



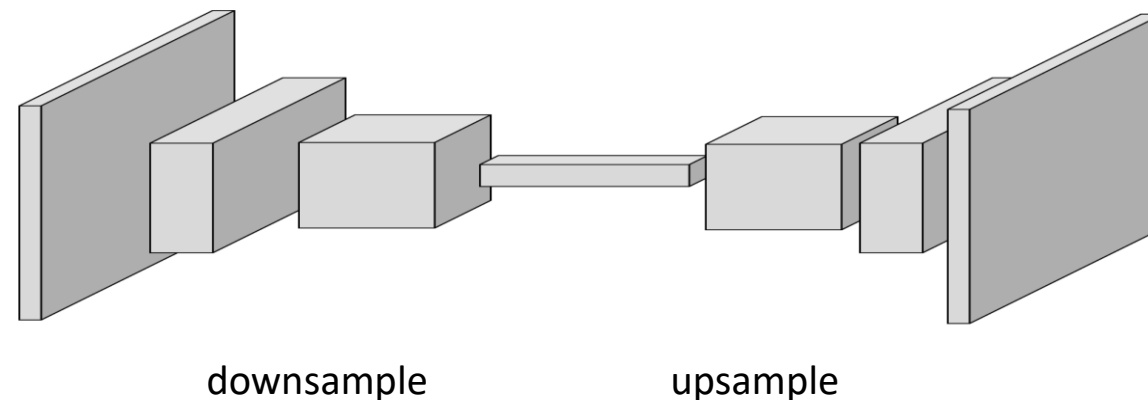
Convolutional Autoencoder

Industrial AI Lab.

Prof. Seungchul Lee

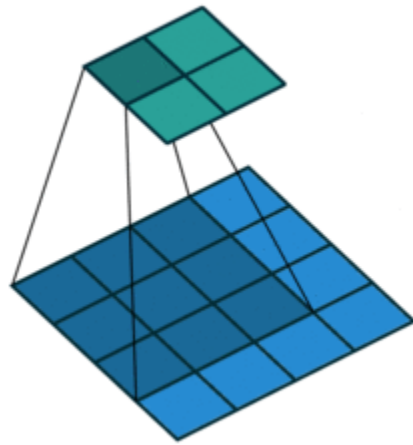
Convolutional Autoencoder

- Convolutional autoencoder extends the basic structure of the simple autoencoder by changing the fully connected layers to convolution layers.
 - the size of the input layer is also the same as output layers
 - the network of encoder change to convolution layers
 - the network of decoder change to **transposed** convolutional layers
 - A transposed 2-D convolution layer upsamples feature maps.
 - This layer is sometimes incorrectly known as a "deconvolution" or "deconv" layer.
 - This layer is the transpose of convolution and does not perform deconvolution.

