

畳み込み実装4

shun sato

C言語で実装してみよう

なんでCで書くのか？

- > 速く動かしたいから

Cで書いたプログラムをPythonで動かそう

- ctypes: PythonでCの関数を呼び出すことができる

手順

1. Cで関数を書く
2. コンパイルして.soファイルを生成
3. Pythonからctypesを使って.soを読み込む
4. ctypesで関数を呼び出す

関数を呼び出す準備

```
import ctypes
from pathlib import Path
import numpy as np

LIBRARY_PATH = Path(__file__).parent / "conv2d.so"

def conv2d(src : np.ndarray, kernel : np.ndarray) -> np.ndarray:
    lib = ctypes.CDLL(LIBRARY_PATH)

    h, w, ch = src.shape
    kh, kw = kernel.shape
    pad_h = kh // 2
    pad_w = kw // 2

    pad_src = np.zeros((h + pad_h * 2, w + pad_w * 2, ch), dtype=np.uint8)
    pad_src[pad_h : h + pad_h, pad_w : w + pad_w, :] = src
    pad_src = np.ascontiguousarray(pad_src, dtype=np.uint8)
    kernel = np.ascontiguousarray(kernel, dtype=np.float32)
    dst = np.zeros((h, w, ch), dtype=np.uint8)

    lib.conv2d(
        pad_src.ctypes.data_as(ctypes.POINTER(ctypes.c_uint8)),
        dst.ctypes.data_as(ctypes.POINTER(ctypes.c_uint8)),
        kernel.ctypes.data_as(ctypes.POINTER(ctypes.c_float)),
        ctypes.c_int(h), ctypes.c_int(w), ctypes.c_int(ch),
        ctypes.c_int(kh), ctypes.c_int(kw)
    )

    return dst
```

np.ascontiguousarray


→ 配列をメモリ連続に変換

ctypesで配列を渡す

→ numpy配列の先頭のポインタを渡す

Cでの関数の設計

出力は戻り値ではなく引数のポインタで返す(一般的なCで配列を返す方法)



```
#include <stdint.h>

void conv2d(uint8_t* pad_src, uint8_t* dst, float* kernel,
            int h, int w, int ch, int kh, int kw) {

}
```

Cプログラムのビルド

GCCを使ってコンパイルを行う

以下のオプションを付けて.soファイルを生成する

```
gcc -shared -fPIC -o conv2d.so conv2d.c
```

実装のポイント

- 配列のインデックス計算をよく考えよう
 - パディング後の画像サイズは横幅が $(w + \text{pad_w} * 2)$
- 値を0~255にクリッピングすることを忘れない！