深層学習を用いた学生の受講態度の推定

吉　武　春　光

西南学院大学商学論集
第65巻　第4号　抜刷
2019（平成31）年3月発行

深層学習を用いた学生の受講態度の推定

吉　武　春　光

1．まえがき

筆者は、吉武(2018)、吉武(2016)において言語データに対して 深層学習 (Deep Learning) を使った研究を行い、深層学習の手法の有効性を確認した。今回は、深層学習の手法を画像認識に用いることにした。

今回の実験の目的は、学生の授業態度を連続撮影し、それを解析することにより、その撮影時の授業雰囲気を自動検出できるようにすることにある。第２章で述べるように、ディープラーニングを使った手法の性能が高そうなので、今回は、それを採用することにした。また、処理対象には、動画を、連続する画像の連続として解析することも考えられるが、まずは、ある学生の、ある瞬間の画像を解析し、その学生は、どういう状態にあるかを自動推定することにした。つまり、本研究はディープラーニングの分類問題ということになる。

2．画像認識の研究の推移

本章では、画像認識の研究動向を概観する。

2.1 画像認識研究の動向

まず、画像処理という研究分野は内容が多岐に渡るために、本稿では、後で述べる研究目的に関係した画像認識研究に絞って、研究動向を述べる(河原,原田2016)。

2000年初頭までは、SIFT等の特徴ベクトルと，SVM 等の識別器を組合せ

た手法が主流となっていた。その手法は職人芸的に設計されており汎用的とは言い難かった。その後、画像認識の研究においては2010年より、世界的に大規模な画像認識の競技会 ILSVRC(ImageNet Large Scale Visual Recognition Challenge)が始まった。この競技会では、画像分類や物体検出などのタスク（競技項目）が与えられ、参加したチームがエラー率の低さを競うものである。但し、毎年、タスクの内容は変更される。参加チームは100を超える。2011年までは画像分類のエラー率が30%ぐらいであったが、2012年に優勝したカナダのトロント大学のチーム（筆頭著者 Hinton の名前をとって AlexNet と呼ばれている）がエラー率 15.3% を出して注目を浴びた。AlexNet が使った手法が深層学習の畳み込みニューラルネットワーク（Convolutional Neural Networks: CNN）であった。そのために以後の画像処理の深層学習の研究では CNN が必ず使われている。2017年に優勝した中国のチームのエラー率は 2.3%であったが、これは人間が目視により画像分類を行った場合のエラー率5% を下回っていると言われている。

2.2 ディープラーニングを使った画像処理の基本事項

本節ではディープラーニングを使った画像処理における基本事項を述べる。

2.2.1　クラス分類(Classification)と物体認識(Detection)

ある画像が与えられたときに、予め決めておいたカテゴリー（クラス）に分けることをクラス分類(Classification)と言う。プログラムに予め、画像とクラスのペアを学習させておいてから、クラスが不明な画像をプログラムが処理して正しいクラスに分類できるようにするということである。そのために、「教師あり学習」と呼ばれている。

2.2.2　既存の学習済モデルを利用して追加学習させる方法

学習済みモデルが公開されている場合は、その学習済みモデルを利用して、追加学習をさせる転移学習(Transfer learning)、及び、ファインチュー

ニング(Fine tuning)という２つの手法がある。出力層付近以外を再学習させないのが転移学習で、出力層付近以外を再学習させるのがファインチューニングである。

2.2.3　ゼロから画像を新規学習させる方法

ディープラーニングでは、ビッグデータと呼ばれるぐらいの多くの画像を学習させないと認識率が上がらないと言われている。しかし、具体的に画像数が何枚ぐらい必要かどうかは目的とするタスク次第である。本研究では、授業風景を撮影することによって得られる画像と、Web に置いてある画像から得た合計数百枚の画像を用いて試みることにした。

2.3 フレームワークの選定

ディープラーニングのプログラムは複雑であり、すべてを自力で実装するのは大きな負担になる。そこで、一般的には、ディープラーニング用のフレームワークを利用してプログラムの実装を行っている。ディープラーニング用のフレームワークとしては何種類も開発されているが、プログラミング言語 Python を中心にして、高速化が必要な箇所はプログラミング言語 Cを組み合わせている場合が多い。なお、ディープラーニングにおいては、定形かつ大量の演算を並列にパイプライン処理する目的のために GPU (Graphics Processing Unit) を利用することが多い。どのディープラーニング用のフレームワークであっても対応している。

