CS5370: Deep Learning for Vision – Assignment 1

Submitted By:
Vishal Singh Yadav
CS20MTECH01001

<1><a>

$$I = \begin{bmatrix} 2 & 0 & 1 \\ 1 & -1 & 2 \end{bmatrix}$$

$$F = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

Using bottom-right padding,

$$I = \begin{bmatrix} 2 & 0 & 1 & 0 \\ 1 & -1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad F^m = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Let final image be G:

$$G[0][0] = 2\times-1 + 0\times1 + 1\times-1 + -1\times1 = -4$$
$$G[0][1] = 0\times-1 + 1\times1 + -1\times-1 + 2\times1 = 4$$
$$G[0][2] = 1\times-1 + 0\times1 + 2\times-1 + 0\times1 = -3$$

$$G[1][0] = 1\times-1 + -1\times1 + 0\times-1 + 0\times1 = -2$$
$$G[1][1] = -1\times-1 + 2\times1 + 0\times-1 + 0\times1 = 3$$
$$G[1][2] = 2\times-1 + 0\times1 + 0\times-1 + 0\times1 = -2$$

$$\therefore G = \begin{bmatrix} -4 & 4 & 3 \\ -2 & 3 & -2 \end{bmatrix}$$

<b> $I = \begin{bmatrix} 2 & 0 & 1 \\ 1 & -1 & 2 \end{bmatrix}$

$F = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$

F can be written as.

$F_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $F_2 = [1 \; -1]$

for $F_1 * I$ $F_1^m = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Padded $I = \begin{bmatrix} 2 & 0 & 1 \\ 1 & -1 & 2 \\ 0 & 0 & 0 \end{bmatrix}$

$G[0][0] = 2 \times 1 + 1 \times 1 = 3$
$G[0][1] = 0 \times 1 + -1 \times 1 = -1$
$G[0][2] = 1 \times 1 + 2 \times 1 = 3$
$G[1][0] = 1 \times 1 + 0 \times 1 = 1$
$G[1][1] = -1 \times 1 + 0 \times 1 = -1$
$G[1][2] = 2 \times 1 + 0 \times 1 = 2$

$G = \begin{bmatrix} 3 & -1 & 3 \\ 1 & -1 & 2 \end{bmatrix}$

for $F_2 * (6 \times 1)$ $\quad F_2^T = [-1 \quad 1]$

Padded 1 = $\begin{bmatrix} 3 & -1 & 3 & 0 \\ 1 & -1 & 2 & 0 \end{bmatrix}$

$G[0][0] = 3 \times -1 + -1 \times 1 = -4$
$G[0][1] \quad -1 \times -1 + 3 \times 1 = 4$
$G[0][2] \quad 3 \times -1 + 0 \times 1 = -3$

$G[1][0] \quad 1 \times -1 + -1 \times 1 = -2$
$G[1][1] \quad -1 \times -1 + 2 \times 1 = 3$
$G[1][2] \quad 2 \times -1 + 0 \times 1 = -2$

$G = \begin{bmatrix} -4 & 4 & -3 \\ -2 & 3 & -2 \end{bmatrix}$

<(> for $f = f_1, f_2$

prove: $f * I = f_2 * (f_1 * I)$

Rewriting eqn:

$$(f * I)[i,j] = f_2 * (f_1 * I).$$

~~Equation~~

$$f_1 * I \;[i,j] = \sum_l I[i-l][j]\, f_2[i].$$

Let $G = f_2 * I[i,j]$

$$f_2 * G[i,j] = \sum_k G[i,j-k]\, f_2[j]$$

$$= \sum_k \left(\sum_l I[i-l][j-k] f_1[i] \right) f_2[j]$$

$$= \sum_k \sum_l I[[i-l][j-k]\; f_1[i] f_2[j].$$

Hence, if $f = f_1 * f_2$

Now $f = f_1 f_2$

$$f[i,j] = f_1[i] f_2[j]$$

Substituting in above eqn.

$$\sum_k \sum_l I[i-l][j-k] f[i,j]$$

$= I * f$. $= f * I$ (convolution is commutative

(d) For direct convolution with zero padding, we have

$$G[i][j] = I_1 \times F_1 + I_2 \times F_2 + I_3 \times F_3 + I_4 \times F_4$$

we have 4 multiplications for each $G[i][j]$.

we need to calculate $3 \times 2 = 6$ $G[i][j]$.

