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ネル数の画像を扱う. 以下では, 画像の縦横の画素数が W × W でチャネル数が K のとき, 画像のサイズを W × W × 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 $(i, j)(i = 0, \cdots, W-1, j = 0, \cdots, W-1)$ 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 $(p, q)(p = 0, \cdots, H-1, q = 0, \cdots, H-1)$ 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 チャネルの画像 $x_{ijk}(k = 0, ..., K-1)$ を受け取り, これに M = 3 種類のフィルタ $h_{pqkm}(m = 0, ..., M-1)$ を適用している. 各フィルタ (m = 0, 1, 2) は通常, 入力と同じチャネル数 K を持ち（サイズを H × H × K とする）, 図 1 のように

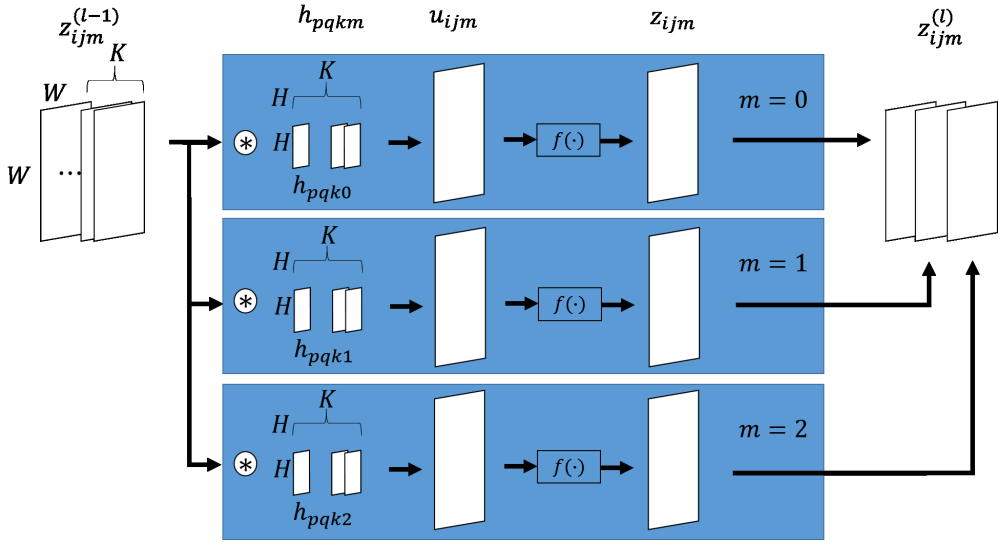

Fig.1　畳み込み層の概要

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ネルごとに, これも並行に画像とフィルタの畳込みを行った後, 結果を画素ごとに全チャネルにわたって加算する.

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

入力画像のチャネル数によらず,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.

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

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 と同数のチャネル数を持つ多チャ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　パディング

畳込みは画像にフィルタを重なり合う画素どおしの積を求めて，フィルタ全体の和を求める．したがって画像からフィルタがはみ出すような位置に重ねることができず，画像内にフィルタ全体が収まる範囲内でフィルタを動かすと，畳込み処理を行った後の画像サイズは入力画像は小さくなる．この様子を Fig. 2 に示す．

このときの画像サイズは

$$\left(W - 2\left\lfloor \frac{H}{2} \right\rfloor\right) \times \left(W - 2\left\lfloor \frac{H}{2} \right\rfloor\right) \tag{3.3}$$

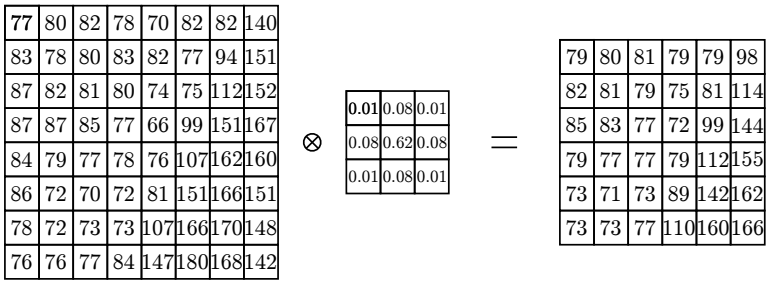

Fig.2　8 × 8 の入力画像を畳込み処理した場合の出力画像のサイズ縮小の様子

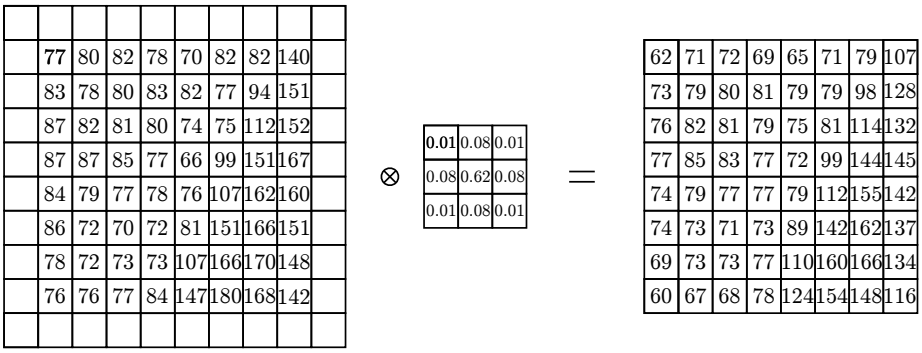

Fig.3　8 × 8 の入力画像にゼロパディングを施した場合

となる．$\lfloor \cdot \rfloor$ は小数点以下を切り下げて整数化する演算子である．

一方で畳込みの結果が入力画像と同サイズに出力する場合，入力画像の外側に幅 $\lfloor H/2 \rfloor$ の余剰を設け，出力画像が元の入力画像と同サイズになるようにする．このとき余分に設けた部分の画素値を 0 とする方法をゼロパディング（zero-padding）と呼ぶ．しかしゼロパディングを行った場合，画像の周辺部が暗くなってしまうため，画像をその 4 辺で折り返して未定部分の画素値を決める方法や画像の最終囲の画素値をそのまま適用する方法等を用いる場合がある．

3.3　ストライド

フィルタの適用位置を 1 画素ずつではなく，数画素ずつずらして計算する場合がある．このずらす間隔をストライド (stride) という．ストライドを s とするとき，出力画像の画素値は

$$u_{ij} = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} x_{si+p,sj+q} h_{pq} \tag{3.4}$$

となり，出力画像のサイズは元の画像サイズの約 $1/s$ 倍となる．サイズは正確には

$$([(W-1)/s]+1) \times ([(W-1)/s]+1) \tag{3.5}$$

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．

このことからもこれから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