フレームワークとして有名なものとしては次の３つがある。

・TensorFlow

Google 社が開発したもので、世界で最も使われていると思われる。

・Caffe

UC Barkeley で画像認識をターゲットに開発されたもの。環境構築が面倒なので取り組み難いが、学習済みモデルが多数、公開されている。

・Chainer

日本の Preferred Networks 社が開発したもの。但し、オープンソースソ

フトウェアになっているので一般利用者も開発に参加できる。ドキュメントは英語である。プログラミング言語 Python のみで利用できる。可視化ツール ChainerUI や、分散処理パッケージ ChainerMN なども開発されている。本研究では、日本製を応援する意味合いもあって、Chainer を採用することにした。

2.3.1 Chainer Colab Notebooks

Google は、完全にクラウド上で実行される Jupyter ノートブック環境 Colaboratory を無償で提供している。Colaboratory は GPU (性能 4.4TFLOPS)にも対応しているが、メモリの制限があったり、順番待ちで利用できないこともある。Chainer のChainer 公式ドキュメント中の Chainer Colab Notebooks はColaboratoryを使って Chainer を実行できるようにしてある。Jupyter ノートブックは、Colaboratoryを使わずに、自分のサーバ上で動作させることも出来る。

2.3.2 Chainer Begginer's Hands-on

Chainer Colab Notebooks の中の Chainer Begginer's Hands-on は、初めて Chainer を使用する場合に参考になる。本研究では、その中の Creating and training convolutional neural networks を理解し拡張する形で研究を進めることにした。

2.3.3 ChainerUI

ChainerUI は Chainer上の実験中の学習ログの可視化や実験管理機能を追加するパッケージである。2019年1月公開のバージョン 0.8.0 では、更に、計算結果の画像を表示する機能も追加されている。但し、本論文では、設定が間に合わなかったので、学習ログの可視化だけを使用している。

2.4 畳み込みニューラルネットワークCNNとは

画像の認識においては、画像のビット列を認識していくために画像の位

置や形が少し変っただけで上手く認識されないという現象が出る。人間の場合は、画像の周りも含めた全体を見るので、多少のずれは問題にならない。そこで、フィルタと呼ばれる小領域を画像の端からスライドさせ、各々の時点のフィルタ内の画像ビット列を１つの特徴量として圧縮し(これを「畳み込み」と言う)全体として１つの畳み込み層(Convolution Layer) と呼ばれる層を作成する仕組みが考案された。これは脳の視覚神経系における受容野をモデル化していると言われている。畳み込み層は図形の特徴を抽出するためのもので、これに続くプーリング層(Pooling Layer)で位置ズレを吸収する。CNN の全体像は、例えば、木田(2017)が解り易い。畳み込み層とプーリング層を交互に繋げ、最後に目的に合わせて全結合層を接続する。全結合層では、入力画像のピクセルを一列に並べることになるので、画像データの元の形状が持っている本質的な特徴を見落としてしまう可能性がある。それに対して、畳み込み層は形状を維持するため、正しく理解できる可能性がある。　ILSVRC で優秀な成績を収めたモデルは、次の２つの傾向が見られる(内田,山下2017)。

・フィルタを小さくし、階層を深くする。
・プーリング層や全結合層を少なくする。

2.4.1 ストライド(stride)とゼロパディング(zero padding)

　ストライド(stride)とは上記のフィルタをスライドさせる移動幅のことである。ゼロパディング(zero padding)とは、画像の端の領域に 0 を追加することにより、画像の端が畳み込みされない現象を避ける手法のことである。単にパディング(padding)とだけ言われることもある。

　CNNでは、次の４つのパラメータを設定する必要がある。

フィルタの数(K):大体は2の累乗の値がとられる(32,64,128...)
フィルタの大きさ(F):
ストライド(S):フィルタを移動させる幅
パディングの幅(P):画像の端の領域をどれくらい埋めるか

ここで、層をつなぐことになるのだが、
フィルタ適用後の層の幅＝(N+P×2－F)/S+1　　式1
の関係が成立している必要がある。
そして、畳み込み層を幾つか、つないでいき、最後は、活性化関数で出力する。

