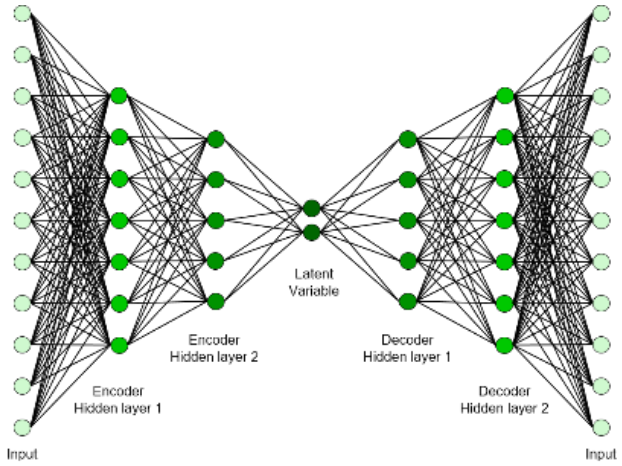
Convolutional Autoencoder (CAE)

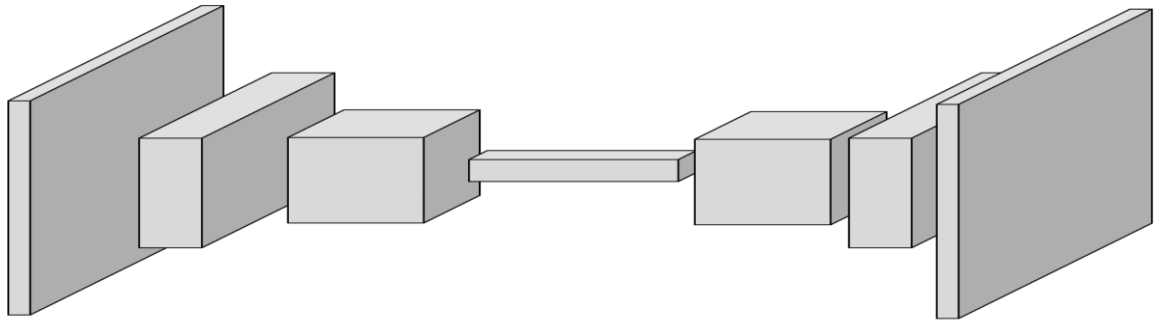
Prof. Seungchul Lee

Industrial AI Lab.

Convolutional Autoencoder

- Motivation: image to autoencoder ?
- Convolutional autoencoder extends the basic structure of the simple autoencoder by changing the fully connected layers to convolution 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.

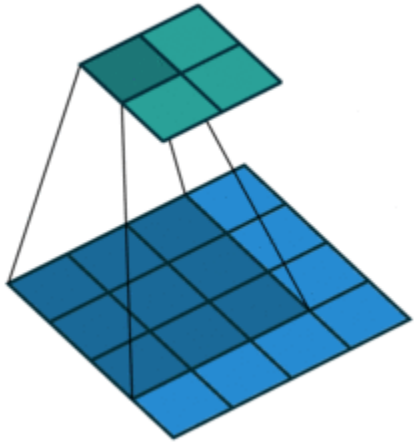




downsample upsample

tf.nn.conv2d

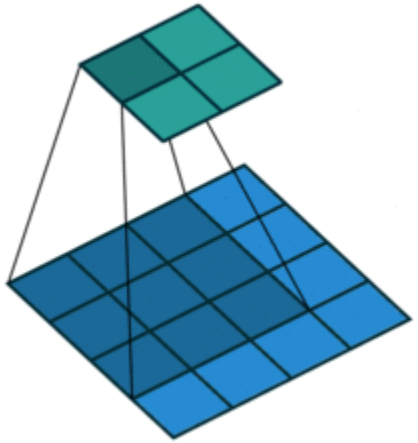
- Encoder
- Padding



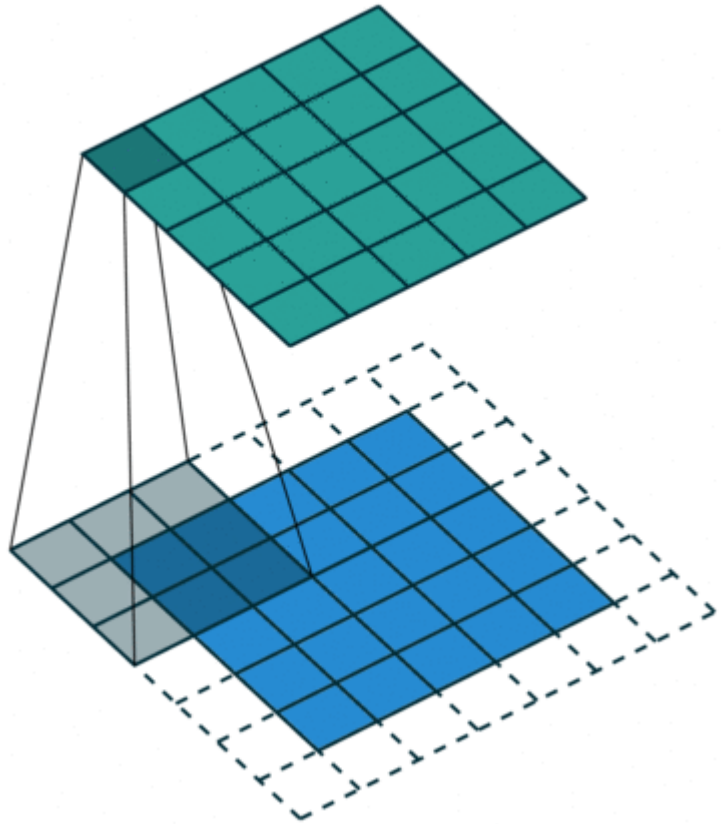
padding = ‘VALID’
strides = [1, 1, 1, 1]

