5ZB-08

モバイル端末上におけるプライバシ保護可能な画像認識モデル構築手法の提案

近藤 華 †　山口 実靖 ††　神山 剛 †††　小口 正人 †
† お茶の水女子大学　†† 工学院大学　††† 長崎大学

1 はじめに

近年，深層学習技術や画像認識技術が Android などのスマートフォンを始めとするモバイル端末で一般的に利用されるようになった．顔認識，物体認識，ジェスチャー認識などが例として挙げられる．また，モバイル端末の高性能化も進んでいる．ハイエンド端末では，CPU のみならず GPU や NPU など行列計算を始めとした高負荷な処理を行うことが可能な高性能プロセッサが搭載されるようになった．これに伴って，モバイル端末内で端末内のデータを使用し端末内専用の画像認識システムが作られるようになった．現状この仕組みを実現する手法として，転移学習技術を用いる手法や端末のデータを外部のサーバに送信し，演算させた結果を受け取る手法が存在する．しかし，これらの手法にはそれぞれ精度があまり高くない，データの内容を外部のサーバに知られるという欠点が存在する．そこで，本稿ではモバイル (Android) 端末内のみでファインチューニングを行いモデルの精度を高める，プライバシに配慮した画像認識モデルを構築する手法の提案を行う．

2 関連技術

2.1 ファインチューニング

汎用的な学習済みモデルをベースモデルとして用い，そのベースモデルに特定の画像認識タスクを行うための出力層を追加し学習させ，その後，出力層と一部の中間層の重みを再学習させる手法である．

2.2 Android Neural Networks API

Android デバイス上で演算負荷の高い機械学習処理を実行するために設計された Android C API である [1]．この API を使用することで，学習処理実行時に Android 端末上の利用可能なより良い高性能プロセッサを利用することができる．

2.3 MobileNet

モバイルや組み込みアプリケーション向けの画像認識用小型 CNN である [2]．畳み込み層において深さ方向とチャネル方向で分離して演算を行うことで，パラメータ数や計算量を削減している．

3 実装方法の検討

3.1 モデルの構造

ファインチューニングのベースモデルとしては MobileNet [2] を ImageNet [3] で学習したものを使用する．MobileNet を使用する理由は，軽量でパラメータ数も比較的少なく学習速度も比較的速いことが示されているモデルであり Android Neural Networks API で実装可能な構成であることから選定した．また，ベースモデルに追加する出力層は，Global Average Pooling 層，Dropout 層，全結合層で構成する．

3.2 対象とする画像認識タスク

本稿では，犬と猫の判別を行う画像分類タスクを行う．また再学習に使用するデータとしては大きさ 160×160 の画像を各種類 100 枚ずつ使用する．100 枚と少ないのは再学習に使用する全てのデータはモバイル端末内のデータであり，モバイル端末内で負荷なく保存可能なデータ量でなければならないためである．

3.3 予備実験

モバイル端末上でのファインチューニングの際に再学習する層と重みを固定する層を決定するため予備実験を行った．この予備実験はモバイル端末上ではなく PC 上で行った．PC 上で実験を行った理由は，PC 用のプログラムは現在 Tensorflow 等のライブラリが存在し，比較的容易に実装が可能なためである．使用したデータセットは”Dogs vs. Cats” dataset [4] であり，このデータセットのうち Dog(犬) の画像 100 枚，Cat(猫) の画像 100 枚を再学習用データとして使用し，再学習用データとは別の Dog の画像 500 枚，Cat の画像 500

Proposal of a Method for Building Privacy-Protected Image Recognition Models on Mobile Devices
†Hana Kondo
††Saneyasu Yamaguchi
†††Takeshi Kamiyama
†Masato Oguchi
†Ochanomizu University
††Kogakuin University
†††Nagasaki University

枚を精度測定に使用した．バッチ数は 4 に設定した．また，学習回数は出力層の学習で 10 エポック，出力層と再学習する層の学習で 10 エポックとした．
転移学習のみを行う手法 (方法 1) とファインチューニングを行う手法 (方法 2) を比較する実験を行った．それぞれ図 1 に矢印と点線で示している層から出力層までの再学習を行い，再学習後のモデルにおける精度を測定した．図 1 における A の層，B の層とは MobileNet(図 2) の A の層と B の層に対応している．
実験結果は表 1 に示す通りになった．この結果より，再学習用のデータが各種類 100 枚と少ない枚数でも，方法 2 の手法である畳み込み層も再学習するファインチューニングを行う手法により精度の良いモデルを構築可能であることが示された．

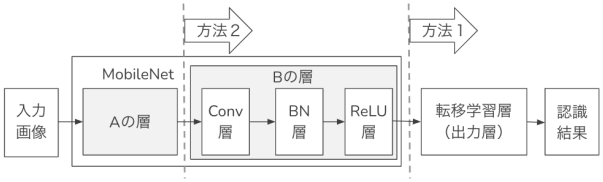

図 1: 検証実験時のモデルの概要

	Type / Stride	Filter Shape	Input Size
A	Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
	Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
	Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
	Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
	Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
	Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
	Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
	Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
	Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
	Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
	Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
	Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
	Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
	$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
	Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
	Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
	Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
	Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
B	Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
	Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
	FC / s1	1024×1000	$1 \times 1 \times 1024$
	Softmax / s1	Classifier	$1 \times 1 \times 1000$

図 2: MobileNet の構造 [5]

表 1: 再学習する層を変化させた際の精度の比較

	方法 1	方法 2
精度 [%]	91.00	97.50

4 まとめと今後の課題

本稿では，モバイル (Android) 端末向けのプライバシに配慮した精度の良い結果がえられる個人用の画像認識モデルを構築する手法として，モバイル端末上で畳み込み層も再学習するファインチューニングを行う手法を提案した．
今後の課題は，モバイル端末上で実行可能なモデルの実装とアプリケーションの作成を行うことである．
モバイル端末上で実行可能なモデルの実装では，3.3 章にて決定した手法と同様の実装を行う．Android 端末内の高性能プロセッサを活用するため，重みを固定する層には Android Neural Networks API を利用する．また実装後に学習速度の計測を行い，実装方法やモデルの改善を行っていきたいと考えている．
アプリケーションの作成は，上記のように実装したモデルがモバイル端末上で実用可能か調査するために行おうと考えている．

参考文献

[1] Android Neural Networks API — Android Developers, https://developer.android.com/ndk/guides/neuralnetworks.(最終アクセス 2020/01/05).

[2] 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. CoRR, abs/1704.04861, 2017.

[3] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. Int. J. Comput. Vision, 115(3):211–252, December 2015.

[4] ”Dogs vs. Cats” dataset, https://www.kaggle.com/c/dogs-vs-cats/data(最終アクセス 2020/01/05)

[5] [2] の Table 1 より転載