Total calculations $= 6 \times 4 = 24$.

For separable convolution with zero padding, we have.

$$G[i][j] = I_1 \times F_1 + I_2 \times F_2$$

i.e. 2 multiplications for each $G[i][j]$.

we need to calculate $2 \times 3 = 6$ times for column wise convolution.

we then perform $2 \times 3 = 6$ times multiplication for row wise convolution.

total multiplications $= (6+6) \times 2$

$= 12 \times 2 = 24$

Hence, both methods require equal number of multiplications.

(e) assuming no padding is applied on image,

$I = M_1 \times N_1$ image

$f = M_2 \times N_2$ filter

(i) for each iteration, every element of filter is multiplied in image.

∴ no of multiplications $= M_2 \times N_2$

we now need to find how many times filter is applied.

horizontally, filter moves from 0 to $M_1 - M_2$

vertically, filter moves from 0 to $N_1 - N_2$

∴ no of times filter is applied on image

$= (M_1 - M_2)(N_1 - N_2)$

Total multiplications:

$M_2 N_2 (M_1 - M_2)(N_1 - N_2)$

(ii) for seperable filter, we get two filters,

$f_1 = 1 \times N_2$ size ; $f_2 = M_2 \times 1$ filter.

for column wise convolution,

horizontally filter moves from 0 to M_1

vertically filter moves from 0 to $N_1 - N_2$

no of times filter applied $= M_1 \times (N_1 - N_2)$

for row wise convolution,
no of rows = $N_1 - N_2$ (from column conv).

horizontally, filter moves from 0 to $M_1 - M_2$
vertically filter moves from 0 to $N_1 - N_2$.
no of times filter applied = $(M_1 - M_2)(N_1 - N_2)$

for column wise convolution,
no of multiplications in filter = N_2
no of multiplications in filter for row wise convolution = M_2.

total calculations =

$$N_2 \times M_1(N_1 - N_2) + M_2 \times (M_1 - M_2)(N_1 - N_2)$$

$$= M_1 N_2 (N_1 - N_2) + M_2 (M_1 - M_2)(N_1 - N_2)$$

(iii) for direct 2D convolution, multiplications =
$M_2 N_2 (M_1 - M_2)(N_1 - N_2)$

for seperable convolution, multiplications :-
$M_1 N_2 (N_1 - N_2) + M_2 (M_1 - M_2)(N_1 - N_2)$

if we conside filter size to be very small in comparision to image size,

We can ignore M_2, N_2 in $(M_1 - M_2)$ and $(N_1 - N_2)$

for direct 2D convolution,

$M_2 N_2 \cdot M_1 N_1$

for seperable convolution,

$$M_1 N_1 \cdot N_2 + M_1 N_1 \cdot M_2$$
$$= M_1 N_1 (M_2 + N_2)$$

since, for any two positive integers, A & B,

$A + B \leq A * B \quad ; A \& B \geq 2$

we can conclude that

$$M_2 + N_2 \leq M_2 N_2$$

Hence, seperable convolutions is more efficient than direct 2D convolution.

<2><a> original pt: (x, y)
new pt: $(x\cos\theta, x\sin\theta)$.

Canny edge detector depends on magnitude of derivative.

at pt (x, y),
horizontal component = D_{xx}.
vertical component = D_{yy}.

at pt $(x'y')$
horizontal component = $D_{x'x'}$
vertical component = $D_{y'y'}$

$D_{x'x'} = x\cos\theta$
$D'_{x'x'} = \cos\theta$

$D_{y'y'} = x\sin\theta$
$D'_{y'y'} = \sin\theta$

In first quadrant, $\sin\theta$ goes from 0 to 1.
$\cos\theta$ goes from 1 to 0.

since atleast one component of will always be non-zero;
there will a magnitude associated with the point at all θ.

Hence, canny edge detector will always detect edge

(b) ~~Long edges with gaps occur~~

Spurious edges appear when the low threshold is too low.
We can increase the low threshold. to remove spurious edges.

Long edges with broken segments occur when some portions of edges go below low threshold.

To solve both problems together, we can reduce the low threshold a little ~~so that~~ and ~~reduce the high threshold as well so more points~~. keep the high threshold same.

This will

To solve both problems together, we can reduce low threshold and increase high threshold. This will make so that spurious edges don't appear. but continous edges will not go below low threshold., therefore giving smooth edges.