# 2018033_CV_HW18

```
# Given

x = np.array([
  [12, 14, 15, 17],
  [34, 56, 73, 32],
  [67, 43, 31, 21],
  [32, 31, 43, 56]
])
```

```
dy = np.array([
  [-2, 3, 4],
  [-1, 5, 3],
  [2, 3, 4]
])
```

```
w = np.array([
  [-1, 0],
  [0, -1]
])
```

```
b = 0.3
lr = 0.1
```

For easy computation wrote a python function convolve2d that takes the input matrix and the filter and outputs the convolved resulting matrix.

```
def convolve2d(inp, filter):
  a,b = inp.shape
  m,n = filter.shape

  output = []

  for i in range(a-m+1):
      output_ = []
      for j in range(b-n+1):
          output_.append(np.sum(inp[i:i+m,j:j+n]*filter))
      output.append(output_)

  return np.array(output)
```

## (1) Compute updated 'b' and 'w' in the current layer:

### Gradient Calculation:

```
[35] dw = convolve2d(x, dy)
     print("Gradient for w:")
     dw
```

```
Gradient for w:
array([[930, 753],
       [962, 825]])
```

```
[36] db = np.sum(dy)
     print("Gradient for b:")
     db
```

```
Gradient for b:
21
```

### Calculating Updated 'b' and 'w':

```
[37] updated_w = w - lr*dw
     print("Updated w:")
     updated_w
```

```
Updated w:
array([[-94. , -75.3],
       [-96.2, -83.5]])
```

```
[38] updated_b = b - lr*db
     print("Updated b:")
     updated_b
```

```
Updated b:
-1.8
```

## (2) Compute 'dy' for the next layer (in the backward direction) [0.5 marks]

**Calculating dx for current layer = dy for the previous layer = dy for next layer in backward direction:**

```
[39] dy_0 = np.pad(dy,(1,1))
     w_dash = np.rot90(np.rot90(w))

     dx = convolve2d(dy_0,w_dash)

     print("dy for the previous layer (next layer in backward direction):")
     dx
```

```
dy for the previous layer (next layer in backward direction):
array([[ 2, -3, -4,  0],
       [ 1, -3, -6, -4],
       [-2, -2, -9, -3],
       [ 0, -2, -3, -4]])
```

**(Code:**https://colab.research.google.com/drive/1pQRZFVwWlVCoOoBGmoL0E8JH4a3WxQxq?usp=sharing**)**