Convolution Networks

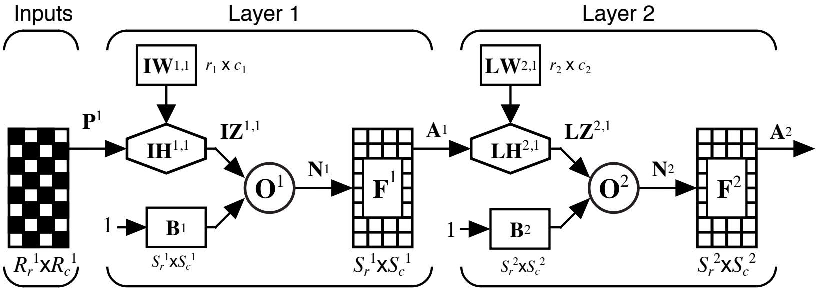
Machine Learning II

Lecture 8

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

- A convolution network is a multilayer feedforward network that has two- or three-dimensional inputs.
- It has weight functions that are not generally viewed as matrix multiplication (or inner product) operations.



- The principal layer type for convolution networks is the convolution layer.
- Let the input image be represented by the $R_r \times R_c$ matrix $\mathbf{V}$.
- The weight function for this layer performs a convolution operation on the image, using the convolution kernel that is represented by the $r \times c$ matrix $\mathbf{W}$.

$$z_{i,j} = \sum_{k=1}^{r} \sum_{l=1}^{c} w_{k,l} v_{i+k-1,j+l-1}$$

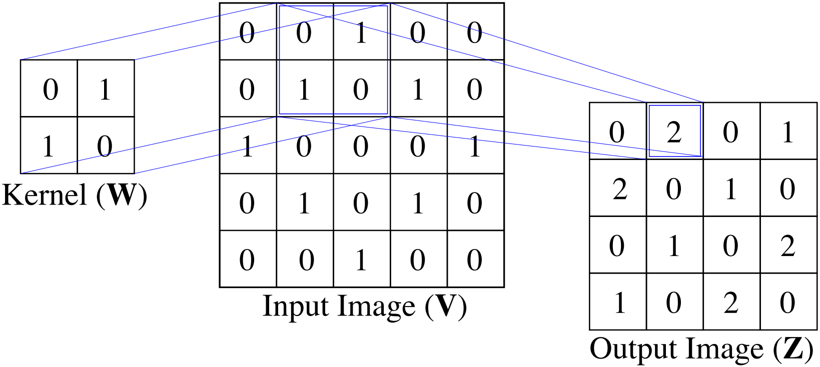
- In matrix form, we will write it as

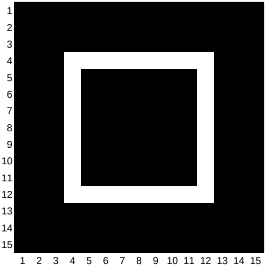
$$\mathbf{Z} = \mathbf{W} \circledast \mathbf{V}$$

Padding and stride

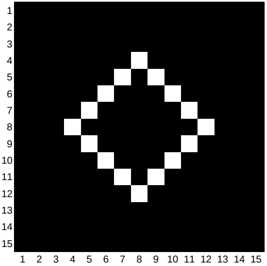
- Convolution will reduce the number of rows in the image by $r - 1$ and the number of columns by $c - 1$.
- To maintain image size, we can pad the outside of the image with zeros before convolving.
- The width of the zero padding is P^d.
- The output image can be made smaller by taking larger strides, or kernel movements. Normally, the kernel is moved one step at a time when performing the convolution. If the stride is increased to 2, the output image size is reduced by a factor of 2.
- The number of steps for the stride is S^t.

Example convolution



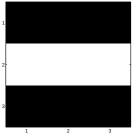


Square

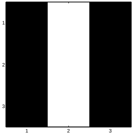


Diamond

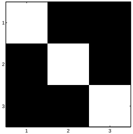
Convolution kernels (filters)



