知能システム学特論レポート

（DL2 班）Caffe on Ubuntu

2015 年 7 月 16 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　勾配の計算

畳み込み層の計算式は，順伝播型ニューラルネットワークの中間層と同様に次のように表すことができる．

$$u^{(l)} = W^{(l)}u^{(l-1)} + b^{(l)} \tag{1}$$

$$z^{(l)} = f^{(l)}(u) \tag{2}$$

ただし W は，サイズ $H \times H \times K$ の M 個のフィルタの係数 h_{pqkm} を畳み込みを再現するように規則的に並べたものである．逆伝播の計算も全結合層の場合と基本的には同じだが，$W^{(l)}$ に同じフィルタの係数が何度も現れることを考慮する必要がある．これを表現するために，フィルタの係数 h_{pqkm} から行列 $W = W^{(l)}$ を作る過程を以下に示す．まず，フィルタの係数を適当な順で並べ，成分数 $H \times H \times K \times M$ のベクトル h を作る．次に h と同じ長さを持ち h との内積が，重み w_{ji} を与えるベクトルを t_{ji} と定義する．

$$W_{ji} = t_{ji}^T h \tag{3}$$

そして t_{ji} の成分を t_{jir} と書き，r を固定したとき t_{jir} を (j,i) 成分にもつ行列を T_r と書く．

この層 l のデルタを $\delta^{(l)}$ と書き，全結合層に対する勾配計算の式を形式的に適応すると，この層の重み $\mathbf{W} = \mathbf{W}^{(l)}$ の勾配は

$$\partial\mathbf{W} = \delta^{(l)}{\mathbf{z}^{(l-1)}}^\top \tag{4}$$

で与えられる．上述のように $\mathbf{W}$ の多くの成分はもともと 0 であり．そうでない成分も重み共有により同じ変数 (フィルタの係数) に対応する．そこでフィルタの係数 $\mathbf{h}$ についての勾配 $\partial\mathbf{W}$ に変形する必要がある．微分の線形性より，$\partial\mathbf{h}$ の成分 r は次のように表せる．

$$(\partial\mathbf{h})_r = \sum_{i,j}(\mathbf{T}_r \odot \partial\mathbf{W})_{ji} \tag{5}$$

$\odot$ は成分ごとの積を表し，和は行列の全成分の和を表す．

プーリング層には学習の対象となるパラメータはないので，勾配計算は必要ないが，さらに下層に伝えるデルタの逆伝播計算は必要である．平均プーリングでは層 $l+1$ のユニット j のサイズ $H \times H$ の受容野（プーリング領域）を P_j として

$$w_{ji}^{(l+1)} = \begin{cases} \frac{1}{H^2} & (\text{if } i \in P_j) \\ 0 & (\text{otherwise}) \end{cases} \tag{6}$$

とする．上の逆伝播計算では出力層ユニットのデルタに $\mathbf{W}^{(l+1)^\top}$ を掛けることで．それらが入力層に均等に振り割られる．最大プーリングでは，層 $l+1$ のユニット j のみ $w_{ji} = 1$ とし，それ以外を $w_{ji} = 0$ とする．つまりサンプルごとに（順伝播時のプーリングの結果により）重みが変化する．逆伝播計算では最大値を返したユニットにデルタがそのまま伝えられる．

4 プログラミング

4.1 学習パラメータの設定

学習を行う上で必要なパラメータについて説明する．この設定が記述されているファイルは cifar10_quick_solver.prototxt である．以下に設定ファイルの内容を示し，各パラメータに関する意味を記述する．

```
net: "examples/cifar10/cifar10_quick_train_test.prototxt"
test_iter: 100
test_interval: 500
base_lr: 0.0001
momentum: 0.9
weight_decay: 0.004
lr_policy: "fixed"
display: 100
max_iter: 4000
snapshot: 4000
snapshot_prefix: "examples/cifar10/cifar10_quick"
solver_mode: GPU
```

