Fine Tuning による学生の受講態度の推定

吉　武　春　光

1．まえがき

ディープラーニングの世界は進展が早い。ここ数年で、吉武（2019）で実験に使用したディープラーニング・フレームワークChainerの開発が終了し、フレームワークPytorchへ移行した。PyTouchには、Chainer同様に学習済モデル（名称pretrained-models.pytorch）が組み込まれていたが、栄枯盛衰の激しい機械学習の世界では更新が追いついていないところがあった。Ross Wightman（ロス・ワイトマン）という開発者が積極的に学習済みモデルライブラリtimm（PyTorch Image Models）を開発して提供している。

筆者は、吉武（2019）において画像データに対して 深層学習（Deep Learning）を使って学生の授業態度の推定を行った。実験時に、学習用データの数が少ないかもしれない、という問題が浮き彫りになった。そこで整形した画像データの用意に多大の時間を要することを痛感した。更に、時間をかけて用意をした割には、認識率が向上しないことも経験した。

一方、世の中で提供されている学習済モデルに対して、条件などを変えて追加学習させることで、高い認識率が得られることが知られている。Fine Tuningとか転移学習とか呼ばれている手法である。

そこで、本研究では、timm中で既に提供されている学習済モデルに対してFine Tuningを行うことで、どの程度の認識率が得られるのかを調べることにした。使用した画像データや画像分類の要件は吉武（2019）と同じとした。

２．PyTorch に備わっている学習済モデルと timm

2021年に Resnet の性能が良いと思っていたが、2022年時点では EfficientNet の性能がより良いようだ。しかし、PyTorch 標準の pretrained-models.pytorch は更新が止まっており（2018年まで）、最新のモデルに追従できていないところがあった。

そこで、開発者：Ross Wightman ロス・ワイトマンさんが PyTorch用ライブラリ PyTorch Image Models（通称 timm）を提供している。2022年12月23日時点で学習済モデルは 770 も提供されている。次の Python スクリプトを使えば、学習済モデル数と学習済モデル一覧が表示される。

```
import timm
len(timm.list_models(pretrained = True))
 -->学習済モデル数を表示
from pprint import pprint
pprint(timm.list_models(pretrained = True))
 -->学習済モデル一覧を表示
```

timm の学習済みモデルは ImageNet に加えて更に５つのデータセットを使って学習/検証したものである。詳しくは timm-bench/results/README.md に載っている。吉武（2019）で述べたように、画像認識の競技会 ILSVRC で使われていたのが ImageNet である。ImageNet は2009年に構築された大規模なカラー写真の教師あり画像データベースである。1400万枚以上のカラー写真（教師ラベルは２万カテゴリー）の画像データが無料でダウンロードできる。ImageNet では，各クラスにつき500枚～1000枚の画像が，256×256 のサイズで用意され，合計約14Million（1400万枚）の画像から構成されている。クラスごとにサンプル数がある程度均一化されており，出力ラベル数のバランスがとれたデータセット分布から，識別モデルを学習できるようになっている。ImageNet で行われた画像認識タスクでは、1000クラスの物体を識別している。そこで、timm で提供される学習済みモ

デルも 1000クラスを識別するものである。ImageNet を使った競技会 ILSVRC は 2017年に終了しており、最近では ImageNet の問題を改良したデータセットが幾つも発表されている。

timm の使い方は公式には timmdocs に記載されているが、Hughes (2022) が参考になる。ImageNet を使った実験で測定した各学習済モデルの認識率は pytorch-image-models/results/results-imagenet.csv に載っている。認識率の差異は僅かであるが、モデル選択の参考になる。

timm の基本的な使い方は、基本的に import timm をした後に timm.create_model という関数を呼ぶだけである。

timm.create_model という関数を呼ぶ際に、引数として model_name に使いたいモデルの名前を指定して「pretrained=True」とすると、pretrained model がダウンロードされて読み込まれる。但し、ImageNet が基準になっているので、出力は1,000次元になっている。それ以外の任意のクラス数の分類をやりたい時は、num_classes という引数で出力クラス数を指定すると、その出力クラス数に対応じた head に自動的に置き換えてくれる。

3．Fine Tuning と転移学習

Fine Tuning と転移学習 は、共に、既に提供されているモデルを利用して更に追加として学習を行う手法のことである。この両者の定義は明確ではないが、一般には、次のように言われており、Fine Tuning の一部が転移学習であるとも言える。

