんどである。

これに対し、深さ適応型ニューラルネットワークやパラメータ適応型ニューラルネットワーク、モデル圧縮といった手法が提案されてきた。しかし深さ適応型手法は学習段階で大規模なモデルを学習するものの、テスト時にはその一部のブロックやパラメータしか使用しない、パラメータ適応型手法、モデル圧縮では入力データに対応したネットワーク構造となっておらず、性能の低下を招いてしまうなど、各手法において、一部の課題は解決されるが、一部の課題は依然として解決されないままとなっている。

そこでYucheng Xingらはモデルサイズを削減しつつ、カーネルの多様性を維持するためにPuppet-CNN[1]というCNNフレームワークを提案した。このフレームワークはpuppetモジュールとpuppeteerモジュールの2つのモジュールから構成される。このうちpuppeteerモジュールではNeural Ordinary Differential Equationsモデル[2]を使用しており、入力に応じた深さとカーネルを生成する。

Puppet-CNNは従来の深さ・パラメータ適応型ニューラルネットワークと比較し、性能と効率の両方で優れた結果を残したという報告がされている。しかしこの手法には入力ごとにパラメータを生成するため、処理に時間がかかってしまうという問題点がある。

本研究では，Puppet-CNNの問題点において連続畳み込み手法を組み合わせることで生成効率向上を提案し、提案手法と従来手法の処理速度、精度の比較し、その結果について報告する。

2. 従来研究

Puppet-CNN[1] はpuppetモジュール・puppeteerモジュールの2つのモジュールから構成されるフレームワークである。CNNモデルであるpuppetモジュールは入力データを直接処理するモジュールとなっている。puppetモジュールのカーネルはすべてpuppeteerモジュールによって逐次的に生成される。また、puppetモジュール、すなわちCNNモデルの深さもpuppeteerモジュールによって決定される。

puppeteerモジュールはNeural Ordinary Differential Equations [2]モデルとして構築されている。Ordinary Differential Equations (ODE)によって層のカーネルが前の層にもとづいて逐次生成されることで、カーネルパラメータ間に関連性が保たれ、人間の脳における情報処理を模倣している。

この2つのモジュールを使用することで、puppetモジュールの最適化は連鎖率を通じてpuppeteerモジュールの最適化へと変換される。これにより保存および最適化すべきパラメータはpuppeteerモジュールのもののみとなるため、従来と比べ少ないパラメータで済むようになる。

式(1)にpuppetモジュール内の畳み込みブロックのカーネルパラメータ$P_{l,F}$を求める式を示す。

$$P_{l.F} = P_{l-1,F} + \int_{l-1}^{l} G(P_{\tau,F})d\tau \qquad (1)$$

ここでG(·)はODEにおける微分関数であり、1層のDepthwise Separable Convolutional[3]のような非常に小規模なモデルでパラメータ化される。単純化のため、式(1)の積分はオイラー法を用いて近似し、式(2)のように書き換えられる。

$$P_{l.F} = P_{l-dl,F} + G(P_{l-dl,F})dl \qquad (2)$$

ここでdlはODEモデルの更新ステップサイズである。

Accelerating Puppet-CNN

Haruka SATO and Yukari YAMAUCHI

この方式により小規模なモデルのパラメータP_Gを用いて、大規模かつ深いCNN構造のパラメータP_Fを逐次生成でき、隣接する層のカーネルパラメータを相互依存的にすることが可能となっている。

しかし、一般的な畳み込み層ではパラメータ全体の次元は(C_{out}, C_{in}, K, K)であり、ここでKはカーネルサイズ、C_{out}、C_{in}は出力および入力チャネル数である。層ごとに必要なパラメータの次元は異なるが、Neural-ODEは1つの次元しか持つことができない。そのためODEの変数の次元を$(C_{out,max}, C_{in,max}, K_{max}, K_{max})$に設定する。ここで各次元はpuppetモジュール内の対応する最大値に等しい。また、生成時にはそれを$(C_{out,max}, C_{in,max}, K_{max} \times K_{max})$にリサイズする。各層lにおいてはODEモジュールの出力を平均プーリングにかけ、必要な次元$(C_{out,l}, C_{in,l}, K_l, K_l)$を持つ$P_{l.F}$を得る。

