We discuss the generation of $\phi(\text{pad}(Z^{m,i}))$
im2col in Existing Packages I

- Due to the wide use of CNN, a subroutine for $\phi(\text{pad}(Z^m,^i))$ has been available in some packages.
- For example, MATLAB has a built-in function im2col that can generate $\phi(\text{pad}(Z^m,^i))$ for $s = 1$ and $s = h$ (width of filter).
- But this function cannot handle general $s$.
- Can we do a reasonably efficient implementation by ourselves?
For an easy description we consider

$$\operatorname{pad}(Z_{m,i}^{m,i}) = Z_{\text{in},i}^{\text{in},i} \rightarrow Z_{\text{out},i}^{\text{out},i} = \phi(Z_{\text{in},i}^{\text{in},i}).$$
Linear Indices and an Example I

For the matrix

\[ Z^{\text{in},i} \]

\[
\begin{bmatrix}
  z_{1,1,1} & z_{2,1,1} & \ldots & z_{a^{\text{in}},1,1} & z_{1,2,1} & \ldots & z_{a^{\text{in}},b^{\text{in}},1} \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
  z_{1,1,j} & z_{2,1,j} & \ldots & z_{a^{\text{in}},1,j} & z_{1,2,j} & \ldots & z_{a^{\text{in}},b^{\text{in}},j} \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
  z_{1,1,d^{\text{in}}} & z_{2,1,d^{\text{in}}} & \ldots & z_{a^{\text{in}},1,d^{\text{in}}} & z_{1,2,d^{\text{in}}} & \ldots & z_{a^{\text{in}},b^{\text{in}},d^{\text{in}}} 
\end{bmatrix}
\]

we count elements in a column-oriented way.
This leads to the following column-oriented linear indices of $Z^{in,i}$:

$$
\begin{bmatrix}
1 & d^{in} + 1 & \ldots & (b^{in}a^{in} - 1)d^{in} + 1 \\
2 & d^{in} + 2 & \ldots & (b^{in}a^{in} - 1)d^{in} + 2 \\
\vdots & \vdots & \ddots & \vdots \\
d^{in} & 2d^{in} & \ldots & (b^{in}a^{in})d^{in}
\end{bmatrix} \in R^{d^{in} \times a^{in}b^{in}}.
$$

(1)

We know every element in

$$
\phi(Z^{in,i}) \in R^{hhd^{in} \times a^{out}b^{out}},
$$

is extracted from $Z^{in,i}$.
Thus the task is to find the mapping between each element in $\phi(Z^{in,i})$ and a linear index of $Z^{in,i}$.

Consider an example with

$$a^{in} = 3, \quad b^{in} = 2, \quad d^{in} = 1.$$  

Because $d^{in} = 1$, we omit the channel subscript.

In addition, we omit the instance index $i$, so the image is

$$
\begin{bmatrix}
Z_{11} & Z_{12} \\
Z_{21} & Z_{22} \\
Z_{31} & Z_{32}
\end{bmatrix}.
$$ (2)
If $h = 2, s = 1$, two sub-images are

\[
\begin{bmatrix}
  z_{11} & z_{12} \\
  z_{21} & z_{22}
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
  z_{21} & z_{22} \\
  z_{31} & z_{32}
\end{bmatrix}
\]

By our earlier way of representing images,

\[
Z_{\text{in},i} = \begin{bmatrix}
  z^i_{1,1,1} & z^i_{2,1,1} & \cdots & z^i_{a_{\text{in}},b_{\text{in}},1} \\
  \vdots & \vdots & \ddots & \vdots \\
  z^i_{1,1,d_{\text{in}}} & z^i_{2,1,d_{\text{in}}} & \cdots & z^i_{a_{\text{in}},b_{\text{in}},d_{\text{in}}}
\end{bmatrix}
\]
we now have

\[ Z^{\text{in}} = \begin{bmatrix} z_{11} & z_{21} & z_{31} & z_{12} & z_{22} & z_{32} \end{bmatrix} \]

- The linear indices from (1) are

\[ \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}. \]
Recall that

\[ \phi(Z_{i}^{in}) = \begin{bmatrix}
    z_{1,1,1}^{i} & z_{1+s,1,1}^{i} \\
    z_{2,1,1}^{i} & z_{2+s,1,1}^{i} \\
    \vdots & \vdots \\
    z_{h,h,1}^{i} & z_{h+s,h,1}^{i} \\
    \vdots & \vdots \\
    z_{h,h,d^{in}}^{i} & z_{h+s,h,d^{in}}^{i}
\end{bmatrix}
\]