Fine Tuning	学習済みモデルの層の重みを微調整する手法のこと。学習済みモデルの重みを初期値とし、再度、学習することによって微調整する。
転移学習	学習済みモデルの重みを固定し、追加した層のみを使用して学習する。

さて、timm の場合であるが、解説ページ Hughes (2022) には Fine Tuning という言葉しか出て来ないので、timm を使う時のパラメータの指定は Fine

Tuningを行っていると見なせるであろう。

４．timmを使ったFine Tuning

吉武（2019）におけるデータを使用してtimmのFine Tuningを行った。条件は吉武（2019）と同じにしたので、４分類である。動作環境も吉武（2019）と同じ筆者のサーバであり、GPUとして性能11.34TFLOPSのNVIDIA GTX 1080 Tiを使用した。但し、Pythonやライブラリのバージョンだけが新しくなっている。

4.1　データの準備

今回は吉武（2019）で使用した画像データを、そのまま使用することにした。但し、吉武（2019）ではデータ拡張（Data Augmentation）というテクニックを使って画像数を増やしたがtimmにはデータ拡張Data Augmentation機能が組み込まれている。Data Augmentationは色々な種類があり、どの程度、行うのかに任意性がある。timmでは既に用意されているCubuk et al.(2019)の「RandAugment, an automated data augmentation method」を使うことにした。

画像データのサイズに関しては、吉武（2019）では2^7=128ピクセルとしたが、今回はImageNetと同じ2^8=256ピクセルとした。

準備したデータは、実験プログラムのディレクトリ下のimage_classificationというディレクトリの中に次の構造として入れた。

図１　本実験のデータ構造

train ディレクトリには訓練用データを、test ディレクトリにはテスト用データを入れた。枚数は次の表の通りである。

表１　使用した画像データの枚数

配置ディレクトリ	ラベル	枚数
image_classification/train/looking-ahead	a	126
image_classification/train/looking-down	d	97
image_classification/train/sleep	s	99
image_classification/train/touching-pda	p	141
image_classification/test		120

4.2 Training

訓練に用いる script に関しては Hugging Face の Training Script が参考になった。例えば、次のようにコマンドを入力した。

```
python3 train.py image_classification/train --pretrained --model
resnet101d --num-classes 4 --epochs 150 --aa rand-m9-mstd0.5 -b 32 --opt
adam --warmup-epochs 5 --weight-decay 1e-4 --sched cosine --reprob 0.4
```

ここで、パラメータの詳細は https://github.com/rwightman/pytorch-image-models/blob/main/train.py に載っているが、主要なものを列挙しておく。

表２　train.pyの主要引数

引数名	説明
第１引数	トレーニング用データのディレクトリ
--pretrained	学習済モデルを使用することを指定する。
--model	学習済モデルの名前を指定する。
--num-classes	出力クラス数を指定する。
--epochs	繰り返し学習回数
--aa	データ拡張を指定する。
-b	バッチサイズ

--opt	optimiserの関数を指定する。
--warmup-epochs	ここで指定した回数で学習率を変える。
--weight-decay	過学習の抑制手法「Weight Decay」の値を指定する。
--sched	使用する学習率スケジュールを指定する。
--reprob	Random Erasingの値を指定する

ここでwarmupとschedというのは、学習率に関するパラメータである。まず、Deep Learningの学習を行う際に、過学習させてしまうと良くないが、学習率を低いままだと収束に時間がかかったりする問題がある。そこで、学習が進むに従って学習率を下げていく「学習率 スケジュール」という機能が実装されるようになってきた。これが、パラメータ schedである。パラメータ warmupというのは、最初の方の学習では低い学習率を使い、学習が進むに従って学習回数を徐々にあげていくという学習率のコントロール方法のことである。

パラメータ reprobは、データの拡張方法Random Erasingの値を指定する。これは画像中に四角な領域を別データで埋める手法である。timmdocs のRandom Erasing Data Augmentationに詳しく載っている。