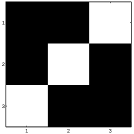
Horizontal kernel



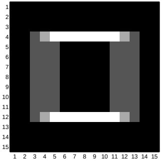
Vertical kernel



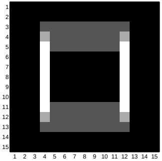
Slash kernel



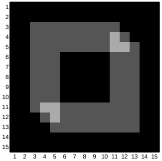
Backslash kernel



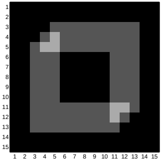
Horizontal filtered square



Vertical filtered square

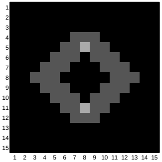


Slash filtered square

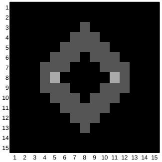


Backslash filtered square

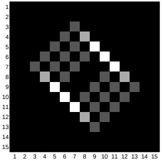
Filtered diamond



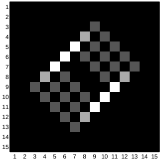
Horizontal filtered diamond



Vertical filtered diamond



Slash filtered diamond



Backslash filtered diamond

- A pooling (or subsampling) layer often follows a convolution layer and consolidates $r \times c$ elements in the input image to 1 element in the output image.
- The purpose is to reduce the spatial size of the feature map.
- This reduces the number of parameters in the network.

Average pooling

$$z_{i,j} = \left\{ \sum_{k=1}^{r} \sum_{l=1}^{c} v_{r(i-1)+k,c(j-1)+l} \right\} w$$

Matrix format

$$\mathbf{Z} = w \boxplus_{r,c}^{ave} \mathbf{V}$$

- For max pooling, the maximum of the elements in the consolidation window is used, rather than the sum.
- Recent studies have shown max pooling to be a little more effective in some applications than average pooling.

Max pooling

$$z_{i,j} = max\left\{v_{r(i-1)+k,c(j-1)+l} \middle| k = 1, ..., r; l = 1, ..., c\right\}$$

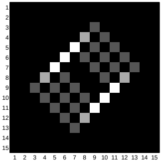
Matrix format

$$\mathbf{Z} = \boxplus_{r,c}^{max}\mathbf{V}$$

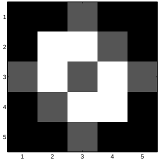
Pooling: choice of stride and size

- As in convolution operations, various strides can be used in pooling operations.
- Normally, the consolidation window is square, and the stride is equal to the window size, so there is no overlap in the consolidations.
- The most common choice is $r = 2$, $c = 2$ and $S^t = 2$.

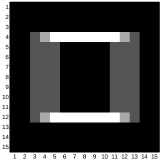
Max pooling (3x3) examples



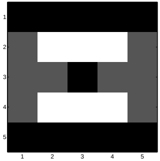
Backslash filtered diamond



Max pooled filtered diamond

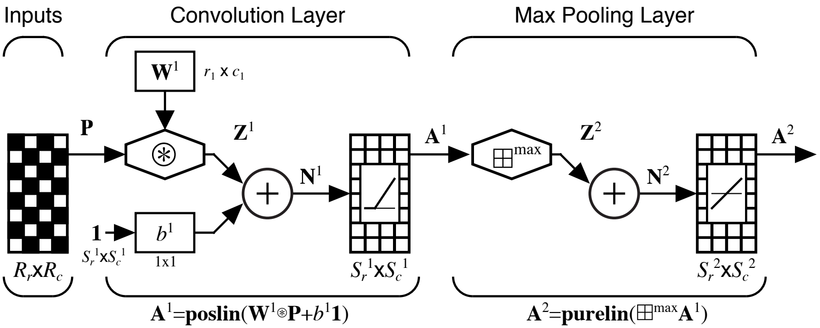


Horizontal filtered square

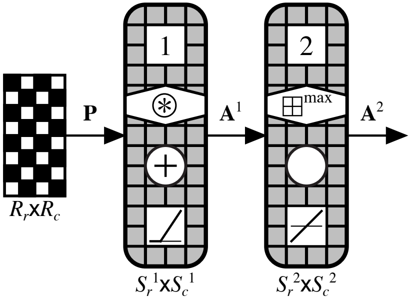


Max pooled filtered square

The following figure represents the combination of a convolution layer and a pooling layer. Notice that the bias in the convolution layer is a scalar, which is added to each element of $\mathbf{IZ}^{1,1}$.



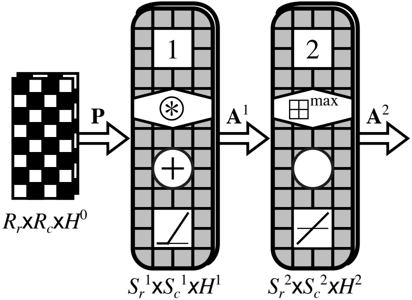
When many layers are involved, it is helpful to have a simplified notation to represent each layer.