2.4.2 勾配法(勾配降下法)

勾配法とはパラメータの極値を見つけるための手法である。例えば２次元関数 $Y=aX^2+bX+C$ でＹの値のピークを見つけるようなものである。但し、ディープラーニングで使われる関数は、数百次元の関数なので、グラフを描いてピークを見つけることはできない。そこで、関数上の適当な１箇所を選び、その少しだけ前後の値が、それよりも下がっているかどうかを調べる。全てが下がっている場合は、その１箇所はピークであろうと思われる。但し、２こぶラクダのようにピークが１箇所とは限らない。その「少しだけ前後」を見るための係数を学習率(重み)という。学習率が小さいと正確に判断できるが、計算回数が多くなる。学習率が大きすぎると、行ったり来たりで収束しない。更には、最適解ではない所に局所解が存在しているかもしれないので、しらみつぶし的に、最適解を探していく必要がある。

何通りもの勾配法が提案されているが、有名なものとしては Adam, AdaGrad, AdaDelta, RMSpropGraves, SGD, MomentumSGDなどがある。

2.4.3 最適化 (Optimization)

ディープラーニングにおいて、数多くあるパラメータを重みという。ディープラーニングにおける最適化 (Optimization) とはモデルの予測値と実際の値との誤差から重みを更新することであり、そこで上記の勾配法が使われる。

2.4.4 Batch Normalization

Batch Normalization は 2015年に Sergey Ioffe とChristian Szegedy が提案した手法である(Ioffe, Szegedy 2015)。過学習を抑制し、ネットワーク全体を安定化させる手法として広く利用されている。

2.4.5 Dropout

Dropout は、過学習を防ぐ手法として知られている。しかし、Li et al.(2018) による Batch Normalization と Dropout を併用すると認識率が上がらないという報告もある。

2.4.6 Weight decay

Weight decay (荷重減衰) も、過学習を防ぐ手法として知られている。しかし、Ilya Loshchilov, Frank Hutter (2017) によると、Weight decay を使う場合は、勾配法 Adam と SGD を比べると Adam との組わせでは汎化性能が良くないと報告されている。

2.4.7 過学習(Overfitting)

過学習とは、特定の学習データに最適になるように学習し過ぎたため、未知のデータに対する誤差（汎化誤差）が逆に上がってしまった症状のことである。学習させる画像データが少ない場合や、記述したネットワーク・モデルが複雑すぎる場合に発生しやすい。

2.4.8 ディープラーニングの手順

ディープラーニングのプログラミングを行うには、次を実装する必要がある。

・ネットワークを記述する。
・ネットワークの順伝搬と逆伝播を実行する。
・勾配法を使用してパラメータ(重み)を最適化する。

しかし、本研究で使用する Chainer はネットワークの逆伝播を自動的に実行してくれる。

機械学習をさせるときに使用するデータの集合をデータセットと呼んでいるが、機械学習を行うためには、そのデータセットを３つに分けておく必要がある。

・訓練(Training)データ　学習のために使用する。
・検証(Validation)データ　学習した結果を評価するために使用する。
・テスト(Test)データ　訓練にも検証にも使っていないデータを使って評価する。

機械学習は訓練と検証を繰り返し行うので、訓練データと検証データに過学習した状態になる可能性がある。そのために、訓練と検証に使用したことのないテストデータを使って、真の能力を評価する。

2.5 幾つかのCNNモデルの概要

CNN はパラメータ数は膨大であるので、似たモデルを参考にして、自分の研究用にパラメータを決めていくのが効率的である。本研究では、画像を3~4クラスに分類する予定である。

2.5.1 The CIFAR-10 dataset

The CIFAR-10 dataset というのは、60000 枚の画像（縦32bit x 横32bit）を 10 クラスに分類してあるデータ集のことである（各クラス内の画像数は6000枚である）。前出の ILSVRC は1000クラス分類という巨大なものであるので、余り参考にはならないが、The CIFAR-10 dataset を処理しているモデルは本研究に最も似ていると言える。前出の Chainer Colab Notebooks の中の Creating and training convolutional neural networks は CIFAR 分類が使われているので参考になる。そこではシンプルなモデルと複雑なモデルが設定してある。なお、Chainer の公式サンプルにある CIFAR 分類の例は既存のモデルを読み込んで追加学習を行う仕様になっているので、今回は参考にしなかった。但し、本研究では、3.2 節で後述するが、クラス分類として「スマホを操作している」状態を識別させたいので、画像の bit 数を大きくして 縦128bit x 横128bit とする。そのために、前出の Chainer Colab