...
Therefore,

\[ \phi(Z^{in}) = \begin{bmatrix} z_{11} & z_{21} \\ z_{21} & z_{31} \\ z_{12} & z_{22} \\ z_{22} & z_{32} \end{bmatrix}. \]  

(3)

Let’s check linear indices using Matlab/Octave:

\[
\text{octave:8> reshape((1:6)', 3, 2)} \\
\text{ans =}
\]

\[
\begin{bmatrix} 1 & 4 \\ 2 & 5 \end{bmatrix}
\]
This gives linear indices of the image in (2):

```
octave:9> im2col(reshape((1:6)', 3, 2), [2,2], "sliding")
```

```
ans =

1  2
2  3
4  5
5  6
```

Here [2 2] is the filter size.
Clearly,

1
2
4
5

correspond to linear indices of the first column in $\phi(Z^{{\text{in}}})$; see (3)
Linear Indices and an Example X

- To handle all instances together, we store
  \[ Z_{\text{in},1}, \ldots, Z_{\text{in},l} \]
  as
  \[
  \begin{bmatrix}
    \text{vec}(Z_{\text{in},1}) & \ldots & \text{vec}(Z_{\text{in},l})
  \end{bmatrix}
  \] (4)

- For our example,
  \[
  \begin{bmatrix}
    \text{vec}(Z^{m,i}) = \\
    1 \\
    2 \\
    3 \\
    4 \\
    5
  \end{bmatrix}, \quad \begin{bmatrix}
    \text{vec}(\phi(Z^{m,i})) = \\
    1 \\
    2 \\
    4 \\
    5 \\
    2
  \end{bmatrix}
  \]
Denote (4) as a MATLAB matrix

\[ Z \]

Then

\[
\begin{bmatrix}
\text{vec}(\phi(Z^{m,1})) & \ldots & \text{vec}(\phi(Z^{m,l}))
\end{bmatrix}
\]

is simply

\[ Z(P,:) \]

in MATLAB, where we store the mapping by

\[ P = [1 \ 2 \ 4 \ 5 \ 2 \ 3 \ 5 \ 6]^T \]
Linear Indices and an Example XII

- All instances are handled in one line
- Moreover, we hope that Matlab’s implementation on this mapping operation is efficient
- But how to obtain $P$?
- Note that

$$\begin{bmatrix} 1 & 2 & 4 & 5 & 2 & 3 & 5 & 6 \end{bmatrix}^T.$$

also corresponds to column indices of non-zero elements in $P^{m}_\phi$. 
Each row has a single non-zero

The column index of the non-zero indicates the extraction of elements from $Z^{in}$ to $\phi(Z^{in})$
Finding the Mapping I

- We begin with checking how linear indices of $Z^{in,i}$ can be mapped to the first column of $\phi(Z^{in,i})$.
- For simplicity, we consider channel $j$ first.
- From

$$Z^{in,i} = \begin{bmatrix}
  z_{1,1,1} & z_{2,1,1} & \cdots & z_{a^{in},1,1} & z_{1,2,1} & \cdots & z_{a^{in},b^{in},1} \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
  z_{1,1,j} & z_{2,1,j} & \cdots & z_{a^{in},1,j} & z_{1,2,j} & \cdots & z_{a^{in},b^{in},j} \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
  z_{1,1,d^{in}} & z_{2,1,d^{in}} & \cdots & z_{a^{in},1,d^{in}} & z_{1,2,d^{in}} & \cdots & z_{a^{in},b^{in},d^{in}}
\end{bmatrix}$$
for the first column of $\phi(Z^{in,i})$, for channel $j$, we select an $h \times h$ matrix from $Z^{in,i}$. 
The selected values and their linear indices are

<table>
<thead>
<tr>
<th>values</th>
<th>linear indices in $Z^{in,i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z_{1,1,j}$</td>
<td>$j$</td>
</tr>
<tr>
<td>$Z_{2,1,j}$</td>
<td>$d^{in} + j$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$Z_{h,1,j}$</td>
<td>$(h - 1)d^{in} + j$</td>
</tr>
<tr>
<td>$Z_{1,2,j}$</td>
<td>$a^{in}d^{in} + j$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$Z_{h,2,j}$</td>
<td>$((h - 1) + a^{in})d^{in} + j$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$Z_{h,h,j}$</td>
<td>$((h - 1) + (h - 1)a^{in})d^{in} + j$</td>
</tr>
</tbody>
</table>
Finding the Mapping IV

