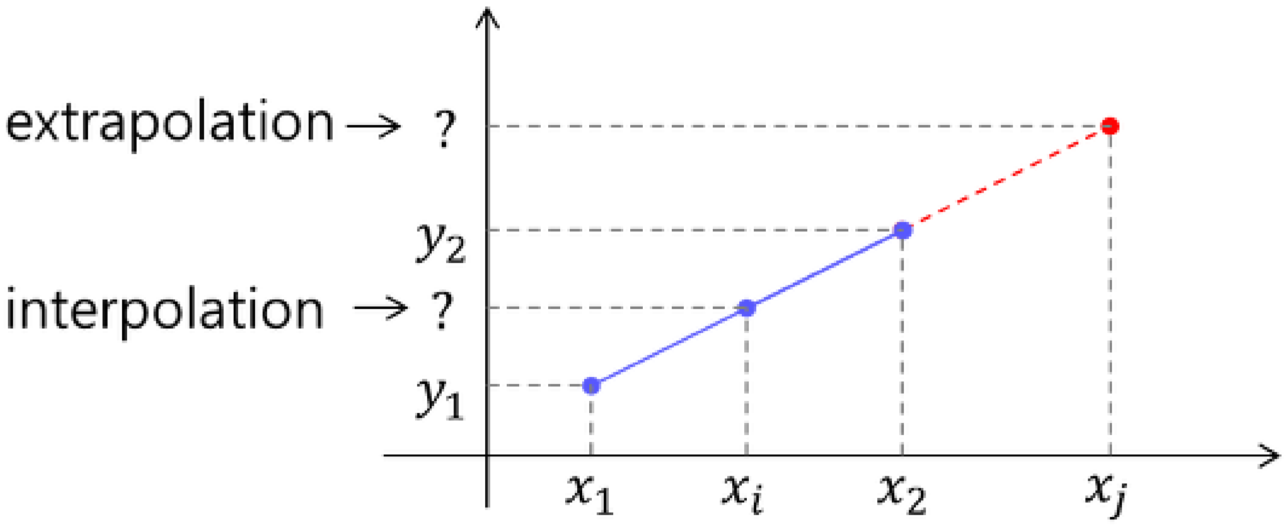
ROI Align Code Review

Presenter: Mira Kim

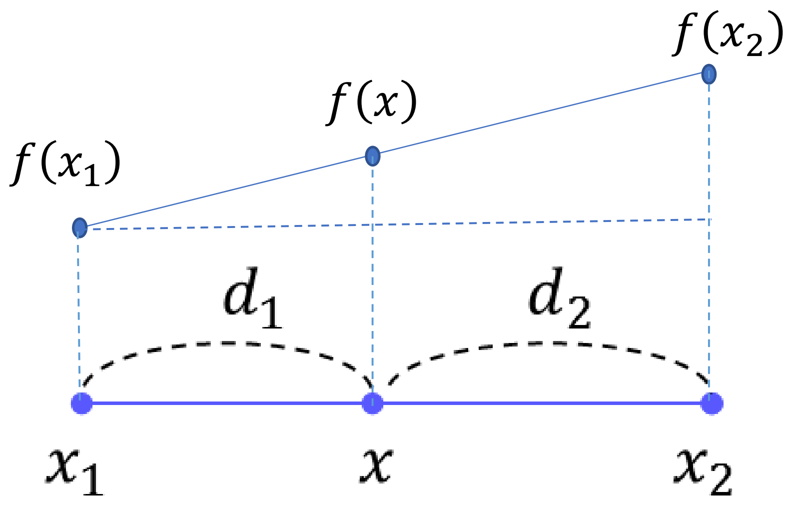
1. Interpolation & Extrapolation



Interpolation : $x_1 \leq x_i \leq x_2$

Extrapolation: $x_i \leq x_1 \leq x_2$ or $x_1 \leq x_2 \leq x_i$

2. 1D Linear Interpolation



$$d_1 + d_2 : f(x_2) - f(x_1) = d_1 : f(x) - f(x_1)$$

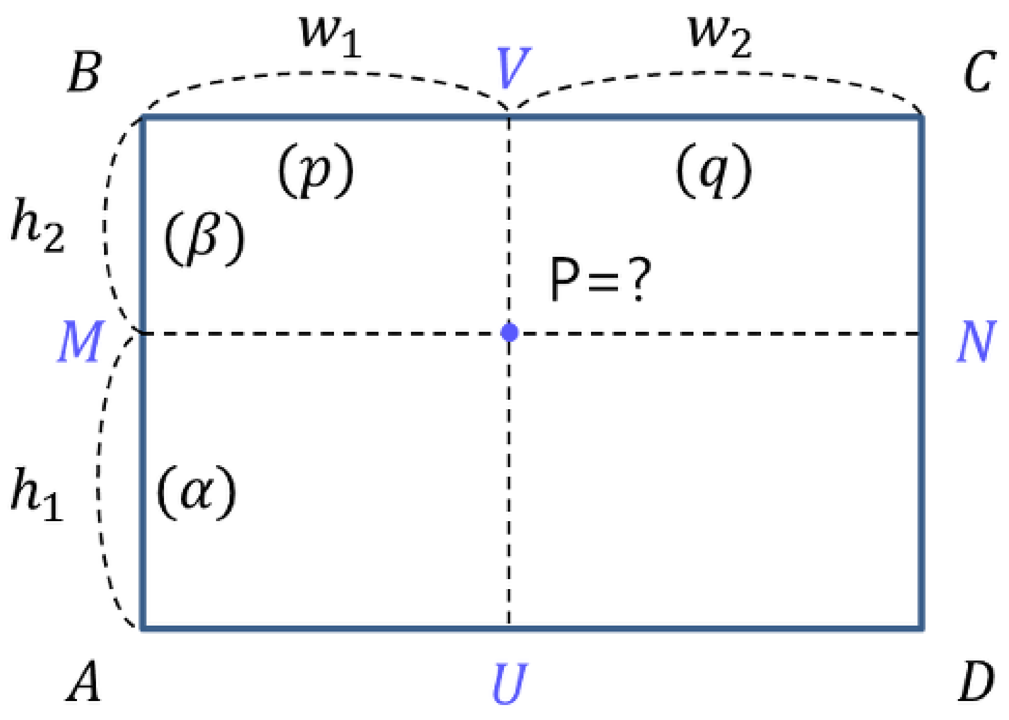
$$f(x) = f(x_1) + \frac{d_1}{d_1 + d_2} * (\mathrm{f}(\mathrm{x}_2) - \mathrm{f}(\mathrm{x}_1))$$

$$f(x) = \frac{d_2}{d_1 + d_2} f(x_1) + \frac{d_1}{d_1 + d_2} f(x_2)$$

