CNNs in Matrix Vector Notation

CAP 5415, FALL 2019.

Correlation Example

$x(n) = 1, 3, 2, 5, 8$

$h(n) = 1, 3, 0, 5, 8$

$x(n) = 0, 0, 0, 0, 1, 3, 2, 5, 8, 0, 0, 0, 0$ (zero padded)

n	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8

$h(n) = 1, 3, 0, 5, 8$

$L = 5$

$$y = Xh$$

$$\begin{bmatrix} 8&0&0&0&0 \\ 5&8&0&0&0 \\ 2&5&8&0&0 \\ 3&2&5&8&0 \\ 1&3&2&5&8 \\ 0&1&3&2&5 \\ 0&0&1&3&2 \\ 0&0&0&1&3 \\ 0&0&0&0&1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 0 \\ 5 \\ 8 \end{bmatrix} = \begin{bmatrix} 8 \\ 29 \\ 17 \\ 49 \\ 99 \\ 53 \\ 31 \\ 29 \\ 8 \end{bmatrix}$$

X $\quad$ h $\quad$ y

Largest Value Occurs when There is maximum Match between The signals

$$y(k) = \sum_{n=0}^{L-1} x(k+n)h(n)$$

$y(k) = 8,29,17,49,99,53,31,29,8$

- Correlation measures how much two signals match
- Same process as convolution, but without reversing the signal

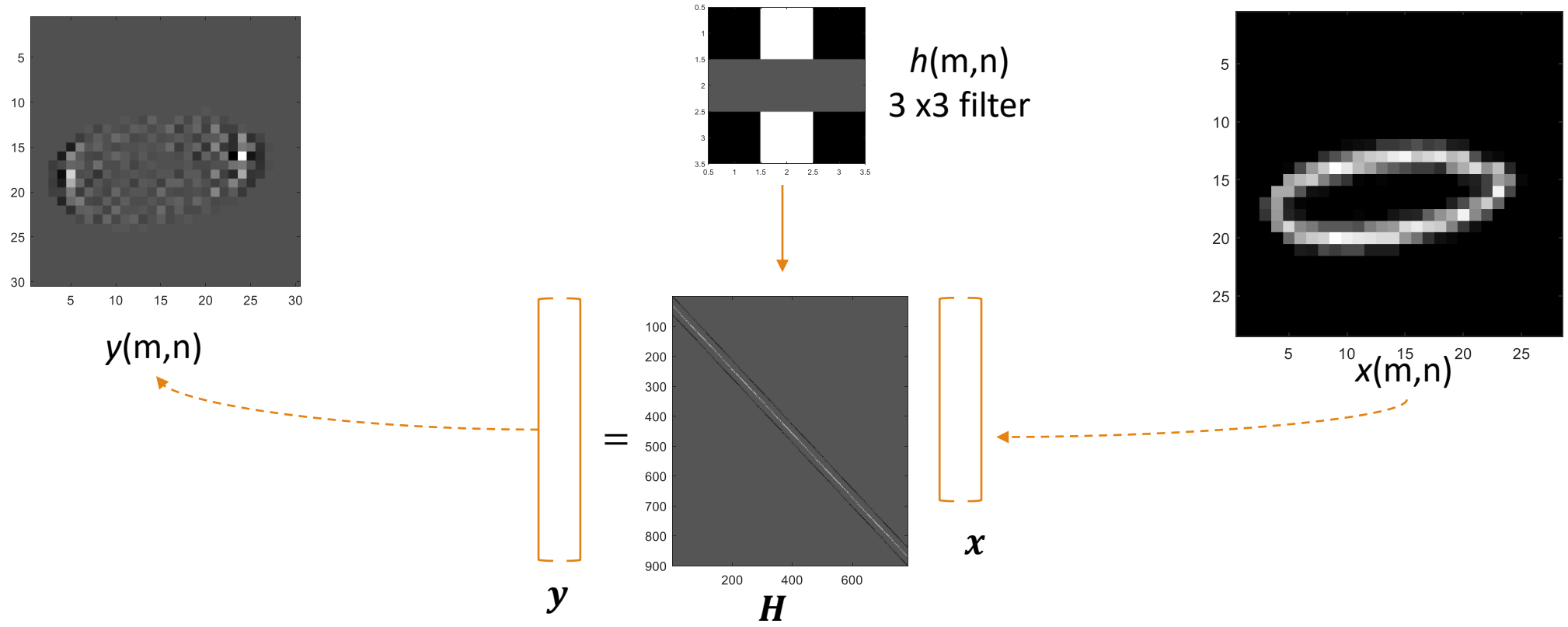
2D Matrix Vector notation for Correlation/Convolution

$$\boldsymbol{y = Xh}$$

$$\boldsymbol{y = Hx}$$

- The convolution result is the same when the input image and the filter kernel are interchanged
  - $y(m,n) = \sum x(m+k, n+l)h(k,l),$ $\quad y(m,n) = \sum h(m+k, n+l)x(k,l)$