Notebooks の中の Creating and training convolutional neural networks 中で使われた CIFAR 分類のモデルで妥当かどうかの検討が必要になる。

2.5.2 Chainer Colab Notebooks の中の Creating and training convolutional neural networks 中で使われた CIFAR 分類の例のシンプルなモデル

このシンプルなモデルは、3つの畳み込み層と2つの全結合層から成っている。この例ではCIFAR画像からランダムに 縦28bit x 横28bit に切り出した画像を入力としている。入力画像はカラーなので、入力画像の bit 数は３である。

表１　CIFAR 分類の例のシンプルなモデルの概要

	層の種類	フィルタの数	フィルタの大きさ (F)	フィルタの移動幅 (S)	パディング (P)	活性化関数
1	畳込層	32	3	3	1	ReLU
2	畳込層	64	3	3	1	ReLU
3	畳込層	128	3	3	1	ReLU
4	全結合層					ReLU
5	全結合層					なし

検証データセットの精度は 60% 程度までしか上がっていない。モデルが学習データにオーバーフィッティングしていると思われる。

2.5.3 VGG

2014年の ILSVRC において２位の成績を収めた VGG (Simonyan, Zisserman 2014)はモデルがシンプルであるので参考になる。VGG のモデルの設定表は次の通りになっている。VGG の基本的な考え方は、フィルタのサイズを小さくするが、その代りに層を深くするということであった。

表２　VGGのモデルの概要

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

2.5.4 「Chainer Colab Notebooks の中の Creating and training convolutional neural networks 中で使われたCIFAR 分類の例の複雑なモデル

このモデルは、８つの畳み込み層と３つの全結合層から成る。VGG の設計方針を真似たものと思われる。この全ての層で活性化関数はReLU を使用している。この例ではCIFAR画像からランダムに 縦28bit x 横28bit に切り出した画像を入力としている。入力画像はカラーなので、入力画像の bit 数は３である。モデルの概要を次の表３に示す。BatchNormalization を BN と、Dropout を DO と、Max Pooling を MAXPL と略記している。

表３　CIFAR分類の例の複雑なモデルの概要

		フィルタの数	フィルタの大きさ (F)	フィルタの移動幅 (S)	パディング (P)	
1	畳込層	64	3	1	1	+BN+ReLU
2	畳込層	64	3	1	1	+BN+ReLU+MaxPL+DO
3	畳込層	128	3	1	1	+BN+ReLU
4	畳込層	128	3	1	1	+BN+ReLU+MaxPL+DO
5	畳込層	256	3	1	1	+BN+ReLU
6	畳込層	256	3	1	1	+BN+ReLU
7	畳込層	256	3	1	1	+BN+ReLU
8	畳込層	256	3	1	1	+BN+ReLU+MaxPL+DO
9	全結合層					+ReLU
10	全結合層					+ReLU
11	全結合層					

３．解析対象データと解析実験

本研究では、フレームワークChainerを使って畳み込みニューラルネットワークを作成し、学生の授業態度を撮影した画像を学習させてから、授業態度を推定する実験を行った。

なお、筆者はGPUを搭載した筆者のサーバ上でJupyterノートブックを動かした。筆者のサーバはGPUとして性能11.34TFLOPSのNVIDIA GTX 1080Tiを搭載している。

3.1 画像データのラベル付

ディープラーニングの分類問題では、『画像と、その画像に対する正解ラベル』というペアを数多く用意し、それをあるモデルに学習させた後、新たな画像のラベルを推定することになる。本実験では、学生の授業態度を表４の４つのラベルに分類することにした。

3.2 画像データの入手

ディープラーニングに用いる画像データはビッグデータと呼べるほど多量が望ましいが、学生の授業風景を撮影した画像を多量に用意するのは難しい。本実験では、Webの検索エンジンを用いて学生の授業風景を撮影した画像を探し、更に、実際に筆者の授業風景を撮影することで多数の画像