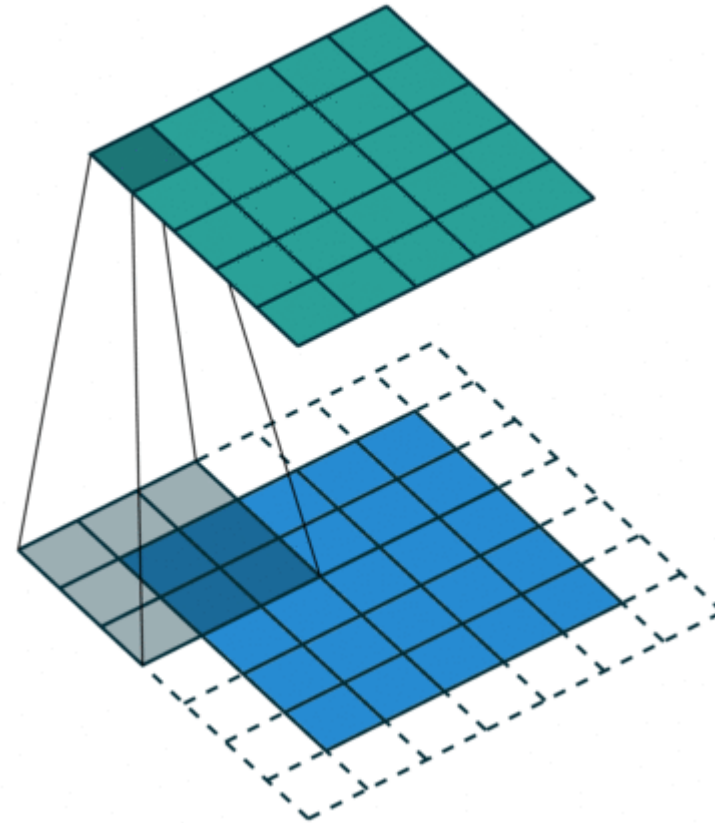


tf.nn.conv2d

- encoder
- padding



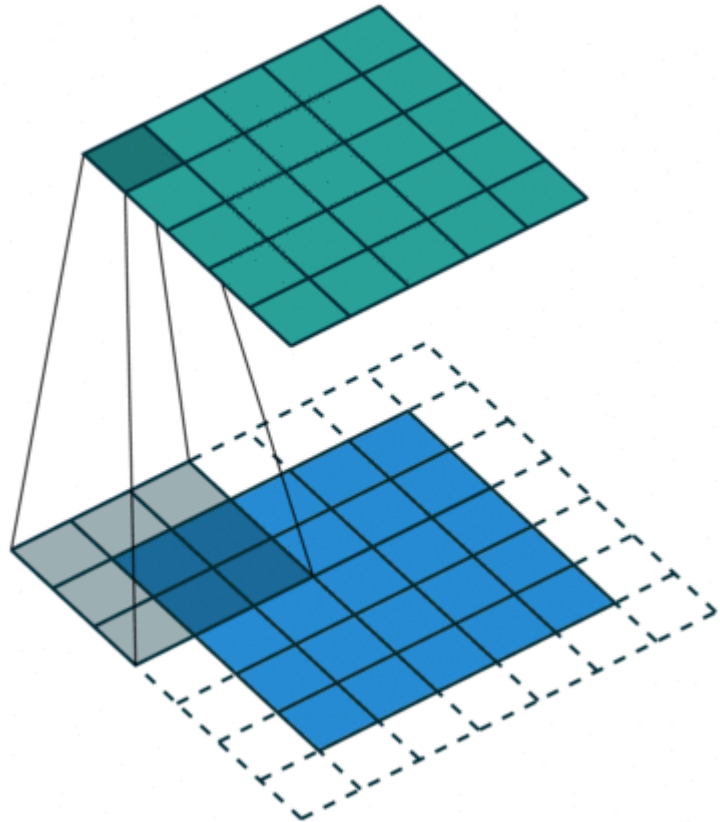
padding = 'VALID'
strides = [1, 1, 1, 1]



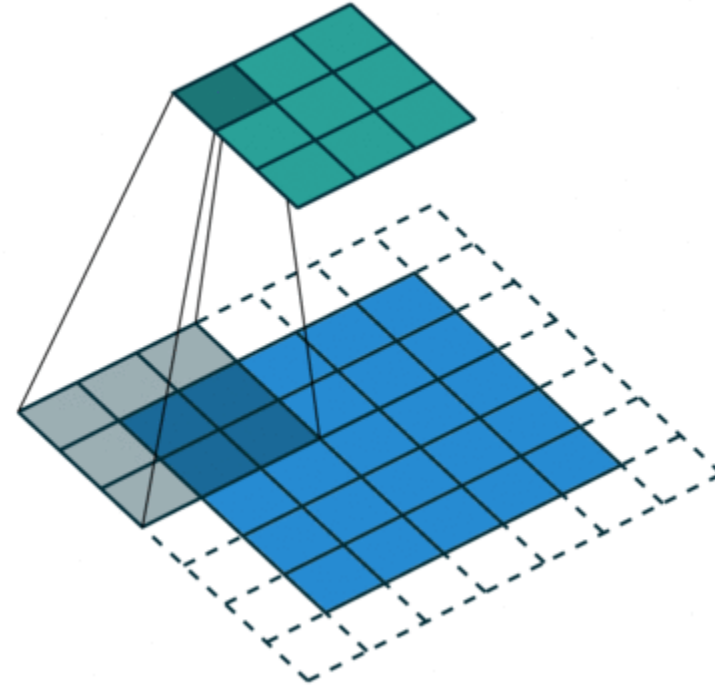
padding = 'SAME'
strides = [1, 1, 1, 1]

