

# 知能システム学特論レポート

(DL2 班) Caffe on Ubuntu

2015 年 7 月 6 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

- 理論研究
- 畳み込みネットワークについて

## 3 理論研究

### 3.1 畳込み層

実用的な畳込みネットでは、グレースケールの画像 1 枚に対してではなく、多チャンネルの画像に対し、複数個のフィルタを並行して畳込む演算を行う。チャンネルの画像とは各画素が複数の値を持つ画像であり、チャンネル数が  $K$  の画像の各画素は  $K$  個の値を持つ。例えば、グレースケールの画像では  $K = 1$ , RGB の 3 色からなるカラー画像では  $K = 3$  となる。畳込みネットの中間層では、さらにそれ以上のチャンネル数の画像を扱う。以下では、画像の縦横の画素数が  $W \times W$  でチャンネル数が  $K$  のとき、画像のサイズを  $W \times W \times K$  と書く。

濃淡地を各画素に格納したグレースケールの画像を考える。画像サイズを  $W \times W$  画素とし、画素をインデックス  $(i, j)$  ( $i = 0, \dots, W - 1, j = 0, \dots, W - 1$ ) で表す。これまでインデックスは 1 から開始していたが、画像（及びそれに類するもの）を扱う場合はインデックスを 0 から始めることとする。画素  $(i, j)$  の画素値を  $x_{ij}$  と書き、負の値を含む実数値をとるとする。そして、この画像に適用する  $H \times H$  画素のフィルタ（サイズの小さい画像）を考える。このフィルタのインデックス  $(p, q)$  ( $p = 0, \dots, H - 1, q = 0, \dots, H - 1$ ) で表し、画素値を  $h_{pq}$  と書く。

図 1 を用いて畳込み層での計算を説明する。この畳込み層は直前の層から  $K$  チャンネルの画像  $x_{ijk}$  ( $k = 0, \dots, K - 1$ ) を受け取り、これに  $M = 3$  種類のフィルタ  $h_{pqkm}$  ( $m = 0, \dots, M - 1$ ) を適用している。各フィルタ ( $m = 0, 1, 2$ ) は通常、入力と同じチャンネル数  $K$  を持ち（サイズを  $H \times H \times K$  とする）、図 1 のように

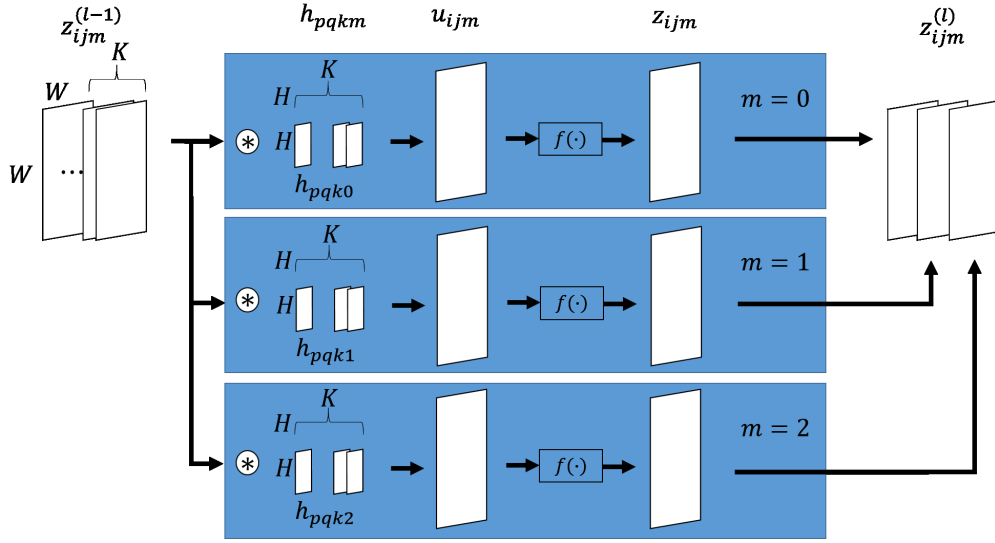


Fig.1 畳み込み層の概要

フィルタごとに計算は並行に実行される。計算の中身は、そのフィルタの各チャンネルごとに、これも並行に画像とフィルタの畳み込みを行った後、結果を画素ごとに全チャンネルにわたって加算する。

$$u_{ijm} = \sum_{p=0}^{K-1} \sum_{q=0}^{H-1} \sum_{k=0}^{H-1} z_{i+p,j+q,k}^{(l-1)} h_{pqkm} + b_{ijm} \quad (3.1)$$