トレーニング結果はoutputディレクトリのtrainサブディレクトリ内に、更に

解析日付－解析時間－モデル名

というディレクトリが作られて、その中に蓄えられる。

図２　出力ディレクトリの構造

この 解析日付ー解析時間ーモデル名 ディレクトリの中は、次に図示したように timm 独自の pth.tar という圧縮形式で蓄えられている。一番認識率が高かった結果は model_best.pth.tar で保存されている。従って、inference.pyで指定する解析結果は、このmodel_best.pth.tarを指定する。

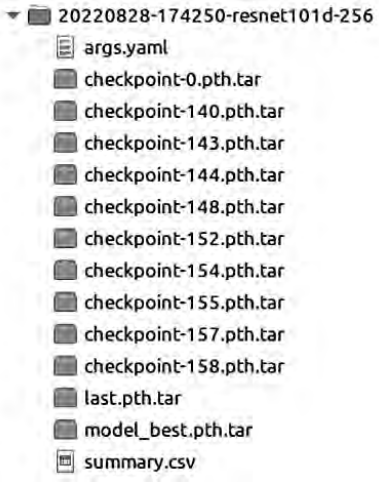

図３　解析結果ディレクトリの構造

この解析結果ディレクトリ内の summary.csv に各 epoch ごとの train_loss や eval_loss が記録されている。

epoch	train_loss	eval_loss	eval_top1	eval_top5
0	1.345971371	1.190778849	70.19438453	100
～～～				
155	1.189249524	0.959537847	64.57883373	100
156	1.170842239	0.960197061	63.93088557	100
157	1.20855758	0.956714809	64.79481646	100
158	1.18719548	0.959211986	65.01079918	100
159	1.166735802	0.95544942	63.93088557	100

図４　summary.csvの内部（一部）

4.3 Validation/Inference

Fine Tuning が終われば、いよいよテストデータを使った推論を行う。この推論に用いる script に関しても Hugging Face の Training Script が参考になった。例えば、次のようにコマンドを入力した。

```
python3 inference.py image_classification/test --model resnet101d
--checkpoint output/train/20220828-174250-resnet101d-256/model_best.
pth.tar --num-classes 4
```

ここで、パラメータの詳細は https://github.com/rwightman/pytorch-image-models/blob/main/ validate.py に載っているが、主要なものを列挙しておく。

表３　inference.pyの引数

引数名	説明
第１引数	テスト用データのディレクトリ
--model	学習済モデルの名前を指定する。
--checkpoint	保存されている pth.tar圧縮形式を指定する。
--num-classes	出力クラス数を指定する。

inference.pyの処理結果は、topk_ids.csvというファイルに出力される。

A	B	C	D	E
01010549.jpg	0	3	2	1
01010736.jpg	1	3	0	2
01010751.jpg	0	3	1	2
01010752.jpg	0	3	2	1
01010756.jpg	0	2	3	1
01010803.jpg	0	1	2	3

図５　topk_ids.csvの内部（一部）

topk_ids.csv の１カラム目はファイル名である。２カラム目（列B）は認識率が最も良かった場合の結果 Top1 である。以下、３カラム目（列C）は２番目に良かった場合の結果 Top2、４カラム目（列D）は Top3、５カラム目（列E）は Top4 である。各カラムの値は、各出力ラベルになっている。本実験では次の通りである。

表４　本実験での出力カラムとラベルとの対応関係

カラムの値	ラベル	状態
0	a（looking-ahead）	顔が正面
1	d（looking-down）	顔が下を向いている
2	s（sleeping）	寝ている
3	p（touching-pda）	スマホを操作している

4.4　実験結果

次のページに各モデルごとの認識率の表が載っている。

pytorch-image-models/results/results-imagenet.csv

本研究では、Resnet-D を試すことにした。パラメータの値は無限に近い組み合わせが考えられるが、今回の本研究の目的に照らして代表的な値を使用した。epoch数、つまり訓練（学習）の回数は色々と試したが50もあれば十分なようであった。実験は、最適化関数 optimiser として、adam, adamp, sgd を使用して行った。テストデータ枚数120の認識結果を次に示す。