$$f(x) = \beta f(x_1) + \alpha f(x_2)$$

$$(\alpha + \beta = 1)$$

3. Bilinear Interpolation



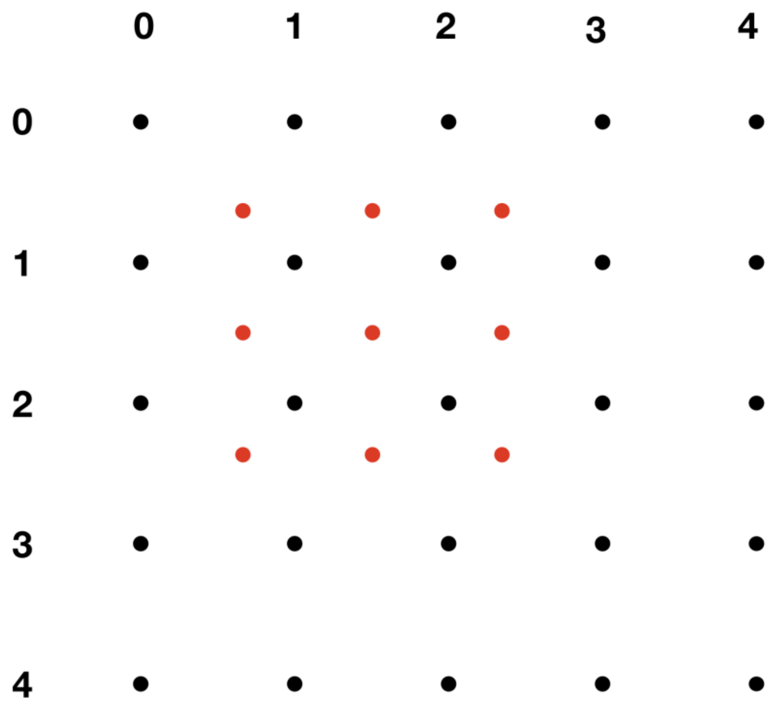
$$f(x)=\beta f(x_1)+\alpha f(x_2)$$

$$
\begin{aligned}
M &= \beta A + \alpha B \\
N &= \beta D + \alpha C \\
P &= qM + pC \\
&= q(\beta A + \alpha B) + p(\beta D + \alpha C) \\
&= q\beta A + q\alpha B + p\beta D + p\alpha C
\end{aligned}
$$

$$(\alpha + \beta = 1,\ p + q = 1)$$

4. ROI Align

0.27.	0.65.	0.24.	0.70.	0.67
0.57.	0.21.	0.51.	0.49.	0.70
0.82.	0.23.	0.66.	0.75.	0.98
0.90.	0.50.	0.38.	0.72.	0.79
0.76.	0.16.	0.27.	0.08.	0.57