tf.nn.conv2d

- encoder
- stride



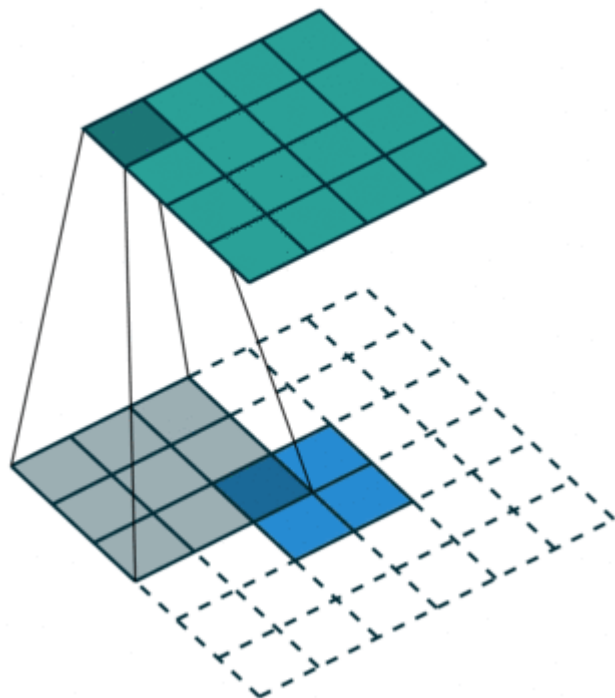
padding = 'SAME'
strides = [1, 1, 1, 1]



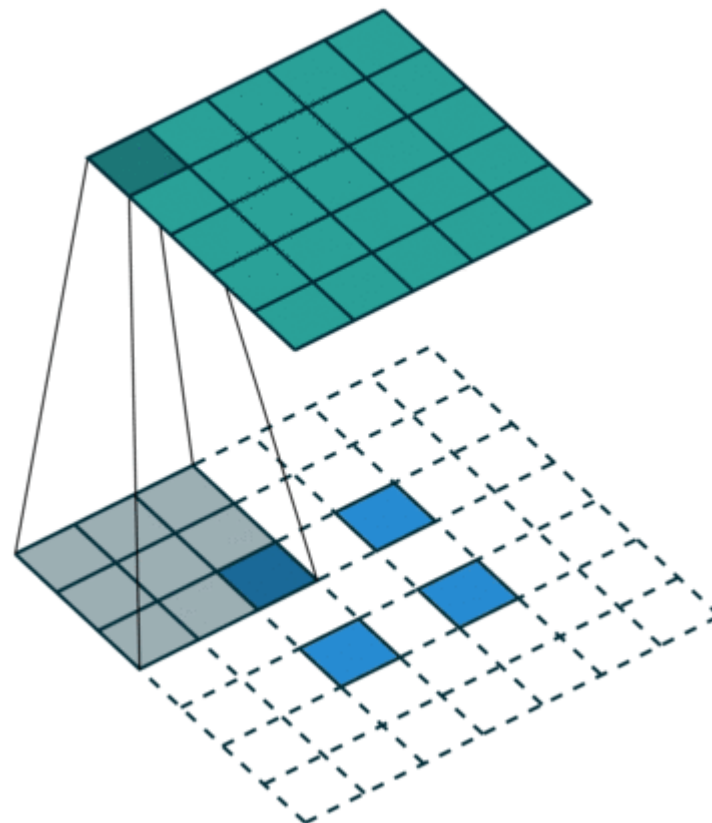
padding = 'SAME'
strides = [1, 2, 2, 1]