Feature maps

- Each convolution kernel (weight) can identify one elemental feature in the input.
- Complex patterns will consist of combinations of many elemental features.
- Multiple kernels can be included in a single convolution layer to extract multiple features.
- LeCun, in his original developement, called the outputs of one kernel operation a feature map (FM).
- This idea can be extended to the input image. For example, a color image consists of red, green and blue planes.
- A convolution layer then takes a set of feature maps as an input, and produces another set of feature maps as an output.

Network diagram with feature maps

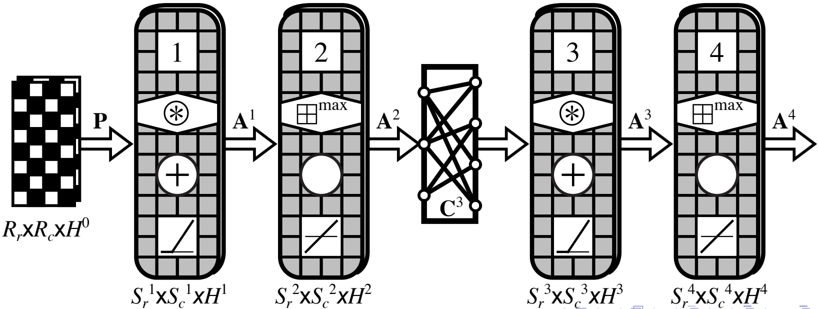


- Assume that there are H^m FMs in Layer m, and H^{m+1} FMs in layer $m+1$.
- If each FM in Layer m is connected to every FM in Layer $m+1$, then there would be $H^m \times H^{m+1}$ convolution kernels in Layer $m+1$.
- We can reduce the number of kernels by using only a subset of the possible connections.
- The connection matrix between Layer m and Layer $m+1$ will be denoted $\mathbf{C}^{m+1}$.
- Element $c_{i,j}^{m+1}$ will equal 1 when FM j in Layer m is connected to FM i in Layer $m+1$. Otherwise, it will equal 0.

$$z_{i,j}^{m,h} = \sum_{l \in C^{m,h}} \sum_{u=1}^{r^m} \sum_{v=1}^{c^m} w_{u,v}^{m,(h,l)} a_{i+u-1,j+v-1}^{m-1,l}$$

$C^{m,h}$ is the set of indices of FMs in layer $m-1$ that connect to FM h in Layer m (from row h of $\mathbf{C}^m$).

$$\mathbf{Z}^{m,h} = \sum_{l \in C^{m,h}} \mathbf{W}^{m,(h,l)} \circledast \mathbf{A}^{m-1,l}$$



Uses of connection matrices

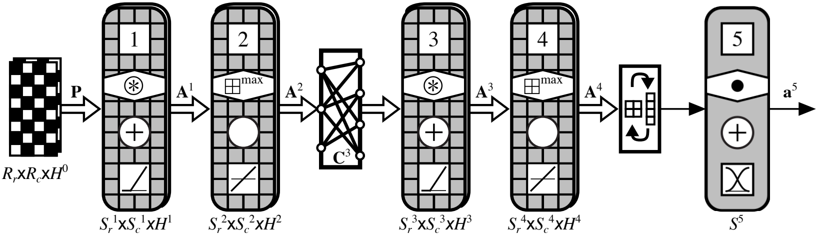
- Connection matrices are generally located between a pooling layer and the following convolution layer.
- If no connection block appears between a pooling layer and the following convolution layer, the FMs are fully connected.
- The number of FMs in a convolution layer and its following pooling layer are equal, and feature map h in the convolution layer is connected only to feature map h in the following pooling layer. No connection block is shown.

Conversion from matrix to vector

- In addition to the connection block, there is one other block that can appear between layers of a convolution network.
- It converts the output of a layer from a set of FMs to a single vector.
- It stacks the columns of the FMs on top of each other, starting from the first FM to the last. ($\mathbf{a}_i$ indicates the i^{th} column of matrix $\mathbf{A}$.)

$$vec(\mathbf{A}^m) = \left[\left(\mathbf{a}_1^{m,1}\right)^T \left(\mathbf{a}_2^{m,1}\right)^T \cdots \left(\mathbf{a}_{S_c^m}^{m,1}\right)^T \left(\mathbf{a}_1^{m,2}\right)^T \cdots \left(\mathbf{a}_{S_c^m}^{m,H^m}\right)^T \right]^T$$

The matrix to vector conversion is normally used before the final layer of a convolution network, which is a standard dot product (matrix multiplication) layer. (The same conversion block can indicate vector to matrix conversion, if needed.)