入力画像のチャンネル数によらず、1つのフィルタからの出力は常に1チャンネルになる。

次にこうして得た  $u_{ijm}$  に活性化関数を適応する。

$$z_{ijm} = f(u_{ijm}) \quad (3.2)$$

図1のように、この値が畳み込み層の最終的な出力となり、その後の層へと伝わる。これらはフィルタ数  $M$  と同数のチャンネル数を持つ多チャンネルの画像とみなせる。つまり入力サイズが  $W \times W \times K$  のとき、出力サイズは  $W \times W \times M$  になる。

式 (6.2), (6.3) は、層間に特別な構造の単層ネットワークとして表現できる。この層の入力と出力のユニット数はそれぞれ  $W \times W \times K$  および  $W \times W \times M$  であり、畳み込み計算の局所性を反映して、出力層のユニットとのみ結合する。その結合の重みがフィルタ係数  $h_{pqkm}$  である。この重みは出力層の同一チャンネルの全ユニットで共有される。これは重み共有 (weigh sharing, weight tying) と呼ばれ、このような結合の局所性と重みを共有することが畳み込み層の特徴である。

### 3.2 プーリング

プーリング層は通常、畳み込み層の直後に設置される。プーリング層は畳み込み層で抽出された特徴の位置感度を低下させる働きがあり、対象とする特徴量の画像内での位置が若干変化した場合においても、プーリング層の出力が不変になるようにする。

プーリング層での計算は次のようにして行う。サイズ  $W \times W \times K$  の入力画像上で画素  $(i, j)$  を中心とする  $H \times H$  の正方領域とり、この中に含まれる画素の集合を  $P_{ij}$  で表す。 $P_{ij}$  内の画素についてチャンネル  $k$  ごと



	77	80	82	78	70	82	82	140	
	83	78	80	83	82	77	94	151	
	87	82	81	80	74	75	112	152	
	87	87	85	77	66	99	151	167	
	84	79	77	78	76	107	162	160	
	86	72	70	72	81	151	166	151	
	78	72	73	73	107	166	170	148	
	76	76	77	84	147	180	168	142	

 $\otimes$ 

0.01	0.08	0.01
0.08	0.62	0.08
0.01	0.08	0.01

 $=$ 

62	71	72	69	65	71	79	107
73	79	80	81	79	79	98	128
76	82	81	79	75	81	114	132
77	85	83	77	72	99	144	145
74	79	77	77	79	112	155	142
74	73	71	73	89	142	162	137
69	73	73	77	110	160	166	134
60	67	68	78	124	154	148	116

Fig.3 8 × 8 の入力画像にゼロパディングを施した場合

と計算される。

大きなサイズの入力画像を扱う際に、畳み込み層の出力側のユニット数が大きくなりすぎるのを防ぐために、2 以上のストライドが使われることがある。しかし、ストライドを大きくすることは画像特徴を取りこぼすことを意味し、性能を悪化させると考えられるため、できるだけ避けるべきである。また、プーリング層においても同様の考え方が用いられる。プーリング層ではその目的から 2 以上のストライドを用いられる。

## 4 プログラミング

### 4.1 学習を実行する PC 環境の見直し

今まで caffe のサンプル実行、中間層の出力は CPU のみの演算で十分な応答を得られた。しかし与えられたデータセットにおいて学習を行うプログラムを実行する場合、非常に時間がかかることがわかった。

現時点では学習を行うために十分なデータ数を有するデータセットを準備できていないため、大規模なデータセットで学習済みの状態から目的とする別のデータセットへ学習し直す方法（ファインチューニング）を行った。ファインチューニングは学習データがあまり多くない場合でも学習を行うことができる。この手法を用いて試験的に学習用データとテスト用データを 6 枚ずつ用意して学習を行ったが、ファインチューニングが完全に完了するまで約 3 時間かかった。

このことから今後の課題研究を可能な限り円滑に行うために、学習時には GPU を使った並列計算が必要であることがわかった。したがって caffe がサポートしている NVIDIA の「CUDA」及び、Deep Learning 用の CUDA ライブラリ「cuDNN」を使用する必要があると考える。後者のライブラリはデベロッパー登録申請（CUDA Registered Developer Program）が必要であるので申請を行い、認可を待っている状態である。

## 5 今後の課題

- 理論研究を進める。
- Caffe を使いこなす