tf.nn.conv2d_transpose

- decoder
- stride



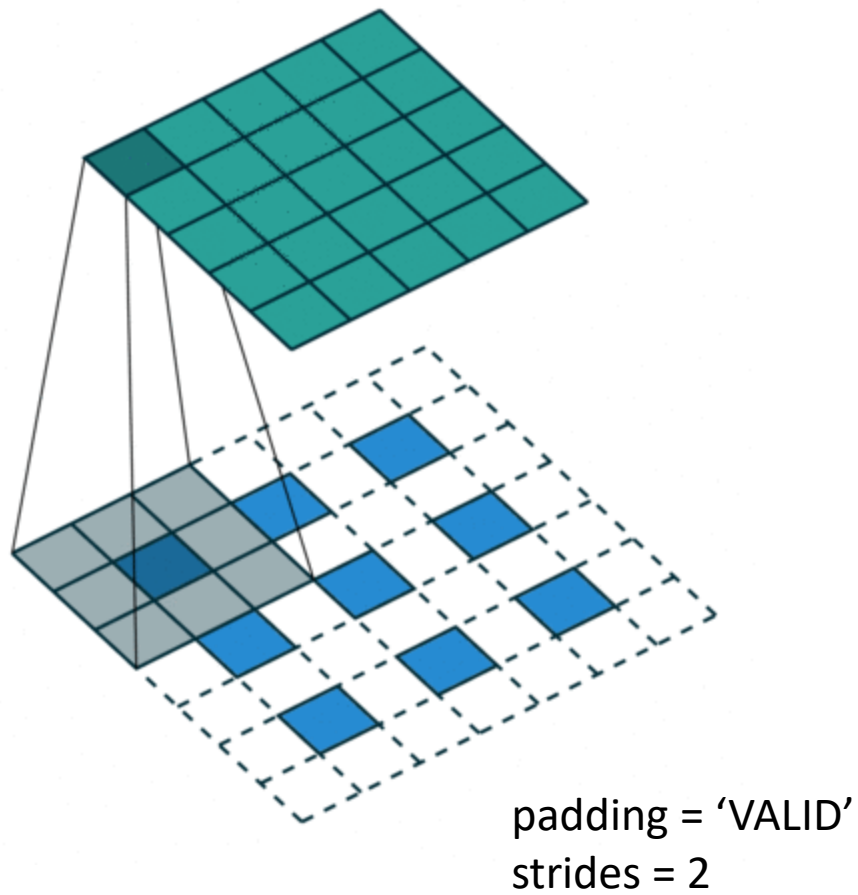
padding = 'VALID'
no strides



padding = 'VALID'
strides = 2

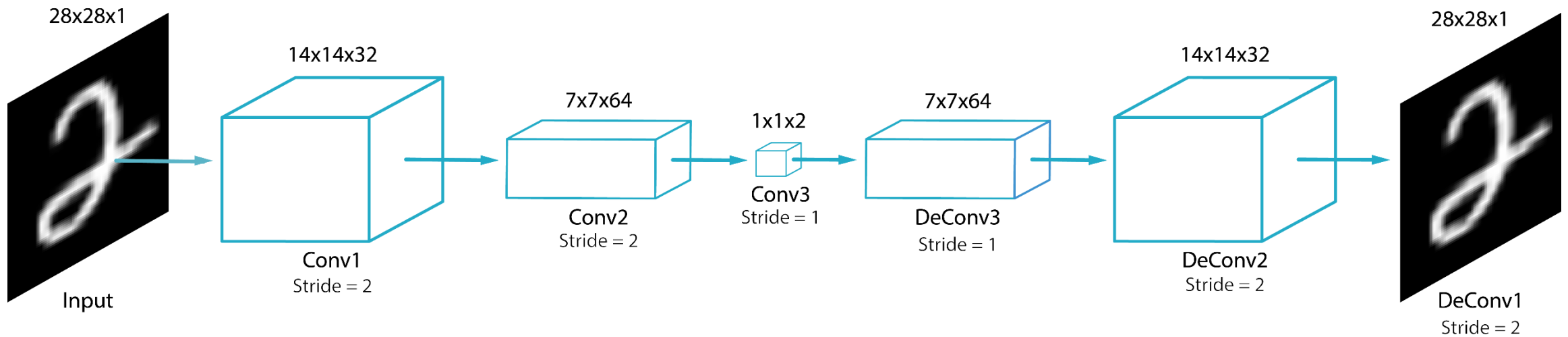
tf.nn.conv2d_transpose

- decoder
- stride



CAE Implementation

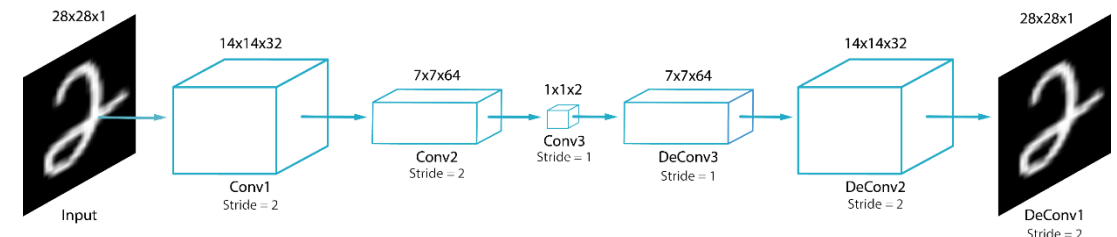
- Fully convolutional
- Note that no dense layer is used



CAE Implementation

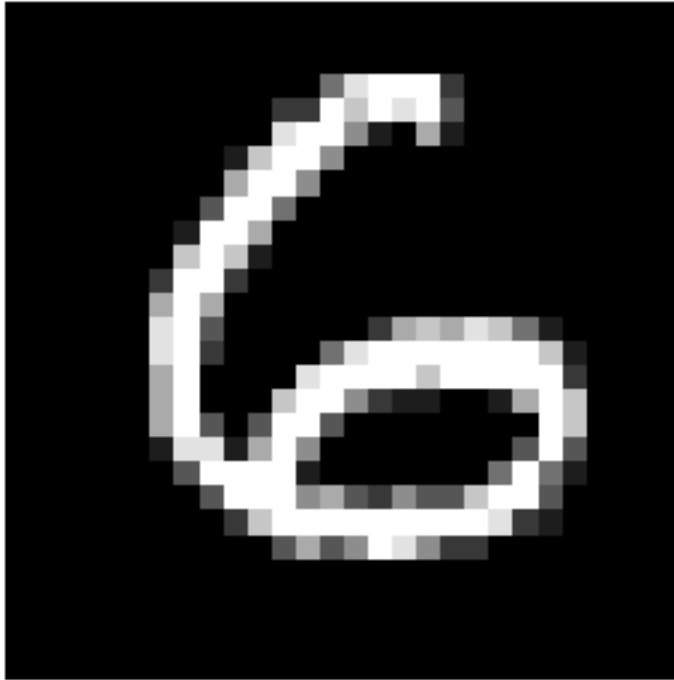
```
def encoder(x):  
    ## First convolution layer  
    conv1 = tf.layers.conv2d(inputs = x,  
                             filters = 32,  
                             kernel_size = [3, 3],  
                             padding = "SAME",  
                             activation = tf.nn.relu)  
    maxp1 = tf.layers.max_pooling2d(inputs = conv1,  
                                    pool_size = [2, 2],  
                                    strides = 2)  
  
    ## Second convolution layer  
    conv2 = tf.layers.conv2d(inputs = maxp1,  
                             filters = 64,  
                             kernel_size = [3, 3],  
                             padding = "SAME",  
                             activation = tf.nn.relu)  
    maxp2 = tf.layers.max_pooling2d(inputs = conv2,  
                                    pool_size = [2, 2],  
                                    strides = 2)  
  
    conv3 = tf.layers.conv2d(inputs = maxp2,  
                             filters = 2,  
                             kernel_size = [7, 7],  
                             padding = "VALID",  
                             use_bias = False)  
  
    return conv3
```

```
def decoder(latent):  
    deconv3 = tf.layers.conv2d_transpose(inputs = latent,  
                                         filters = 64,  
                                         kernel_size = [7, 7],  
                                         padding = 'VALID',  
                                         activation = tf.nn.relu)  
  
    ## First deconvolution layer  
    deconv2 = tf.layers.conv2d_transpose(inputs = deconv3,  
                                         filters = 32,  
                                         kernel_size = [14, 14],  
                                         strides = (2, 2),  
                                         padding = 'SAME',  
                                         activation = tf.nn.relu)  
  
    ## Second deconvolution layer  
    deconv1 = tf.layers.conv2d_transpose(inputs = deconv2,  
                                         filters = 1,  
                                         kernel_size = [28, 28],  
                                         strides = (2, 2),  
                                         padding = 'SAME')  
  
    return deconv1
```



Reconstruction Result

Input image



Reconstructed image