5. ROI Align Code Implementation

```
[ ]  y_coordinates = np.linspace(y_min, y_max, height) * (img_height - 1)
     x_coordinates = np.linspace(x_min, x_max, width) * (img_width - 1)
```

where `img_height` and `img_width` are height and width of input feature map, and `height` and `width` are the desired height and width of output feature map.

numpy.linspace

numpy.linspace(*start*, *stop*, *num=50*, *endpoint=True*, *retstep=False*, *dtype=None*, *axis=0*) [source]

```
>>> np.linspace(2.0, 3.0, num=5)
array([2.  , 2.25, 2.5 , 2.75, 3.  ])
>>> np.linspace(2.0, 3.0, num=5, endpoint=False)
array([2. ,  2.2,  2.4,  2.6,  2.8])
>>> np.linspace(2.0, 3.0, num=5, retstep=True)
(array([2.  ,  2.25,  2.5 ,  2.75,  3.  ]), 0.25)
```

5. ROI Align Code Implementation

Given the coordinate(may be fractional) `[y, x]` of a red point, we can find the coordinate of the upper left, upper right, lower left, lower right neighbor: `[y_l, x_l]`, `[y_l, x_h]`, `[y_h, x_l]`, `[y_h, x_h]`

where

```
[ ]  y_l, y_h = np.floor(y).astype('int32'), np.ceil(y).astype('int32')
     x_l, x_h = np.floor(x).astype('int32'), np.ceil(x).astype('int32')
```

5. ROI Align Code Implementation

```
import numpy as np

def roi_align(image, box, height, width):
    """
    `image` is a 2-D array, representing the input feature map
    `box` is a list of four numbers
    `height` and `width` are the desired spatial size of output feature map
    """
    y_min, x_min, y_max, x_max = box

    img_height, img_width = image.shape

    feature_map = []

    for y in np.linspace(y_min, y_max, height) * (img_height - 1):
        for x in np.linspace(x_min, x_max, width) * (img_width - 1):

            y_l, y_h = np.floor(y).astype('int32'), np.ceil(y).astype('int32')
            x_l, x_h = np.floor(x).astype('int32'), np.ceil(x).astype('int32')

            a = image[y_l, x_l]
            b = image[y_l, x_h]
            c = image[y_h, x_l]
            d = image[y_h, x_h]

            y_weight = y - y_l
            x_weight = x - x_l

            val = a * (1 - x_weight) * (1 - y_weight) + b * x_weight * (1 - y_weight) + c *  y_weight * (1 - x_weight) + d * x_weight * y_weight

            feature_map.append(val)

    return np.array(feature_map).reshape(height, width)
```

5. ROI Align Code Implementation

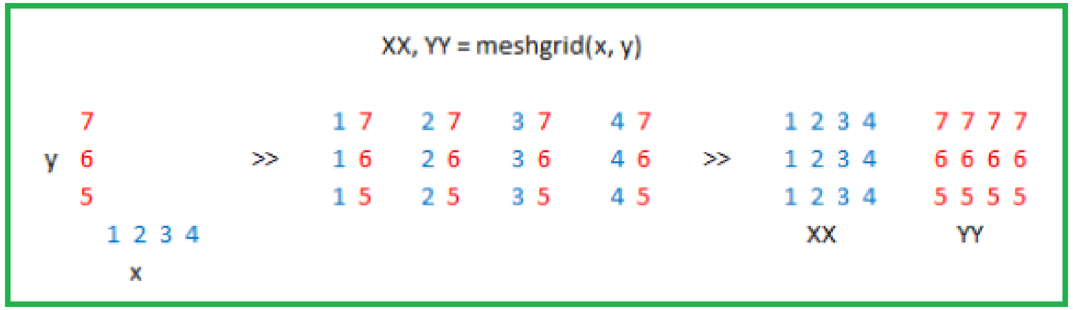
```
import numpy as np

def roi_align_vectorized(image, box, height, width):
    """
    `image` is a 2-D array, representing the input feature map
    `box` is a list of four numbers
    `height` and `width` are the desired spatial size of output feature map
    """
    y_min, x_min, y_max, x_max = box        # 0.32 0.05 0.43 0.54

    img_height, img_width = image.shape     # 5 6

    y, x = np.meshgrid(
        np.linspace(y_min, y_max, height) * (img_height - 1),
        np.linspace(x_min, x_max, width) * (img_width - 1)
    )

    print(y)            # [[1.28 1.5 1.72][1.28 1.5 1.72]]
    print(x)            # [[0.25 0.25 0.25][2.7 2.7 2.7]]
```

numpy.meshgrid

numpy.meshgrid(**xi, **kwargs*)