を得た。予め、受講生に撮影と計算機処理の承諾を得ている。なお、「スマホを操作している」状態の画像は皆無であったために、発想を転換し、eラーニングシステム Moodle の小テストをスマホで受験させるようにし、半強制的に「スマホを操作している」学生の画像を撮影することが出来た。

撮影は、最初、ドライブレコーダーを用いて動画を撮影し、その動画から静止画像を切り出した。しかし、画質的に満足の行くものではなかったので、４Kビデオカメラを使用するように変更した。また、併せて、デジカメでの撮影も追加した。得られた画像は、一枚の画像に複数学生が写っているものがほとんどだったため、学生が写っている部分だけを人手により切り出し、表４に掲載した枚数を使用した。

表４　使用した画像データの枚数(拡張前)

状態	ラベル	枚数
顔が正面	A (looking-ahead)	151
顔が下を向いている（何か書いている）	D (looking-down)	100
スマホを操作している	P (touching-pda)	135
寝ている	S (sleeping)	86

3.3 画像データの品質とデータ・クレンジング(Data Cleanging)

機械学習で使用する画像データに関する品質とは、「人間が判断できないような画像は品質が悪い」ということである。人間が判断できる画像であれば、多少ボケていても、ズレていても、品質に問題はない、ということである。そして、生データから品質の悪いデータを取り除くことをデータ・クレンジング(Data Cleanging)と言う。注意すべきは、極めて品質の良い画像だけで学習したら、いろいろなバリエーションのある本番画像の認識率が悪くなる可能性がある、ということである。画像認識においては、認識対象の画像がきれいに写っているものだけとは限らず、一部が隠れていたり、角度が悪かったり、かすれていたりする。したがって、本番データに強くするためには、生データとしては、いろいろなシチュエーションの画像を入手すべき、ということになる。これを外乱や障害に強いという意味でロバスト性が高いと言う(麻生2013)。

3.4 画像データのサイズ

撮影した画像中の学生の像はサイズがバラバラであるが、機械学習させるためには、すべての画像をある一定のサイズに揃えなければならない。

画像サイズを決定する際に考慮したことは２つある。

1)　画像に写っている学生の状態を判別しやすいかどうかである。画像サイズが小さすぎると学生の姿勢や目線などが読み取りづらい恐れがあり、一方で大きすぎると、収集した画像のうちの、もともと小さい画像が強制的に引き延ばして拡大されることになり、ぼやけたような画像になってしまう。中村(2018)によると一辺が60 から80 ピクセル程度が最適であると判断しているが、本実験では、スマホを操作しているかどうかという些細な判断が必要となるため、それよりも大きなピクセル数にすることにした。

2)　畳み込みニューラルネットワークのプーリング層の数をどれくらいに設定するか、という点である。

プーリングを適用する領域サイズとストライドには同じ値を設定するのが一般的であるため、本研究ではプーリング層のフィルターサイズとストライドを２に設定した。式１を用いて計算すると、画像をプーリング処理するたびにサイズが1/2 になる。例えば、縦64 ピクセル横64 ピクセルの画像をプーリング処理すると、縦32 ピクセル横32 ピクセルの画像になる。サイズが奇数になっている画像にプーリング処理を行うことはできない。そのため、一つの画像に何度もプーリング処理を行うためには、データセットの画像サイズが2 の累乗の倍数になっていることが望ましい。CNN にN 個のプーリング層を組み込んだ場合、データセットの画像サイズは2 のN 乗の倍数である必要があるため、例えば、CNN に４つのプーリング層を組み込んだとすると、画像サイズは2^4=16 の倍数、すなわち32x32 ピクセル、48x48 ピクセル、64x64 ピクセル…などのようになっていなければならない。プーリング層の数は、後に述べるランダムサーチによる実験によって決定するが、3 層から5 層を見込んでいる。画像サイズが2^5=32 の倍数であれば3 層から5 層のいずれの層数であっても問題ない。本研究では2^7=128 ピクセルの場合を試すことにした