表５　認識結果

モデル名	最適化関数	間違った枚数	認識率（%）
Resnet-D	adam	14	88.3
	adamp	22	81.7
	sgd	11	90.8

認識率は約90%であった。吉武（2019）での認識率は50%であったので、かなり精度が出ていることになる。

最も認識率が良かった optimiser=sgd の場合の推定結果と正解が異なっている画像を図６に示す。

図６　推論を誤った例

なお、画像の一部では目の付近を、ぼかして掲載している。

4.5　実験結果の検討

推定結果と正解が異なっている画像を整理していると吉武（2019）と同様に、次の傾向が読み取れる。

・スマホが小さいため、上手くスマホを捉えているかどうかの検討が必要である。
推論を誤った例の 01010736.jpg、08131118.jpg、19042247.jpg などのように、手の先に白い物体があると、スマホを触っている「ラベル p」と判断されてしまう傾向があった。一方、08131221bad.jpg や 08181136.jpg のように、スマホを触っているのだが正面を向いている場合には「ラベル a」と判断されてしまう場合があった。

・正解ラベルを付与した基準そのものが不明確な場合がある。
前出 08131221bad.jpg や 08181136.jpg の場合は、前を向いてスマホを触っているので、「ラベル a」と「ラベル p」を同時に付与するのが本当であろう。timm が使用している ImageNet の問題点としては、アノテーション（＝ラベル付け）に関するものが有名である。具体的には、ImageNet では「１つの画像＝１つのラベルしか付与しない」仕様となっているが、画像内に複数の対象が映り込んでいるケースがある。今回の実験では、まさしく、この問題に行き当たったと思われる。Yun et al.(2021) などが ImageNet の拡張を行っている。

５．あとがき

本研究では、既存の学習済モデルを利用して Fine Tuning させる方法に挑戦した。結果は、データとなる画像を追加することなく、より高い認識率を得ることができた。

画像を用意するのに労力を要すること、学習に用いる高性能な GPU に資金が必要なこと、学習に多大な時間が要すること、などを考えると、個人の研究者レベルでは、既存の高性能な学習済モデルを利用して追加学習さ

せる Fine Tuning などの方法が効率が良いと思われる。実際、既存の高性能な学習済モデルの開発は巨大な研究機関が行っている。

本研究と吉武（2019）で浮かび上がったラベル付けの問題を解決するために、今後は１つの画像に複数のラベルを付与する研究に挑戦したい。

本研究は、科学研究費平成28年度（2016年度）基盤研究（C）（代表：菅沼　明）「学生の振る舞い検出による授業雰囲気の推定に関する研究」で行った研究を継続しているものである。

参考文献

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, Quoc V. Le (2019)：
"RandAugment: Practical automated data augmentation with a reduced search space", arXiv:1909.13719

Hugging Face の timm docs：
https://huggingface.co/docs/hub/timm, 2022.12.22 アクセス

Hugging Face の Training Script：
https://huggingface.co/docs/timm/training_script, 2022.12.22 アクセス

Chris Hughes (2022)："Getting Started with PyTorch Image Models (timm): A Practitioner's Guide", https://towardsdatascience.com/getting-started-with-pytorch-image-models-timm-a-practitioners-guide-4e77b4bf9055, 2022.12.22 アクセス

ImageNet：https://cvml-expertguide.net/terms/dataset/image-dataset/imagenet/, 2022.12.22 アクセス

timmdocs：https://timm.fast.ai/, 2023.1.10 アクセス

timm-bench/results/README.md：https://github.com/MadryLab/timm-bench/blob/master/results/README.md, 2022.12.22 アクセス

timmdocs の Random Erasing Data Augmentation：
https://timm.fast.ai/RandomErase, 2023.1.10 アクセス

PyTorch Image Models (通称 timm)：https://github.com/rwightman/pytorch-

image-models, 2022.12.22 アクセス

pretrained-models.pytorch：https://github.com/Cadene/pretrained-models.pytorch, 2022.12.22 アクセス

pytorch-image-models/results/results-imagenet.csv：https://github.com/rwightman/pytorch-image-models/blob/main/results/results-imagenet.csv, 2022.12.22 アクセス

Sangdoo Yun, Seong Joon Oh, Byeongho Heo, Dongyoon Han, Junsuk Choe, Sanghyuk Chun (2021)："Re-labeling ImageNet: from Single to Multi-Labels, from Global to Localized Labels", https://arxiv.org/pdf/2101.05022

吉武春光 (2019) : "深層学習を用いた学生の受講態度の推定", 西南学院大学商学論集, Vol. 65, No. 4, pp. 215-236, 2019.3月