Example of 2D convolution matrix



- The 3 x 3 filter kernel *h*(m,n) is used to form a *Toeplitz* matrix ***H.***
  - Multiplying the image vector ***x*** by ***H*** yields the convolution result as a vector ***y***
  - $y(m,n) = \sum h(m+k, n+l)x(k,l)$

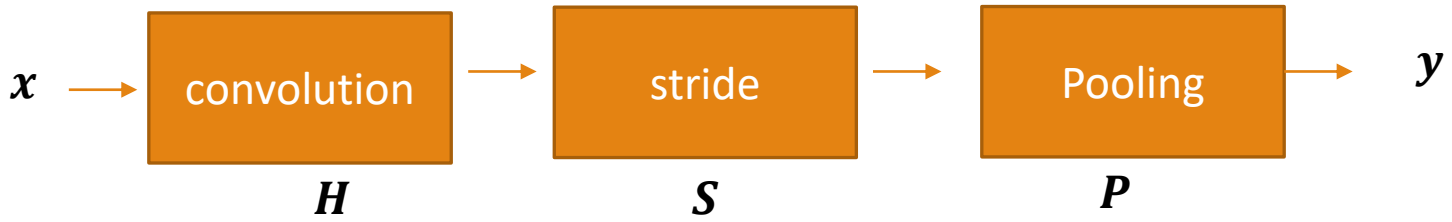
Stride and Pooling

$$\underset{\boldsymbol{y}}{\begin{bmatrix} a & b & e & f \\ i & j & m & n \end{bmatrix}} = \underset{\boldsymbol{S}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}} \underset{\boldsymbol{H}}{\begin{bmatrix} a & b & e & f \\ c & d & g & h \\ i & j & m & n \\ k & l & o & p \end{bmatrix}}$$

$$\underset{\boldsymbol{y}}{\begin{bmatrix} a+c & b+d & e+g & f+h \\ c+i & d+j & g+m & h+n \\ i+k & j+l & m+o & n+p \\ k & l & o & p \end{bmatrix}} = \underset{\boldsymbol{P}}{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \underset{\boldsymbol{H}}{\begin{bmatrix} a & b & e & f \\ c & d & g & h \\ i & j & m & n \\ k & l & o & p \end{bmatrix}}$$

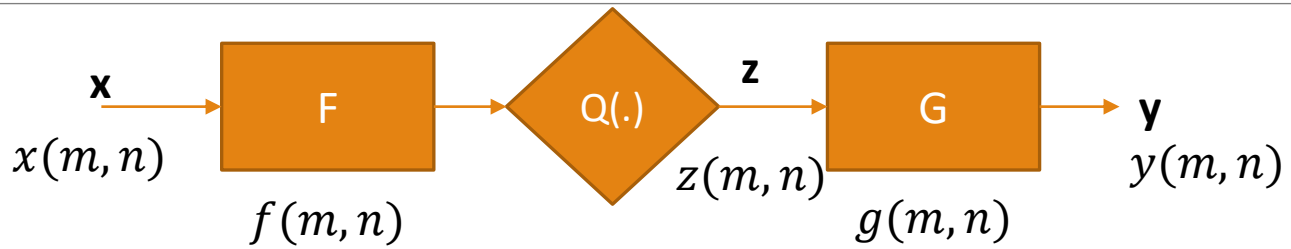
- The Stride matrix ***S*** can be implemented by a matrix operation with ‘1s’ for the rows that need to be selected in the ***H*** matrix
- Average pooling can be implemented by using a using a matrix ***P*** that has ‘1s’ for all the elements that need to be added
  - This is equivalent to an averaging filter.

A typical Convolution layer with Stride and Pooling



- These matrices can be multiplied in the order the operations are desired to get the output
  - $y = PSHx$
- The product of these matrices can be combined into one matrix $F = PSH$ so that
  - $y = Fx$

