

知能システム学特論レポート

(DL2 班) Caffe on Ubuntu

2015 年 7 月 9 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

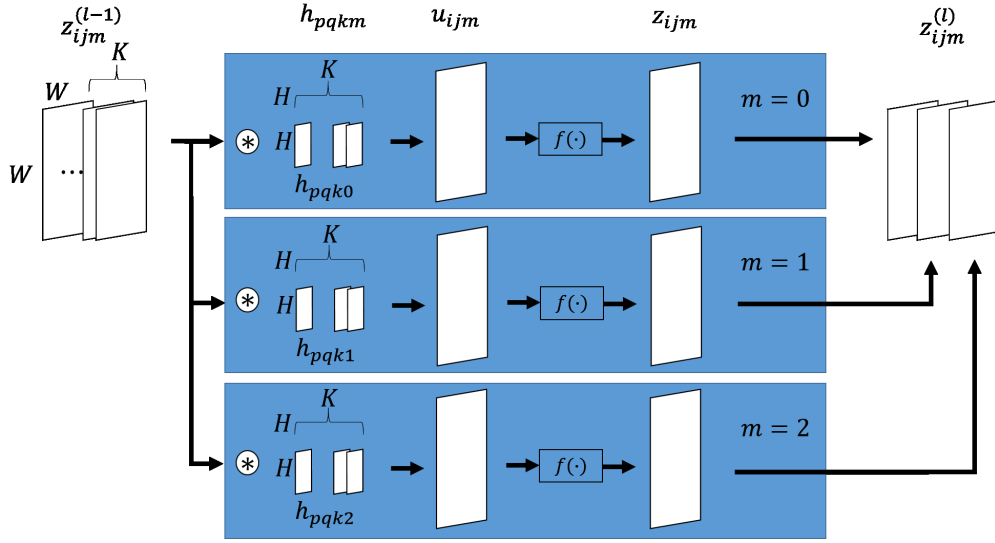


Fig.1 畳み込み層の概要

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 パディング

畳み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, 畳み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. ?? に示す.

このときの画像サイズは

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

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

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

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

$$((\lfloor (W-1)/s \rfloor + 1) \times (\lfloor (W-1)/s \rfloor + 1)) \quad (3.5)$$

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

学習を行う PC 環境を 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．後者のライブラリはデベロッパー登録申請 (CUDA Registered Developer Program) が認可され，Ubuntu 14.04 のマシンにインストールすることができた．正確な計測は行っていないが，明らかに学習速度の向上が見られた．

4.2 データセット作成のためのプログラムの作成

独自のデータセットを作成する練習として，写真や動画から人の顔を切り出し，データセットを作成，そして学習という一連の操作を行う計画を立てた．まずデータセットを作成するためには大量の人物が一人ずつ写った写真データを用意する必要がある．したがって OpenCV に実装されている顔検出アルゴリズムを用いて，人物が写った部分を自動的に切り出すプログラムを作成した．プログラミングに用いた言語は C++，Python である．

4.2.1 画像データからの切り出し

実行は以下のようにソースとなる画像が含まれたディレクトリと，出力先のディレクトリを指定する．



Fig.2 画像からの切り出し出力結果

```
$ python facedetect.py src output # 入力先と出力先ディレクトリ
```

Python で実装した機能を以下に示す。

- ディレクトリを指定したらそのディレクトリに入っているすべての画像ファイルを顔認識して切り取る。
- 指定したディレクトリの中に含まれているサブディレクトリの中もすべて探索してすべて取り込む。
- プログレスバーを設置して進捗を可視化

```
1 $ python facedetect.py src output
2 Input directoryname = src/
3 Output directoryname = output/
4 Exist "output/" folder.
5 File num = 1
6 66% (1 of 1) |#####| Elapsed Time: 0:00:07 ETA: 0:04:26
```

- 壊れた画像ファイルを読み込まれた場合でも例外処理で次の画像へ。

4.2.2 動画データからの切り出し

画像とほぼ同様の機能を有しており、ソースは動画ファイルを直接指定する。

```
$ python facedetect_video.py src/test.mp4 output # 動画ファイルと出力先ディレクトリ
```

4.2.3 データセットの作成

このプログラムを用いて、アニメのキャラクターやアイドルグループのメンバーなどの写真から顔の部分のみを切り出すことが可能となる。しかし切り出したあとは自分で顔を識別してフォルダ分けする作業が必要である。

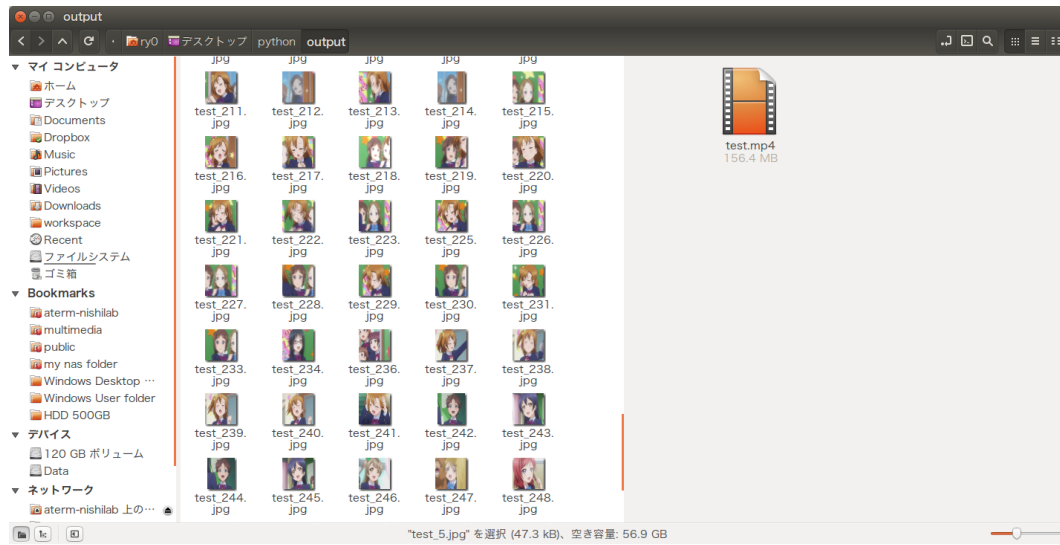


Fig.3 動画からの切り出し出力結果

5 今後の課題

- 理論研究を進める.
- データセットの作成, 学習の実行