3.5 データ拡張（Data Augmentation）

元の学習データに変換を加えてデータ量を増やすテクニックのことである。例えば、次がある。

・変形

拡大縮小,反転(左右/上下),回転,シフト(水平/垂直),

トリミング(Random Crop),部分マスク(CutoutやRandom Erasing)

・変色

コントラストを調整,明るさを調整（ガンマ変換）

・平滑化（平均化フィルタ）

・ノイズを増やす

・背景を差し替える

データ拡張の目的としては、単にデータ量を増やすというだけではなく、先ほど述べたロバスト性を高めるという意味合いもある。注意点としては、本番データを意識したデータ拡張をすべきであることと、前述した過学習にならないようにすることがある。本研究では、撮影の際に明るさ、ノイズ、背景などが異なっていること、更には、画像切り出しの際に部分マスクを行った画像もあることを踏まえ、左右反転とトリミング(Random Crop)を採用することにした。左右反転することで２倍の枚数になり、更にトリミングすることで更に２倍の枚数になった。

3.6 実験データ生成

2.4.8 項で述べたように、３種類のデータセットを作る必要がある。本研究では、まず、全データセットからランダムに抜き出した 20% をテストデータセットとした。そして残った 80% に対してデータ拡張を行い４倍の枚数を得た。そして、Chainerのプログラム中で、その中から更に20% を検証データセットとし、80% を訓練データセットに分割した。

表５　使用した画像データの枚数(実験に使用)

ラベル	原画像枚数	テスト枚数	訓練＋検証枚数
A (looking-ahead)	151	30	484
D (looking-down)	100	20	320
P (touching-pda)	135	17	276
S (sleeping)	86	27	432

3.7 実験（1）

まず手始めとして2.5.4項と同じモデルで試行することにした。８つの畳み込み層と３つの全結合層で構成する。但し、入力画像は 縦128bit x 横128bitである。入力画像はカラーなので、入力画像のbit数は３である。

	層	フィルタの数	フィルタの大きさ (F)	フィルタの移動幅 (S)	パディング (P)	
1	畳込層	64	3	1	1	+BN+ReLU
2	畳込層	64	3	1	1	+BN+ReLU+MaxPL+DO
3	畳込層	128	3	1	1	+BN+ReLU
4	畳込層	128	3	1	1	+BN+ReLU+MaxPL+DO
5	畳込層	256	3	1	1	+BN+ReLU
6	畳込層	256	3	1	1	+BN+ReLU
7	畳込層	256	3	1	1	+BN+ReLU
8	畳込層	256	3	1	1	+BN+ReLU+MaxPL+DO
9	全結合層					+ReLU
10	全結合層					+ReLU
11	全結合層					

なお、実験ではChainerが用意してくれているcuDNNのautotune機能を使用するように設定した。

```
chainer.cuda.set_max_workspace_size(512*1024*1024)
chainer.config.autotune = True
```

まず、最適化関数として Adam を使用してパラメータを色々と変えながら試行した。最も良かった結果を図１に示す。収束が一定ではないことが読み取れる。なお、main/accuracy は訓練(学習)の際の精度であり、val/main/accuracyは検証の際の精度である。（以下、同様）