We take $d^{in}$ out and rewrite the earlier table as

$$
\begin{pmatrix}
0 + 0a^{in} \\
\vdots \\
(h - 1) + 0a^{in} \\
0 + 1a^{in} \\
\vdots \\
(h - 1) + 1a^{in} \\
\vdots \\
0 + (h - 1)a^{in} \\
\vdots \\
(h - 1) + (h - 1)a^{in}
\end{pmatrix}
\begin{pmatrix}
0 \\
\vdots \\
h - 1 \\
0 \\
\vdots \\
h - 1 \\
0 \\
\vdots \\
h - 1
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
\vdots \\
h - 1 \\
0 \\
\vdots \\
h - 1 \\
0 \\
\vdots \\
h - 1
\end{pmatrix}
\begin{pmatrix}
0 \\
\vdots \\
h - 1 \\
0 \\
\vdots \\
h - 1 \\
0 \\
\vdots \\
h - 1
\end{pmatrix}
\begin{pmatrix}
0 \\
\vdots \\
h - 1 \\
0 \\
\vdots \\
h - 1 \\
0 \\
\vdots \\
h - 1
\end{pmatrix}
(6)
$$
We will see that the right-side of (6) will be used in the practical implementation.

Every linear index in (6) can be represented as

\[(p + qa^\text{in})d^\text{in} + j,\]  \hspace{1cm} (7)

where

\[p, q \in \{0, \ldots, h - 1\}\]

Thus \((p + 1, q + 1)\) corresponds to the pixel position in the first sub-image (or say the convolutional filter).
Next we consider other columns in $\phi(Z^{in,i})$ by still fixing the channel to be $j$. 
From

$$\phi(Z^{\text{in},i}) =$$

$$\begin{bmatrix}
\vdots & \vdots & \vdots & \vdots \\
Z_{1,1,j}^i & Z_{1+s,1,j}^i & Z_{1+(a_{\text{out}}-1)s,1+(b_{\text{out}}-1)s,j}^i \\
Z_{2,1,j}^i & Z_{2+s,1,j}^i & Z_{2+(a_{\text{out}}-1)s,1+(b_{\text{out}}-1)s,j}^i \\
\vdots & \vdots & \vdots & \vdots \\
Z_{h,h,j}^i & Z_{h+s,h,j}^i & Z_{h+(a_{\text{out}}-1)s,h+(b_{\text{out}}-1)s,j}^i \\
\vdots & \vdots & \vdots & \vdots 
\end{bmatrix}$$
each column contains the following elements from the $j$th channel of $Z^{in,i}$.

$$Z_{1+p+as,1+q+bs,j}, \quad a = 0, 1, \ldots, a^{out} - 1,$$
$$b = 0, 1, \ldots, b^{out} - 1, \quad (8)$$
$$p, \quad q \in \{0, \ldots, h - 1\}.$$

Clearly, when

$$p = q = 0,$$

we have that

$$(1 + as, 1 + bs)$$
Finding the Mapping IX

is the top-left position of a sub-image in the channel $j$ of $Z^\text{in,}i$.

- This is reasonable as we now use stride $s$ to generate $a^\text{out} b^\text{out}$ sub-images
- The linear index of each element in (8) is

$$
\left( (1 + p + as - 1) + (1 + q + bs - 1)a^\text{in} \right) d^\text{in} + j
$$

column index in $Z^\text{in,}i - 1$

$$
= ((p + as) + (q + bs)a^\text{in})d^\text{in} + j
$$

$$
= (a + ba^\text{in})sd^\text{in} + (p + qa^\text{in})d^\text{in} + j.
$$

(9)

see (7)
Finding the Mapping X

- Now we have known for each element of $\phi(Z^{in,i})$ what the corresponding linear index in $Z^{in,i}$ is.
- Next we discuss the implementation details
First, we compute elements in (6) with \( j = 1 \) by MATLAB’s ‘+’ operator.

This operator has implicit expansion behavior to compute the outer sum of two arrays.