ここでバッチ数と呼ばれる単位を導入する．バッチ数は教師データを一度にいくつ処理するか（バッチサイズ）を決定し，これを 1[batch] とする．Caffe では繰り返し数をバッチ数で指定している．

net： 学習用ネットワーク定義ファイルを指定する．

test_iter： 学習中の正答率評価を 1 回行うのに使う評価セットのデータ数をバッチ数で指定．評価セットのデータ数とバッチサイズの除算を行い，この数値を設定することで正答率評価にすべての評価セットを用いることができる．

test_interval： テストデータから正答率評価を行う間隔をバッチ数で指定．データ数が多い場合，正答評価に多くの時間がかかるので用いるデータセットの規模によって適切な値に設定する必要がある．

base_lr, momentum, weight_decay, lr_policy： 学習率に関する設定．

display： 学習中のステータスを出力する回数をバッチ数で指定．

max_iter： 学習の計算を最大どれだけ続けるかを訓練データのバッチ数で指定．ここで指定された数値とバッチサイズの積算が学習が終了するまでの処理する画像枚数となる．

snapshot, snapshot_prefix： 学習の途中経過を保存する間隔と場所を指定．

solver_mode： 学習を CPU のみ，あるいは GPU を用いるかを指定．

4.2 訓練データの作成

独自の訓練データを用いて学習を行うということで今回はアニメ「ラブライブ！」における主要キャラクターの識別を Deep Learning で行うことを目標とした．そこで，まず訓練データを作成した．訓練データはキャラクター毎に約 2000 枚程度集めた．これはアニメの中から Haar-like 特徴量を用いて顔検出を行い，その顔検出によって得られた矩形を切り出し保存したものを人間の目によって分類して集めた．また，キャラクターの顔画像の他に負例として 7000 枚ほど当該キャラクター以外の顔画像やそもそも顔画像ではない画像も集めている．以上の訓練データの画像は全て 200×200 にリサイズを行っている．

その後，前回の説明であった通り，1 : 9 の割合でテストデータと訓練データにランダムに分け，caffe で用いられるデータ形式に変換後，学習を実行する．

4.3 学習

学習は以下のコマンドで実行される．

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

今回，アニメキャラクターの顔認識を行うにあたり，ネットワークモデルとして cifar10 のモデルを用いて学習を行った．cifar10 のモデルは畳込み層が 3 層，プーリング層が 3 層，全結合層が 2 層からなる CNN であり，活性化関数には Softmax 関数を用いている．

また，学習を行った際のパラメータの一部を以下に示す．

- test_iter : 100
- max_iter : 4000
- batch_size : 100
- solver_mode : CPU

4.4 学習結果

学習を行った結果得られた，トレーニングの繰り返し回数と損失関数の値及びテストデータにおける精度のグラフを Fig.1 に示す．精度は 500 イテレートあたりで急激に上昇し，最終的に 94.2% の精度となった．

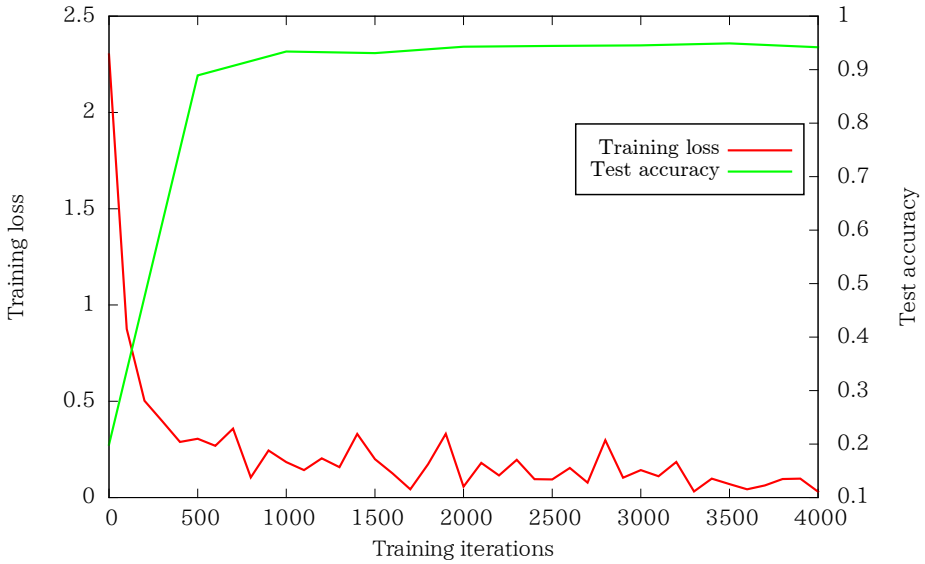

Fig.1　損失関数の値と精度

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．