Back Propagation for CNNs in Matrix Vector form



- Consider a two layer where the input is **x** and the output is **y**.
  - The operator $Q(\cdot)$ is a point-by-point non-linearity
  - $\boldsymbol{z} = Q(\boldsymbol{F}\boldsymbol{x})$ and $\boldsymbol{y} = \boldsymbol{G}\boldsymbol{z}$
- Consider a cost function $J(\boldsymbol{y})$
  - We wish to optimize $J(\boldsymbol{y})$ w.r.t to the weights $f(m,n)$ and $g(m,n)$
  - Assume that $J(\boldsymbol{y})$ has an analytical derivative $\frac{dJ(\boldsymbol{y})}{d\boldsymbol{y}}$ which is known
- To learn $g(m,n)$ w.r.t to this cost function we need $\frac{dJ(\boldsymbol{y})}{d\boldsymbol{g}}$.
  - $\frac{dJ(\boldsymbol{y})}{d\boldsymbol{g}} = \frac{d\boldsymbol{y}}{d\boldsymbol{g}}\frac{dJ(\boldsymbol{y})}{d\boldsymbol{y}}$
  - We observe that $\boldsymbol{y} = \boldsymbol{G}\boldsymbol{z} = \boldsymbol{Z}\boldsymbol{g}$. Therefore $\frac{d\boldsymbol{y}}{d\boldsymbol{g}} = \boldsymbol{Z}$, and $\frac{dJ(\boldsymbol{y})}{d\boldsymbol{g}} = \boldsymbol{Z}\frac{dJ(\boldsymbol{y})}{d\boldsymbol{y}}$
- To learn $f(m,n)$ w.r.t to this cost function we need $\frac{dJ(\boldsymbol{y})}{d\boldsymbol{f}}$
  - $\frac{dJ(\boldsymbol{y})}{d\boldsymbol{f}} = \frac{d\boldsymbol{z}}{d\boldsymbol{f}}\frac{d\boldsymbol{y}}{d\boldsymbol{z}}\frac{dJ(\boldsymbol{y})}{d\boldsymbol{y}}$
- Note that $\frac{d\boldsymbol{y}}{d\boldsymbol{z}} = \boldsymbol{G}$
- What is $\frac{d\boldsymbol{z}}{d\boldsymbol{f}}$?

Derivative of non-linearity in matrix vector notation

- What is $\frac{d\boldsymbol{z}}{d\boldsymbol{f}}$?
- Consider a 2 x 2 case

$$\begin{bmatrix} z_1 \\ z_1 \end{bmatrix} = Q\left(\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}\right) = \begin{bmatrix} Q(x_{11}f_1 + x_{12}f_2) \\ Q(x_{21}f_1 + x_{22}f_2) \end{bmatrix}$$

- Therefore,
  - $$\begin{bmatrix} \frac{dz_1}{df_1} & \frac{dz_1}{df_2} \\ \frac{dz_2}{df_1} & \frac{dz_2}{df_2} \end{bmatrix} = \begin{bmatrix} \frac{dQ(x_{11}f_1+x_{12}f_2)}{df_1} & \frac{dQ(x_{11}f_1+x_{12}f_2)}{df_2} \\ \frac{dQ(x_{21}f_1+x_{22}f_2)}{df_1} & \frac{dQ(x_{21}f_1+x_{22}f_2)}{df_2} \end{bmatrix} = \begin{bmatrix} Q'(x_{11}f_1 + x_{12}f_2)x_{11} & Q'(x_{11}f_1 + x_{12}f_2)x_{12} \\ Q'(x_{21}f_1 + x_{22}f_2)x_{21} & Q'(x_{21}f_1 + x_{22}f_2)x_{22} \end{bmatrix}$$
- Or

$$\begin{bmatrix} \frac{dz_1}{df_1} & \frac{dz_1}{df_2} \\ \frac{dz_2}{df_1} & \frac{dz_2}{df_2} \end{bmatrix} = \begin{bmatrix} Q'(x_{11}f_1 + x_{12}f_2) & \\ & Q'(x_{21}f_1 + x_{22}f_2) \end{bmatrix}\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$$

$$\begin{bmatrix} \frac{dz_1}{df_1} & \frac{dz_1}{df_2} \\ \frac{dz_2}{df_1} & \frac{dz_2}{df_2} \end{bmatrix} = \boldsymbol{D}_Q \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$$

  - Where $\boldsymbol{D}_Q$ is a diagonal matrix with the derivative of the non-linearity evaluated at for each row of $\boldsymbol{X}$ (or for each element of $\boldsymbol{Xf}$).
- ***Therefore*** $\frac{d\boldsymbol{z}}{d\boldsymbol{f}} = \boldsymbol{D}_Q\mathbf{X}$

Gradients for Back Propagation

$$\frac{dJ(\boldsymbol{y})}{d\boldsymbol{g}} = \boldsymbol{Z}\frac{dJ(\boldsymbol{y})}{d\boldsymbol{y}}$$

$$\boldsymbol{g}_{k+1} = \boldsymbol{g}_k - \mu\boldsymbol{Z}\frac{dJ(\boldsymbol{y})}{d\boldsymbol{y}}$$

$$\frac{dJ(\boldsymbol{y})}{d\boldsymbol{f}} = \frac{d\boldsymbol{z}}{d\boldsymbol{f}}\frac{d\boldsymbol{y}}{d\boldsymbol{z}}\frac{dJ(\boldsymbol{y})}{d\boldsymbol{y}} = \boldsymbol{D}_Q\mathbf{XG}\frac{dJ(\boldsymbol{y})}{d\boldsymbol{y}}$$

$$\boldsymbol{f}_{k+1} = \boldsymbol{f}_k - \boldsymbol{D}_Q\mathbf{XG}\frac{dJ(\boldsymbol{y})}{d\boldsymbol{y}}$$