From the second part of (6), we calculate the outer sum of

\[
\begin{bmatrix}
0 \\
\vdots \\
h - 1
\end{bmatrix} d^{\text{in}} + 1
\]

and

\[
\begin{bmatrix}
0 & \ldots & h - 1
\end{bmatrix} a^{\text{in}} d^{\text{in}}
\]
The result is the following matrix

\[
\begin{bmatrix}
1 & a^{\text{in}}d^{\text{in}} + 1 & \ldots & (h - 1)a^{\text{in}}d^{\text{in}} + 1 \\
 d^{\text{in}} + 1 & (1 + a^{\text{in}})d^{\text{in}} + 1 & \ldots & (1 + (h - 1)a^{\text{in}})d^{\text{in}} + 1 \\
 \vdots & \vdots & \ddots & \vdots \\
 (h - 1)d^{\text{in}} + 1 & ((h - 1) + a^{\text{in}})d^{\text{in}} + 1 & \ldots & ((h - 1) + (h - 1)a^{\text{in}})d^{\text{in}} + 1 
\end{bmatrix}
\]

(10)

- If columns are concatenated, we get (6) with \( j = 1 \)
- To get (7) for all channels \( j = 1, \ldots, d^{\text{in}} \), we have

\[
\begin{bmatrix}
\text{vec}((10)) + 0 \\
\vdots \\
\text{vec}((10)) + d^{\text{in}} - 1
\end{bmatrix}
\]
This can be computed by the following outer sum:

$$\text{vec}((10)) + \begin{bmatrix} 0 & 1 & \ldots & d^{\text{in}} - 1 \end{bmatrix} \quad (11)$$

Note that in (11) we have

$$0, \ldots, d^{\text{in}} - 1$$

instead of

$$1, \ldots, d^{\text{in}}$$

because in (10) we have already done “+1” for \( j = 1 \)
Now we have linear indices of the first column of $\phi(Z_{\text{in},i})$.

Next, we obtain other columns in $\phi(Z_{\text{in},i})$.

In the linear indices in (9), the second term corresponds to indices of the first column, while the first term is the following column offset:

$$(a + ba_{\text{in}})sd_{\text{in}}, \quad \forall a = 0, 1, \ldots, a_{\text{out}} - 1, \quad b = 0, 1, \ldots, b_{\text{out}} - 1.$$
This column offset is the outer sum of the following two arrays.

\[
\begin{bmatrix}
    0 \\
    \vdots \\
    a^{\text{out}} - 1
\end{bmatrix} \times s^{\text{in}} \quad \text{and} \quad \begin{bmatrix}
    0 & \ldots & b^{\text{out}} - 1
\end{bmatrix} \times a^{\text{in}} s^{\text{in}}
\]

Finally, we compute the outer sum of

- the column offset and
- the linear indices in the first column of \( \phi(Z^{\text{in},i}) \)
A MATLAB Implementation VI

in the following operation

$$\text{vec}((12))^T + \text{vec}((11)),$$

(13)

where

$$\text{vec}((12)) \in R^{a_{\text{out}}b_{\text{out}} \times 1} \text{ and } \text{vec}((11)) \in R^{hhd_{\text{in}} \times 1}$$

In the end we store

$$\text{vec}((13)) \in R^{hhd_{\text{in}}a_{\text{out}}b_{\text{out}} \times 1}$$

It is a vector collecting
column index of the non-zero in each row of $P^m_\phi$

- Note that each row in the 0/1 matrix $P^m_\phi$ contains exactly only one non-zero element.
- See the example in (5)
- The obtained linear indices are independent of the values of $Z^{in,i}$.
- Thus the above procedure only needs to be run once in the beginning.
function idx = find_index_phiZ(a,b,d,h,s)

first_channel_idx = ([0:h-1]'*d+1) + [0:h-1]*a*d;
first_col_idx = first_channel_idx(:) + [0:d-1];
a_out = floor((a - h)/s) + 1;
b_out = floor((b - h)/s) + 1;
column_offset = ([0:a_out-1]' + [0:b_out-1]*a)*s*d;
idx = column_offset(:)' + first_col_idx(:);
idx = idx(:);
The code is simple and short
We assume that Matlab operations used here are efficient and so is our resulting code
But is that really the case?
We will do experiments to check this
Some works have tried to do similar things (e.g., https://github.com/wiseodd/hipsternet), though we don’t see complete documents and evaluation