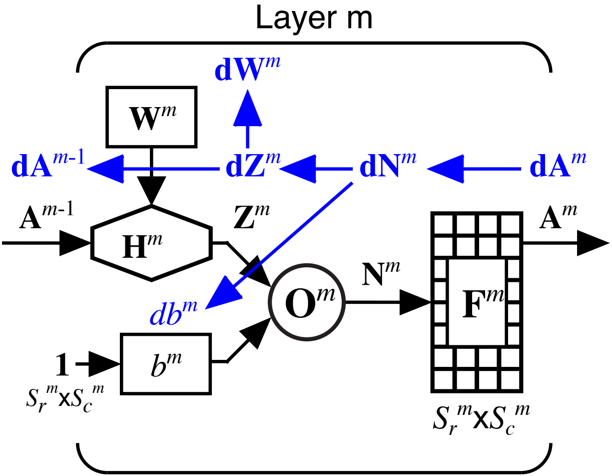


In order to compute the gradient of the performance with respect to the weights and biases, we need to perform a backpropagation operation. Because the net input is a matrix, we will use a different notation for sensitivity – $\mathbf{dN}$. Other derivatives will be defined as:

$$\mathbf{dN} \equiv \frac{\partial \hat{F}}{\partial \mathbf{N}}, \mathbf{dA} \equiv \frac{\partial \hat{F}}{\partial \mathbf{A}}, \mathbf{dZ} \equiv \frac{\partial \hat{F}}{\partial \mathbf{Z}}, \mathbf{dW} \equiv \frac{\partial \hat{F}}{\partial \mathbf{W}}, \mathbf{db} \equiv \frac{\partial \hat{F}}{\partial \mathbf{b}}$$

As we backpropagate across Layer m, we will be performing the following operations.

$$\mathbf{dA}^m \rightarrow \mathbf{dN}^m \rightarrow \mathbf{dZ}^m \rightarrow \mathbf{dA}^{m-1}$$


Layer m
$\mathbf{dW}^m$
$\mathbf{W}^m$
$\mathbf{dA}^{m\text{-}1}$
$\mathbf{dZ}^m$
$\mathbf{dN}^m$
$\mathbf{dA}^m$
$\mathbf{A}^{m\text{-}1}$
$\mathbf{H}^m$
$\mathbf{Z}^m$
$\mathbf{A}^m$
$\mathbf{N}^m$
$\mathbf{O}^m$
$\mathbf{F}^m$
db^m
$\mathbf{1}$
b^m
$S_r^m \mathrm{x} S_c^m$
$S_r^m \mathrm{x} S_c^m$

Backpropagating across the transfer function

$$dn_{i,j}^{m,h} \equiv \frac{\partial \hat{F}}{\partial n_{i,j}^{m,h}} = \frac{\partial a_{i,j}^{m,h}}{\partial n_{i,j}^{m,h}} \times \frac{\partial \hat{F}}{\partial a_{i,j}^{m,h}}$$

$$\frac{\partial \hat{F}}{\partial a_{i,j}^{m,h}} = da_{i,j}^{m,h}$$

$$dn_{i,j}^{m,h} = \dot{f}^{m,h}(n_{i,j}^{m,h}) \times da_{i,j}^{m,h}$$

Matrix form (**dN**)

$$\mathbf{dN}^{m,h} = \dot{\mathbf{F}}^{m,h} \circ \mathbf{dA}^{m,h}$$

Here $\circ$ is the Hadamard product, which is an element by element matrix multiplication.