また、puppeteerモジュールはカーネルの生成だけでなく、入力データの複雑さに基づいたネットワークの深さも決定する。

式(3)に入力データの複雑さを測定するための情報エントロピー関数を示す。

$$E(X_0) = -\sum p(x_{0,i}) \log p(x_{0,i}) \qquad (3)$$

ここでX_0は入力データ、$x_{0,i}$は画像の画素値、p(·)はその確率である。式(4)の画素値だけでは入力の複雑さを十分に反映できないため、フーリエ変換によって得られる周波数マップ$Y(X_0)$に対しても情報エントロピーを計算し、両者を組み合わせて最終的な複雑さ$H(X_0)$を決定する。

式(4)に$H(X_0)$を求める式を示す。

$$H(X_0) = \frac{1}{2}E(X_0) + \frac{1}{2}E(Y(X_0)) \qquad (4)$$

この$H(X_0)$を用いてpuppeteerモジュールにおける初期値$P_{0.F}$および更新ステップサイズdlを推定する。以下に更新ステップサイズdlを求める式(5)を示す。

$$dl(H(X_0)) = \tanh(H(X_0)^{-1}) \qquad (5)$$

更新ステップサイズdlはODEにおけるパラメータ生成の刻み幅であり、層間のパラメータの変化率やネットワークの深さを制御する要素である。

また、以下にpuppetモジュールの層の深さDを求める式(6)を示す。

$$D(H(X_0)) = \left\lfloor \frac{1}{dl(H(X_0))} \right\rfloor \qquad (6)$$

puppeteerモジュールで生成されるカーネルパラメータはその都度特徴マップとの畳み込みに用いられる。そしてパラメータ適応は初期値$P_{0.F}$と更新ステップサイズdlによって決定される。

以下に初期値$P_{0.F}$を求める式(7)を示す。

$$P_{0,F} = \exp(1 - H(X_0)^{-2}) \qquad (7)$$

これはpuppetモジュールで決定されるカーネルサイズを持ち、同じ値で埋められたテンソルである。

Pupppet-CNN は puppet モジュールと puppeteerモジュールが分離可能であるため、任意のCNNをpuppetモジュールとして使用することが可能となっており、puppeteerを通して最適化することができる。また、puppeteerモジュールはNeural-ODEモデルであるため、逆伝搬の際に中間の出力値を全て保持する必要が無く、省メモリ化が実現可能である。さらにpuppeteerモジュールの逆伝搬はODEを解くことと同義であるため、従来のCNNよりも計算コストが少ない。加えて従来の「train-and-use」型CNNとは異なり、パラメータ生成・生成モデルで処理、という2段階手段になっている。これにより入力データの複雑さに応じてpuppetモジュールの構造やパラメータを動的に調整が可能である。

3. 提案手法

Puppeteer モジュールにおけるパラメータ生成における生成時間短縮のため、連続畳み込み手法を組み合わせることによって精度向上を目指す。

4. まとめ

従来手法に対して、高速化を図るべく連続畳み込み手法を組み合わせることを提案した。

参考文献

1) Yuchang Xing and Xin Wang, “PUPPET-CNN: Input-Adaptive Convolutional Neural Networks with Model Compression using Ordinaly Differential Equation”, arXiv preprint arXiv: 2411.12876 (2024).

2) Francois Chollet. ” Xception: Deep Learning with Depthwise Separable Convolutions”, arXiv preprint arXiv: 1610.02357 (2016).

3) Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”, arXiv preprint arXiv: 1704.04861 (2017).