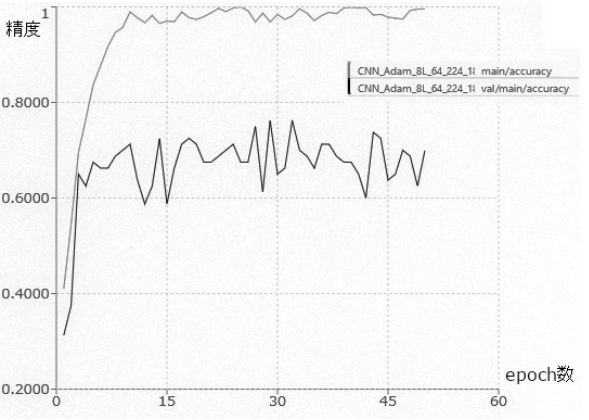

図１　Adam使用の８層の結果

ここでepoch数とは、訓練(学習)の回数のことである。

次に、最適化関数としてMomentumSGDを使用してパラメータを色々と変えながら試行した。最も良かった結果を図２に示す。

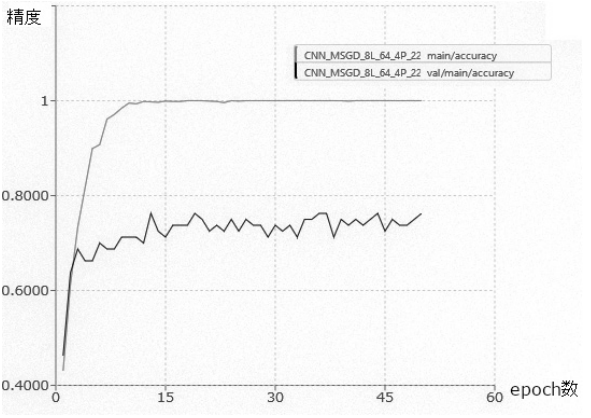

図２　MomentumSGD使用の８層の結果

これを見ると、最適化関数として MomentumSGDを使用した場合の方が、上手く収束しているようである。

3.8 実験（2）

次に層の数を減らして実験を行った。３つの畳み込み層と１つの全結合層から構成する。但し、入力画像は縦128bit x 横128bit である。入力画像はカラーなので、入力画像のbit 数は３である。

		フィルタの数	フィルタの大きさ (F)	フィルタの移動幅 (S)	パディング (P)	
1	畳込層	32	3	1	1	+BN+ReLU+MaxPL+DO
2	畳込層	32	3	1	1	+BN+ReLU+MaxPL+DO
3	畳込層	64	3	1	1	+BN+ReLU+MaxPL+DO
4	全結合層					

最適化関数として MomentumSGD使用してパラメータを色々と変えながら試行した。最も良かった結果を図３に示す。

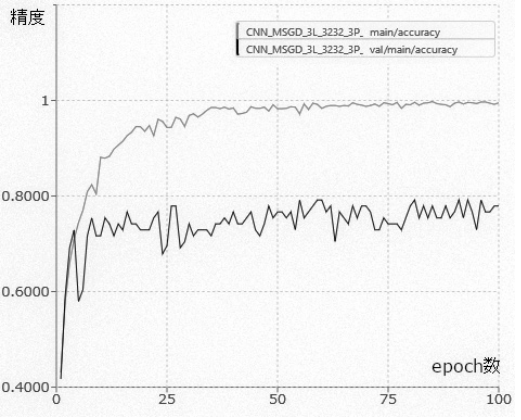

図３　MomentumSGD使用の３層の結果

3.9 実験結果の検討

この図３の結果に対して、テストデータセットを用いて検証を行った。その結果の精度は 50% であった。次に、推定と正解が一致した画像を図４に示す。

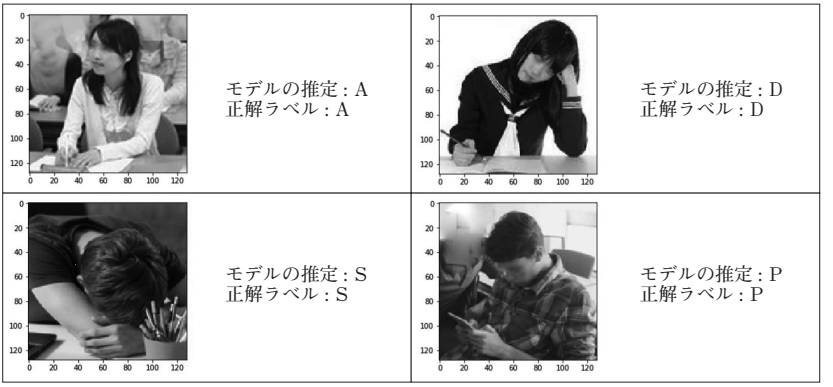

図４　モデルが正しく推定した例

次に、推定と正解が異なっている画像を図５に示す。

図５　モデルが推定を誤った例

なお、一部の図では目の付近を、ぼかして掲載している。

推定と正解が異なっている画像を整理していると次の傾向が読み取れる。
・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。
・顔が写っていない場合は識別が難しいように思える。
・スマホを操作している箇所が小さいため、畳み込み層が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。

