モバイル端末上での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と実装

近藤　華　（指導教員：小口 正人）

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．加えて，モバイル端末に搭載される保存領域も増加傾向にある．これによりユーザはモバイル端末内に以前より多くのデータを保存することが可能になった．
上記のような理由から，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送信す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．本稿では特に，入力画像のデータ量が少ない場合でもファインチューニングを行えるかどうかに着目する．

2　関連技術

2.1　NNAPI(Android Neural Networks API)

NNAPI とは 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，に演算処理を実行する際に 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．

2.1.1　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．ファインチューニングの転移学習との相違点は，転移学習では追加した出力層のみ再学習を行うが，ファインチューニングでは Batch Normalization 層や畳み込み層の再学習を行うという点である．
画像認識タスクにおける転移学習とファインチューニングの精度の比較を行なっている先行研究では，ファインチューニングがより精度が高いことが示されている [2].

2.2　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 [3]．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．また，MobileNet の構造は NNAPI で対応可能な層のみで構成されている．このような特性から本研究では MobileNet の構造を使用することにした．

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 を ImageNet [4] で学習したものを使用する．また，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．

4　評価実験

4.1　実験 1

本実験では，少数の再学習用データのみを使用するという条件下にて，ファインチューニングが転移学習と比べ精度向上に貢献可能かの確認を行った．

4.1.1　実験概要

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 [5] 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 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．

4.1.2　実験結果

実験結果を表 1 に示す．
方法 1，方法 2 ともに殆ど収束するまで学習させた．従って方法 1 よりも方法 2 の方が精度が高くなることが確認できる．

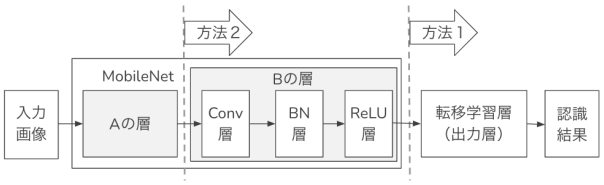

図 1: 検証実験時のモデルの概要

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

図 2: MobileNet の構造 [6]

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	95.99

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．

4.2 実験 2

本実験では，4.1 の方法 2 の手法を採用した際に，再学習を行わない層での演算を Android 端末上で行う際に NNAPI を使うことで実行速度の向上が見られるかどうかの確認を行った．

4.2.1 実験概要

TensorFlow により再学習を行わない層のみのモデルを作成し，そのモデルを TensorFlow Converter を使用して TensorFlow Lite 形式に変換した．その後，そのモデルの実行速度のベンチマークを測定した．
ベンチマークの測定には，ベンチマークアプリケーション [7] を使用した．また，ベンチマークの測定を行った端末は，Pixel4，Pixel5，POCO F2 Pro の 3 種類である．それぞれの端末の性能は表 2 に示す．

表 2: 実験 2 の際に使用した端末の性能

	Pixel4	Pixel5	POCO F2 Pro
OS	Android 12	Android 11	Android 10
SoC(Snapdragon)	855	765G	865
メモリ	6 GB	8 GB	8 GB

4.2.2 実験結果

実験結果を表 3 に示す．

表 3: ベンチマークの測定結果

Model number	Pixel4	Pixel5	POCO F2 Pro
NNAPI 未使用時	16203.4	16697.2	13664.8
NNAPI 使用時	5660.83	13826.7	6593.78

いずれの端末でも，NNAPI 使用時の方が未使用時より実行速度が速くなることが確認できた．
この結果に基づき，モバイル端末上で再学習を行わない層の演算を行うモデルの実装を行った．Tensorflow Lite を使用し，NNAPI デリゲートを設定することで NNAPI を使用するようにした．

5 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．
また，モバイル端末上で実行可能なモデルの実装において，重みを固定する層では Android Neural Networks API を利用すると学習速度向上することを示した．
今後の課題は，Android 端末上で実行可能なモデルの実装を行うことである．4.1 章の方法 2 の手法に従い，再学習を行う部分の実装も行う．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．

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] Xiang, Qian, et al. ”Fruit image classification based on Mobilenetv2 with transfer learning technique.” Proceedings of the 3rd International Conference on Computer Science and Application Engineering. 2019.

[3] Howard, Andrew G., et al. ”Mobilenets: Efficient convolutional neural networks for mobile vision applications.” arXiv preprint arXiv:1704.04861 (2017).

[4] Russakovsky, Olga, et al. ”Imagenet large scale visual recognition challenge.” International journal of computer vision 115.3 (2015): 211-252.

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

[6] [3] の Table 1 より転載

[7] パフォーマンス測定 — TensorFlow Lite https://tensorflow.google.cn/lite/performance/measurement(最終アクセス 2020/01/05)