Convolution, De-convolution, Transposed convolution, Fractional-stride convolution
Convolution (no padding, stride=1)

- Input: 4x4, Filter: 3x3, Output: 2x2 (2=4-3+1)
Convolution (padding, stride=1)

- Input: 5x5, Filter: 3x3, Output: 5x5
Convolution (padding, stride=2)

- Input: 6x6, Filter: 3x3, Output: 3x3
TRANSPOSED CONVOLUTION
Motivations

• Need to use a transformation going in the opposite direction of a normal convolution
  • Decoding layer of a convolutional auto-encoder
  • Project feature maps to a higher-dimensional space (up-sampling)
Operation

- Input: 5x5, Filter: 3x3, Output: 5x5, Stride=1
WHY IT IS CALLED TRANSPOSED CONV.
FRACTIONAL-STRIDE CONV.
Transposed convolution with stride

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$c$</td>
<td>$d$</td>
<td>$e$</td>
<td>$f$</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccccccc}
  & w_1 \times a & w_2 \times a & w_3 \times a \\
  & w_1 \times b & w_2 \times b & w_3 \times b \\
  & w_1 \times c & w_2 \times c & w_3 \times c \\
  & w_1 \times d & w_2 \times d & w_3 \times d \\
  & w_1 \times e & w_2 \times e & w_3 \times e \\
  & w_1 \times f & w_2 \times f & w_3 \times f \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  w_2a + w_1b \\
  w_3a + w_2b + w_1c \\
  w_3b + w_2c + w_1d \\
  w_3c + w_2d + w_1e \\
  w_3d + w_2e + w_1f \\
  w_3e + w_2f \\
\end{array}
\]
### Transposed convolution with stride

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
<td>(d)</td>
<td>(e)</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
  w_3a + w_1b & + w_1c \\
  w_3a + w_2b & + w_1d \\
  w_3b + w_2c & + w_1e \\
  w_3c + w_2d & + w_1f \\
  w_3e + w_2f & \\
\end{align*}
\]

: Transpose filter & Do the convolution!
Transposed convolution with stride

| a | 0 | b | 0 | c | 0 | d | 0 | e | 0 | f |

\[
\begin{array}{cccccc}
  w_1 \times a & w_2 \times a & w_3 \times a \\
  w_1 \times b & w_2 \times b & w_3 \times b \\
  w_1 \times c & w_2 \times c & w_3 \times c \\
  w_1 \times d & w_2 \times d & w_3 \times d \\
  w_1 \times e & w_2 \times e & w_3 \times e \\
  w_1 \times f & w_2 \times f & w_3 \times f \\
\end{array}
\]

\[
\begin{array}{ccccccc}
  w_2 a & w_3 a & w_2 b & w_3 b + w_1 b & w_2 c & w_3 c + w_1 c & w_2 d & w_3 d + w_1 d & w_2 e & w_3 e + w_1 e & w_2 f
\end{array}
\]
Transposed convolution with stride

\[
\begin{array}{cccccccc}
a & 0 & b & 0 & c & 0 & d & 0 & e & 0 & f \\
\end{array}
\]

\[
\begin{array}{ccc}
w_3 & w_2 & w_1 \\
\end{array}
\]: Transpose filter & Do the convolution!
TRANSPOSED CONVOLUTION SIDE EFFECTS

http://distill.pub/2016/deconv-checkerboard/
Checkerboard Artifacts

Radford, et al., 2015
Salimans et al., 2016
Donahue, et al., 2016
Dumoulin, et al., 2016
Alternatives

• The same holds for
  • Back propagation

Solutions

• Transposed convolution → Resize convolution
• Jittering
IMPLEMENTATION
In TensorFlow

In tensorflow,

```python
def deconv2d(value, filter, output_shape, strides, padding="SAME",
             name=None):
    ...
    return gen_nn_ops.conv2d_backprop_input(input_sizes=output_shape,
                                             filter=filter,
                                             out_backprop=value,
                                             strides=strides,
                                             padding=padding,
                                             name=name)
```

- Performing transposed convolution, by putting input into the backpropagation operation.