Backpropagating across the summation net input function

Scalar form (dz)

$$dz_{i,j}^{m,h} = dn_{i,j}^{m,h}$$

Matrix form ($\mathbf{dZ}$)

$$\mathbf{dZ}^{m,h} = \mathbf{dN}^{m,h}$$

Scalar form (db)

$$db^{m,h} = \frac{\partial \hat{F}}{\partial b^{m,h}} = \frac{\partial \hat{F}}{\partial n_{i,j}^{m,h}} \times \frac{\partial n_{i,j}^{m,h}}{\partial b^{m,h}}$$

$$db^{m,h} = \sum_{i=1}^{S_r^m} \sum_{j=1}^{S_c^m} dn_{i,j}^{m,h}$$

Matrix form (db)

$$db^{m,h} = (\boxplus_{S_r^m, S_c^m} \mathbf{dN}^{m,h}))$$

Backpropagating across the convolution weight function

Scalar form (da)

$$da_{i,j}^{m-1,l} = \frac{\partial \hat{F}}{\partial a_{i,j}^{m,l}} = \frac{\partial \hat{F}}{\partial z_{u,v}^{m,h}} \times \frac{\partial z_{u,v}^{m,h}}{\partial a_{i,j}^{m-1,l}}$$

$$da_{i,j}^{m-1,l} = \sum_{h \epsilon C_b^{m,l}} \sum_{u=1}^{S_r^m} \sum_{v=1}^{S_c^m} dz_{i,j}^{m,h} w_{i-u+1,j-v+1}^{m,(h,l)}$$

$C_b^{m,l}$ – FMs in layer m where lth FM in layer $m-1$ connects.

Matrix form ($\mathbf{dA}$)

$$\mathbf{dA}^{m-1,l} = \sum_{h \epsilon C_b^{m,l}} (rot180(\mathbf{W}^{m,(h,l)})) \star (\mathbf{dZ}^{m,h})$$

Here rot180 means matrix is rotated by 180 degrees.

Scalar form (dw)

$$dw_{u,v}^{m,h,l} = \frac{\partial \hat{F}}{\partial w_{u,v}^{m,(h,l)}} = \frac{\partial \hat{F}}{\partial z_{i,j}^{m,h}} \times \frac{\partial z_{i,j}^{m,h}}{\partial w_{u,v}^{m,(h,l)}}$$

$$dW_{u,v}^{m,h,l} = \sum_{i=1}^{S_r^m} \sum_{j=1}^{S_c^m} dZ_{i,j}^{m,h} v_{u+i-1,v+j-1}^{m,l}$$

here l is feature map in layer $m-1$.

Matrix form (**dW**)

$$\mathbf{dW}^{m,h,l} = \mathbf{dZ}^{m,h} \star \mathbf{V}^{m,l}$$

Backpropagating across average pooling

Scalar form (da)

$$da^{m-1,h}_{r^{m-1}(i-1)+k,c^{m-1}(j-1)+l} = \frac{\partial \hat{F}}{\partial a^{m-1,h}_{i,j}} = \frac{\partial \hat{F}}{\partial z^{m,h}_{i,j}} \times \frac{\partial z^{m,h}_{i,j}}{\partial a^{m-1,h}_{i,j}}$$

$$= dZ^{m,h}_{i,j} \times w^{m,h}$$

For k=1 to r^m and l=1 to c^m.

Matrix form (**dA**)

$$\mathbf{dA}^{m-1,h} = \left(w^{m,h}\right) \boxtimes^{ave}_{r^m,c^m} \left(\mathbf{dZ}^{m,h}\right)$$

$\boxtimes^{ave}_{j,k}\mathbf{A}$ takes each element of the matrix $\mathbf{A}$ and expands to j rows and k columns (reverse of $\boxplus^{ave}_{j,k}\mathbf{A}$).

Gradient for average pooling weight

Scalar form (dw)

$$dw^{m,h} = \frac{\partial \hat{F}}{\partial w^{m,h}} = \frac{\partial \hat{F}}{\partial z_{i,j}^{m,h}} \times \frac{\partial z_{i,j}^{m,h}}{\partial w^{m,h}}$$

$$dw^{m,h} = \sum_{i=1}^{S_r^m} \sum_{j=1}^{S_c^m} dz_{i,j}^{m,h} \left\{ \sum_{k=1}^{r^m} \sum_{l=1}^{c^m} a_{r^m(i-1)+k,c^m(j-1)+l}^{m-1,h} \right\}$$

Matrix form (dw)

$$dw^{m,h} = \boxplus_{S_r^m,S_c^m}^{ave}(\mathbf{dZ}^{m,h} \circ (\boxplus_{r^m,c^m}^{ave} \mathbf{A}^{m-1,h}))$$

First it performs a reduction operation that produces a matrix of size S_r^m by S_c^m, and then, after a Hadamard matrix multiplication, it performs another reduction to produce the scalar gradient.