畳み込み実装3

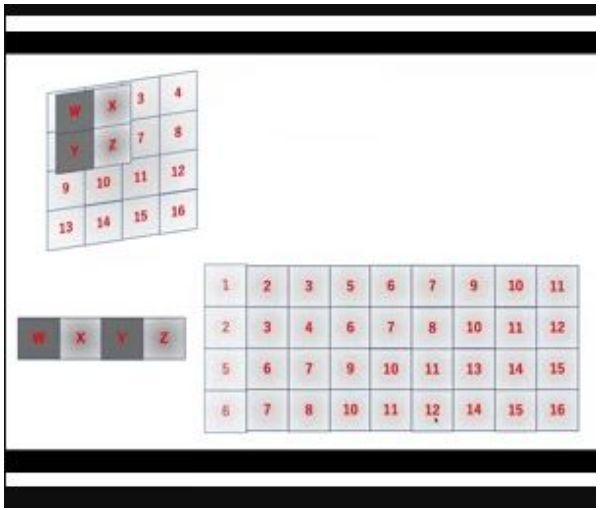
shun sato

畳み込みを速くしてみよう

畳み込み処理を行列の演算に変換できないか？

- 現状の画像の形状(h, w, c)とカーネルの形状(kh, kw)だと行列積は取れない
  - 画像を変換(c, h*w, kh*kw)&カーネルをベクトル化(kh * kw)したらできそう
  - 形状の計算：**(c, h*w, kh*kw)・(kh*kw) ➔ (c, h*w)**

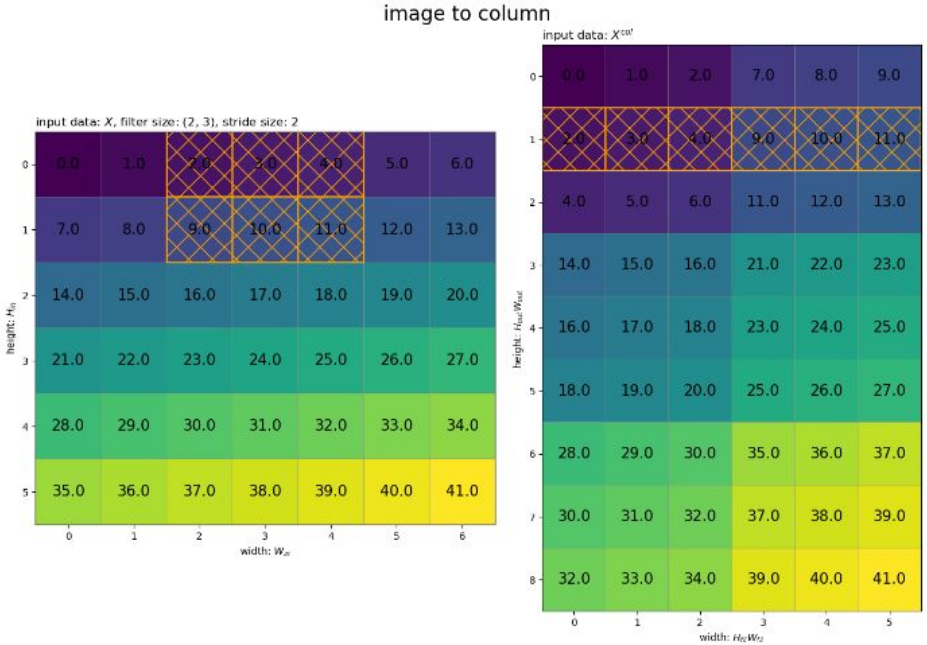
- 画像の形状を変換する：**im2col**



https://qiita.com/kuroitu/items/35d7b5a4bde470f69570

im2colのイメージ

- 画像のカーネル領域ごとにベクトル化
  - これを結合して大きな配列を作る
- ベクトル化したカーネルで行列積を計算する



im2colによる展開前後のデータ

https://www.anarchive-beta.com/entry/2024/06/14/235511

実装してみよう

```python
def conv2d(src : np.ndarray, kernel : np.ndarray):
    assert src.ndim == 3 or src.ndim == 2
    assert kernel.ndim == 2

    h, w, ch = src.shape
    kh, kw = kernel.shape

    trans_src = src.transpose(2, 0, 1)

    pad_h = kh // 2
    pad_w = kw // 2

    pad_src = np.zeros((ch, h + pad_h * 2, w + pad_w * 2), dtype=np.float32)
    pad_src[:, pad_h : h + pad_h, pad_w : w + pad_w] = trans_src

    flatten_kernel = kernel.reshape(kh * kw)
    col_src = im2col_naive(pad_src, kh, kw)
    col_dst = col_src @ flatten_kernel
    trans_dst = col_dst.reshape(ch, h, w)
    dst = trans_dst.transpose(1, 2, 0)

    dst = np.clip(dst, 0, 255)
    dst = dst.astype(np.uint8)

    return dst
```

1. 左の実装を参考に大枠を作る

2. im2col_naive関数を実装する
   a. (c, h, w)を(c, h*w, kh*kw)に変換
   b. h*wの部分はパディングを考慮して少し変える

numpyの操作説明

- transpose : 配列の次元を入れ替える
- reshape : 配列の形状を変化させる
- @ : テンソル積を計算する(np.dotでも可)
- clip : 配列の値を範囲内に収める
- astype : データ型を変更する(uint8 = [0, 255])

im2colをもっと速く

- 先程の実装は(c, h*w, kh*kw)
  - 変換の際にh*w回のループを行う必要があった

- (c, kh*kw, h*w)にしたらどうか？
  - 変換のループがkh*kw回になる
  - 形状の計算：**(kh*kw)・(c, kh*kw, h*w) ➔ (c, h*w)**
  - 先程と行列積の順番は入れ替える

- im2col_fastを実装してみよう！
  - 実行時間の計測と比較も行ってみよう