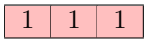
Exercise 7: Up and down convolution - Solution

Name: ..... UTID: .....

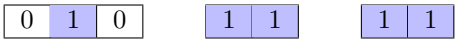
Let's look at some example of up and down (strided) convolutions. For simplicity we'll look at one dimensional convolutions, but all concepts generalize to higher dimensions. For all examples, the input and output channels of convolutions are 1.

Consider the three input vectors.

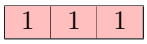
Let's convolve each with a kernel of ones (no bias) with kernel size 3, stride 2, and no padding.



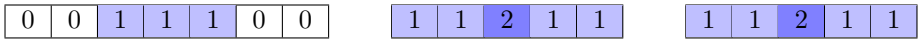
a) Visually and numerically show the results of the convolution for each input.



b) Next we run an up-convolution with a kernel of ones (no bias) with kernel size 3, stride 2, and no padding on each of the outputs of a). Hint: The output dimension is similar to the original input.



Show the results again visually and numerically.



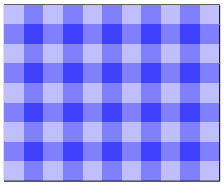
You should notice two things. First, not all output dimensions match the original input.

c) Briefly explain why this is the case. How would you address this?

Strided convolution rounds the output dimensions down. In our example both the length 6 and length 5 vector are mapped to a length 2 vector. Up-convolution has no way to telling the difference between the two vectors.

The address this, in pytorch the ConvTranspose2d operations has an additional argument ***output_padding****. It allows you to add a value to the output dimension. For the size 6 input you would use* ***output_padding=1*** *to obtain a size 6 output.*

Seccond, outputs are not all the same. They have a checkerboard pattern. You can see a visualization of this effect on a 2D image on the right. This can significantly deterirate the quality of a produced image.



d) Briefly explain where these checkerboard artifacts come from. Feel free to use your one dimensional example.

Each output is the sum of a different number of inputs in up-convolutions. Patterns with more inputs generally produce larger values (unless the network weights are carefully calibrated).

The typical solution to those upsampling artifacts is to first interpolate, and then convolve the interpolated (higher resolution) output. The two most common interpolations are nearest neighbor and (bi)linear. Consider the following low resolution input $\boxed{0\ 1\ 1}$. Nearest neighbor interpolation produces $\boxed{0\ 0\ 1\ 1\ 1}$, linear interpolation $\boxed{0\ \frac{1}{2}\ 1\ 1\ 1}$.

e) Run a stride 1 convolution with a kernel of ones with kernel size 3 and padding 2 on interpolated values. Draw and numerically compute the output for both linear and nearest upsampling. Hint: there are 7 output values.

$\boxed{0\ 0\ 1\ 2\ 3\ 2\ 1}$ *and* $\boxed{0\ \frac{1}{2}\ 1.5\ 2.5\ 3\ 2\ 1}$

f) One final issue you're facing is that twowards the side of the resulting vector the values become smaller. Why is this? How do you prevent this?

Padding with zeros leads to smaller responses. Using the leftmost (or rightmost) value for padding can prevent this. This is known as 'replicate' padding. Here is the the result of a convolution with replicate padding: $\boxed{0\ 0\ 1\ 2\ 3\ 3\ 3}$ *and* $\boxed{0\ \frac{1}{2}\ 1.5\ 2.5\ 3\ 3\ 3}$

Another option is the mirror the input vector when padding. This is known as 'reflection' padding. This option is generally preferred on images, as the resulting padded image still looks like a natural image, while a long streak of repeated color values does not.