本研究で使用したデータ数が少ないために、層の数を少なくしたが、精度は僅かしか向上しなかった。

また、中村(2018) では、ほぼ同じ画像枚数を使って、[顔が正面]、[顔が下を向いている]、[寝ている]という３クラス分類を行い、もっと良い精度を出している。今回、新たに追加した [スマホを操作している] というクラスが、他の[顔が正面]と[顔が下を向いている]のクラスと似ていることが、本実験の精度が上がらなかった原因かもしれない。

４．あとがき

本研究では、フレームワーク Chainer を使って畳み込みニューラルネットワークを作成し、学生の授業態度を撮影した画像を 1512枚、訓練(学習)＋検証させてから、新規の学生の画像97枚に対して授業態度を推定するという実験を行った。結果の精度は 50% であった。精度の向上のためには、画像数を増やすことが必要と思われる。

実験をしていて一番、困ったのは、処理の途中の段階を可視化していないので、正しい処理をしているのかどうかの判断がつかないことであった。折しも、本論文の執筆中に ChainerUI の新バージョン 0.8.0 がリリースされた。これには、処理の途中の段階を可視化して表示する機能が追加されている。是非とも処理途中の可視化を行い、処理が正しく行われていたのか、層の構成が妥当かどうか、などの検討を進めたい。また、2.2.2項で述べた既存の学習済モデルを利用して追加学習させる方法にも挑戦したい。

畳み込みニューラルネットワークは流行りであるが、フレームワーク

Chainer を使えば、こんなに簡単に実現できることが分かったことは、今後の研究に向けて意欲が高まる。

筆者の専門は自然言語処理であるので、今後は、自然言語処理に対するニューラルネットワーク処理へも挑戦したい。

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」の一貫として行ったものである。

参考文献

Chainer 公式ドキュメント, https://docs.chainer.org/en/stable/, 2019.1.10アクセス

ChainerUI, https://github.com/chainer/chainerui, 2019.1.10アクセス

Chainerの公式サンプルにあるCIFAR分類の例：
https://github.com/chainer/chainer/tree/master/examples/cifar, 2019.1.10アクセス

Karen Simonyan, Andrew Zisserman (2014)“Very Deep Convolutional Networks for Large-Scale Image Recognition” arXiv:1409.1556 [cs.CV] (Sep 2014).

Ilya Loshchilov, Frank Hutter (2017)“Decoupled Weight Decay Regularization” arXiv:1711.05101 [cs.LG] (Nov 2017).

ILSVRC, http://image-net.org/challenges/LSVRC/, 2019.1.10アクセス

Ioffe, Sergey・Szegedy , Christian (2015)“Batch Normalization: Accelerating

Deep Network Training by Reducing Internal Covariate Shift" arXiv:1502.03167 (cs) (Feb 2015).

The CIFAR-10 dataset：https://www.cs.toronto.edu/~kriz/cifar.html, 2019.1.10 アクセス

Xiang Li, Shuo Chen, Xiaolin Hu, Jian Yang (2018)"Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift" arXiv:1801.05134 [cs.LG] (Jan 2018).

麻生英樹(2013)："多層ニューラルネットワークによる深層表現の学習"、人工知能学会誌,2013.

内田祐介,山下隆義(2017)："畳み込みニューラルネットワークの研究動向", 電子情報通信学会技術研究報告2017, http://www.vision.cs.chubu.ac.jp/MPRG/F_group/F188_uchida2017.pdf

河原 達也, 原田 達也(2016)："音声認識・画像認識における機械学習の最近の動向",システム制御情報学会,学会誌60巻3号,2016.3

木田智士(2017)："Deep learningで画像認識④〜畳み込みニューラルネットワークの構成〜", https://lp-tech.net/articles/LVB9R, 2019.1.10アクセス

中谷秀洋(2012):"最適化のための勾配法",http://gihyo.jp/dev/serial/01/machine-learning/0016,2019.1.10アクセス

中村勇気(2018)："ディープラーニングを用いた学生の受講態度の推定",西南学院大学大学院経営学研究科修士論文、2018.1.

吉武春光(2016):“日経記事の解析に文脈ベクトルを使うための環境整備”, 西南学院大学商学論集, Vol. 62, No.3&4, pp. 263-284, 2016.3月

吉武春光(2018):“学生の提出レポート解析に文脈ベクトルを使う”, 西南学院大学商学論集, Vol. 64, No. 4, pp. 79-95, 2018.3月