ダウンロード・データあります

画像
データ
音

第1章　定番データセットの文字認識で初体験

画像向きCNN①…Hello World的手描き認識

牧野 浩二，西崎 博光

```
$ python eval_myimage_cnn.py -i tegaki/tegaki1.png -m result_CNN/MLP.model

自分の手書き文字を学習したモデルで評価してみるプログラム
#  入力画像ファイル： 2.png
#  学習済みモデルファイル： result/MLP.model

判定結果は2です。
```

（a）入力画像　　（b）実行結果

図1　手書き文字を分類する

ディープ・ラーニングの仕組みや，フレームワークChainerの動かし方を理解したら，いよいよ実践に移りましょう．

第3部では3大ディープ・ラーニング（CNN，RNN，AE）を全て試します．

畳み込みニューラル・ネットワーク（Convolutional Neural Network：CNN）は「画像処理に強い」ディープ・ラーニングです．

最近では，リカレント・ニューラル・ネットワーク（RNN）やオート・エンコーダ（AE）に組み入れられて使うケースが増えています．

MNIST[1]は，手書き文字を学習したり，正しく認識できるかどうかをチェックしたりする際に利用できるデータセットです．画像認識ですので，畳み込みニューラル・ネットワークの方が向いています．

ここでは，畳み込みニューラル・ネットワークで手書き文字を認識してみます．

● やること…画像向きCNNを使って精度よく手書き文字を認識する

手書き文字の画像データを入力し，文字を認識します．実行結果を**図1**に示します．10個の画像を分類した結果を**表1**に示します．6だけうまく分類できませんでした．

第2部第5章の自作ディープ・ニューラル・ネットワークでは，3と6の画像がうまく分類できませんでしたので，やはり畳み込みニューラル・ネットワークの方が良い結果が得られそうです．

実験

● 準備

実験には，**表2**のプログラムとデータを使います．ディープ・ニューラル・ネットワークで作ったプログラムを，畳み込みニューラル・ネットワーク用に改造したものです．これは本誌ウェブ・ページからダウンロードできます．

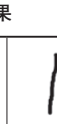
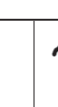
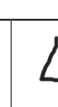
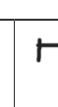
● ステップ1：学習する

サンプル・プログラムを以下のコマンドで実行します．

```
$ python train_mnist_cnn.py
```

学習時間は10分くらいでした．実行結果を**図2**に

表1　手書き文字の分類結果

入力画像										
ファイル名	tegaki0.png	tegaki1.png	tegaki2.png	tegaki3.png	tegaki4.png	tegaki5.png	tegaki6.png	tegaki7.png	tegaki8.png	tegaki9.png
分類結果	0	1	2	3	4	5	5	7	8	9