知能システム学特論レポート

（DL2 班）Caffe on Ubuntu

2015 年 7 月 13 日

1　報告者

15344203　有田 裕太
15344206　緒形 裕太
15344209　株丹 亮
12104125　宮本 和

2　進行状況

- 畳み込みネットワークと正規化層の理論について
- データセットの作成準備

3　理論研究

3.1　畳込み層

3.2　単一チャネルの正規化

単一チャネルの画像 x_{ij} に対し，プーリングと同様，画素 (i,j) を中心とする $H \times H$ の正方領域 P_{ij} を考える．減算正規化とは，入力画像の各画素濃淡から，P_{ij} に含まれる画素の濃淡の平均，つまり $\bar{x}_{ij} = \sum_{(p,q)\in P_{ij}} x_{i+p,j+q}$ を差し引く．

$$z_{ij} = x_{ij} - \bar{x}_{ij} \tag{1}$$

ここで差し引く $\bar{x}_{ij}$ には，重み付き平均

$$\bar{x}_{ij} = \sum_{(p,q)\in P_{ij}} w_{pq} x_{i+p,j+q} \tag{2}$$

を使う場合もある．その場合 w_{pq} は

$$\sum_{(p,q)\in P_{ij}} w_{pq} = \sum_{q=0}^{H-1} \sum_{q=0}^{H-1} w_{pq} = 1 \tag{3}$$

であり，領域の中央で最大値をとり，周辺部へ向けて低下するようなものとする．領域の中央部をより重視し，周辺部の相対的な影響を少なくするためである．

同じ領域内で，さらに画素値の分散を抑える操作が除算正規化である．P_{ij} 内の画素値の分散は

$$\omega_{ij}^2 = \sum_{(p,q)\in P_{ij}} w_{pq}(x_{i+p,j+q} - \bar{x}_{ij})^2 \tag{4}$$

となるが，減算正規化を施した入力画像をこの標準偏差で割る．

$$z_{ij} = \frac{x_{ij} - \bar{x}_{ij}}{\omega_{ij}} \tag{5}$$

この計算をそのまま行うと，濃淡変化が少ない局所領域ほど濃淡変化が増幅され，ノイズが強調される．そこで，入力画像のコントラストが大きい部分にのみ適用するために，ある定数 c を設定し，濃淡の標準偏差がこれを下回る ($\omega_{ij} < c$) で除算する

$$z_{ij} = \frac{x_{ij} - \bar{x}_{ij}}{max(\omega_{ij} < c)} \tag{6}$$

や，同様の効果が ω_{ij} に応じて連続的に変化する

$$z_{ij} = \frac{x_{ij} - \bar{x}_{ij}}{\sqrt{\omega_{ij} + c}} \tag{7}$$

を使う．減算正規化および式**??**による除算正規化の計算例を図**??**に示す．これらの正規化の結果，画像の画素値は負の値をとり得るため，画素値の最大値と最小値が [0,255] の範囲に収まるように画素値を線形変換している．

4　プログラミング

4.1　画像を学習させて分類器を作るまでの流れ

データセットを作成し，学習を行ってモデルを作成するまでの流れを Fig. 1 に示す．Fig. 1 より (1) の大量の写真データまたは動画からの切り出しに関しては前回の発表で述べた手法，作成したプログラムによって顔のみを切り出した画像ファイルを作成する．

4.2　切り出した画像からキャラクターごとにクラスタリングを行う

Fig. 1 より (2) の作業は (1) で生成された顔画像をそれぞれの人物，キャラクターごとにクラス分けする作業である．各画像はラベル名のディレクトリに入れ，ラベル名のディレクトリはこのように（0/ 1/ 2/...）数字にしておく．この数字とラベル名の関連付けは実際に画像を分類する際に必要なので記録しておく．

4.3　画像の正規化

Fig. 1 の (3) で，生成した画像を正規化する．正規化の処理は ImageMagick を用いた．クラスタリングした各ディレクトリで以下のコマンドを実行する．

```
1 for file in ‘ls‘; do convert ${file} -equalize ${file}; done
```

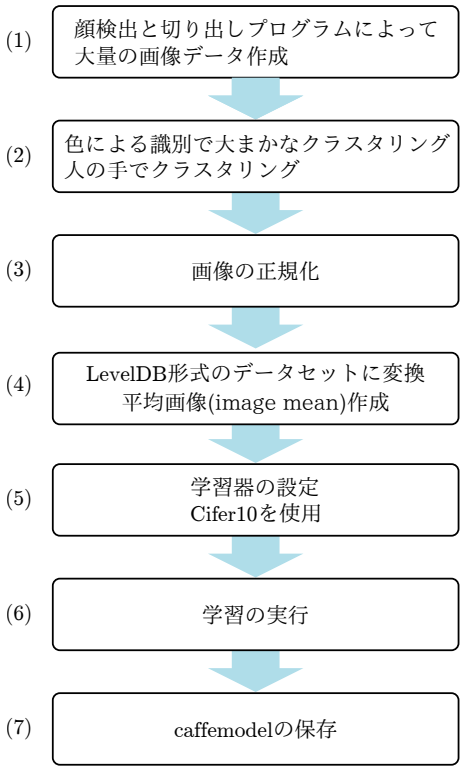

Fig.1　モデルを作成するまでの流れ

4.4　LevelDB データセットの作成

用意した画像を Caffe で読み込む LevelDB と呼ばれる形式に変換する必要がある．変換スクリプトは SIG2D[1] に掲載されているものを使用した．このスクリプトによって全入力画像から 9 割を訓練データ，1 割をテストデータに割り振ることができる．

4.5　平均画像の作成

平均画像を作成するために以下のコマンドを実行する．保存名は mean.binaryproto である．

```
build/tools/compute_image_mean.bin -backend=leveldb ./examples/cifar10/
    cifar10_train_leveldb ./examples/cifar10/mean.binaryproto
```

4.6　学習機の設定

今回は基本的には cifer10 の学習器をそのまま使用した．ただし，最終的な出力は対象とするものによって数が変わるため，変更を行った．変更するファイルは cifar10_quick_train_test.prototxt と cifar10_quick.prototxt である．

```
layers {
  name: "ip2"
```

```
  type: INNER_PRODUCT
  bottom: "ip1"
  top: "ip2"
  blobs_lr: 1
  blobs_lr: 2
  inner_product_param {
    num_output: 10       ここを変更
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
```

4.7 学習の実行

以下のコマンドによって学習を実行する．学習は前回発表したように，GPU を用いて行った．GPU を用いる場合，標準設定が GPU で行うように設定してあるので，ファイルの記述を変更する必要はない．CPU のみを用いる場合，cifar10_full_solver.prototxt を最後の行にある solver_mode: GPU を CPU に変更する．

```
caffe train --solver examples/cifar10/cifar10_quick_solver.prototxt
```

学習が完了すると，cifar10_quick_iter_4000.caffemodel というファイルが生成される．

5 今後の課題

- 理論研究を進める．
- データセットの作成，学習実行結果の評価と過程の可視化．

参考文献

[1] SIG2D，“SIG2D’ 14 Proceedings of the 3rd Interdimensional Conference on 2D Information Processing ”，http://sig2d.org/publications/，2014.