Return coordinate matrices from coordinate vectors.

```
>>> nx, ny = (3, 2)
>>> x = np.linspace(0, 1, nx)
>>> y = np.linspace(0, 1, ny)
>>> xv, yv = np.meshgrid(x, y)
>>> xv
array([[0. , 0.5, 1. ],
       [0. , 0.5, 1. ]])
>>> yv
array([[0.,  0.,  0.],
       [1.,  1.,  1.]])
```



5. ROI Align Code Implementation

```
y = y.transpose().ravel()                    # [1.28 1.28 1.5  1.5  1.72 1.72]
x = x.transpose().ravel()                    # [0.25 2.7  0.25 2.7  0.25 2.7 ]

image =image.ravel()     # [251.  44.  47. 104. 178. 101.  93.  46.  73. 218. 192.  22.  98.  85. 122. 144.

y_l, y_h = np.floor(y).astype('int32'), np.ceil(y).astype('int32')
x_l, x_h = np.floor(x).astype('int32'), np.ceil(x).astype('int32')

a = image[y_l * img_width + x_l]
b = image[y_l * img_width + x_h]
c = image[y_h * img_width + x_l]
d = image[y_h * img_width + x_h]

y_weight = y - y_l
x_weight = x - x_l

feature_map = a * (1 - x_weight) * (1 - y_weight) + b * (x_weight) * (1 - y_weight) + ₩
               c * y_weight * (1 - x_weight) + d * x_weight * y_weight

return feature_map.reshape(height, width)
```

6. ROI Align Result

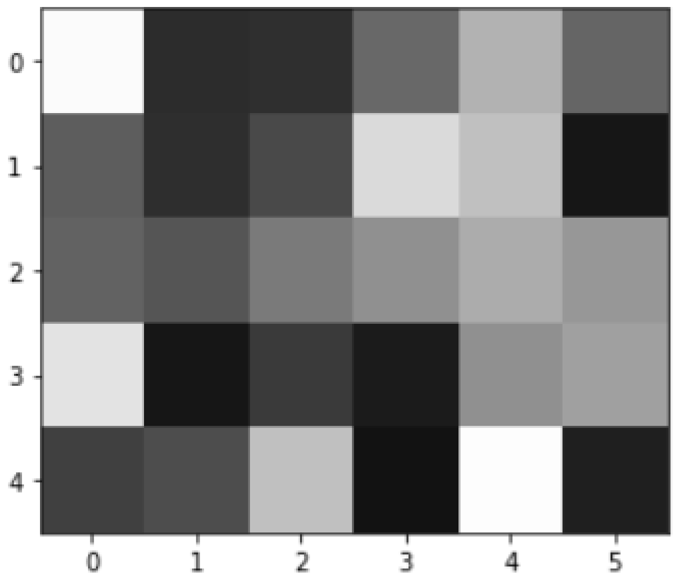
```
image = np.array([[251.,  44.,  47., 104., 178., 101.],
       [93.,  46.,  73., 218., 192.,  22.],
       [98.,  85., 122., 144., 172., 151.],
       [227.,  22.,  58.,  27., 144., 160.],
       [ 64.,  77., 192.,  18., 253.,  31.]])

bounding_box = np.array([0.32, 0.05, 0.43, 0.54])
```

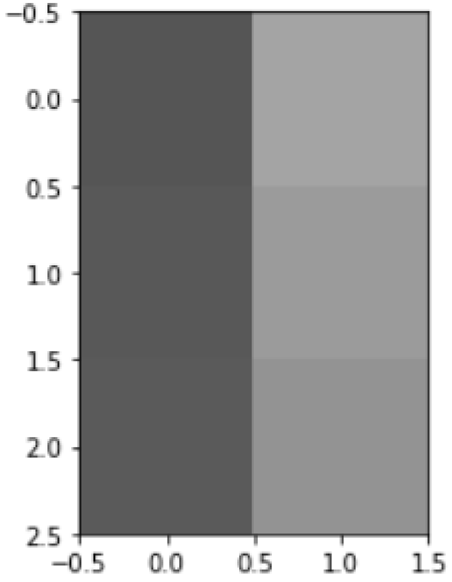
```
array([[ 85.03 , 164.112],
       [ 88.   , 155.95 ],
       [ 90.97 , 147.788]])
```

```
roi_aligned = tf.image.crop_and_resize(image_new, box, box_indices, crop_size, ₩
                                       method='bilinear', extrapolation_value=0.0)

print(roi_aligned)
```

```
tf.Tensor(
[[[[ 85.03 ]
   [164.112]]

  [[ 88.   ]
   [155.95 ]]

  [[ 90.97 ]
   [147.788]]]], shape=(1, 3, 2, 1), dtype=float32)
```

6. ROI Align Result





Thank You ☺