tf.nn.conv2d

- Encoder
- Padding



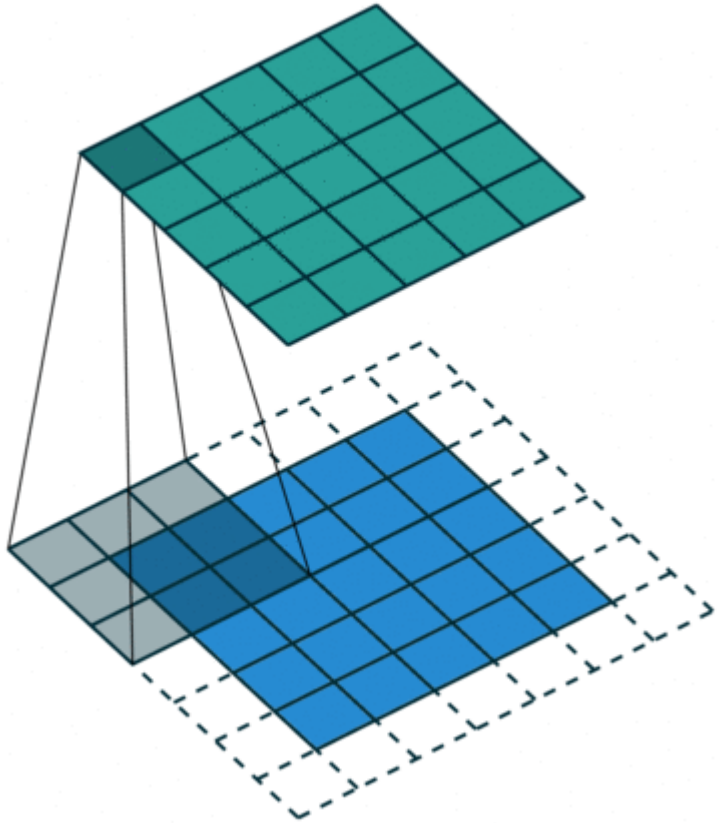
padding = ‘VALID’
strides = [1, 1, 1, 1]



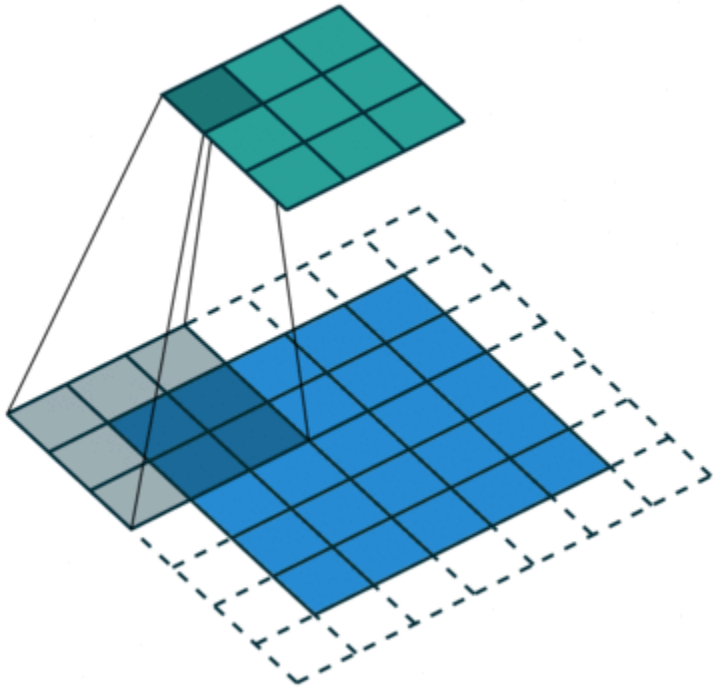
padding = ‘SAME’
strides = [1, 1, 1, 1]

tf.nn.conv2d

- Encoder
- Stride



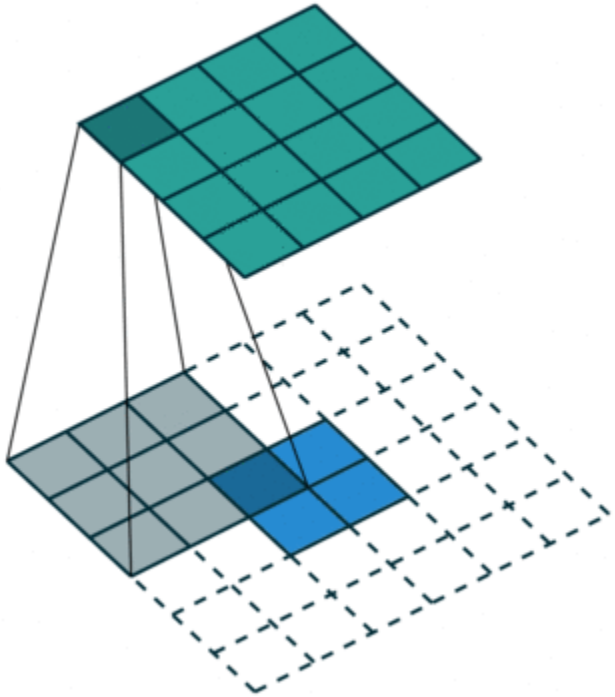
padding = ‘SAME’
strides = [1, 1, 1, 1]



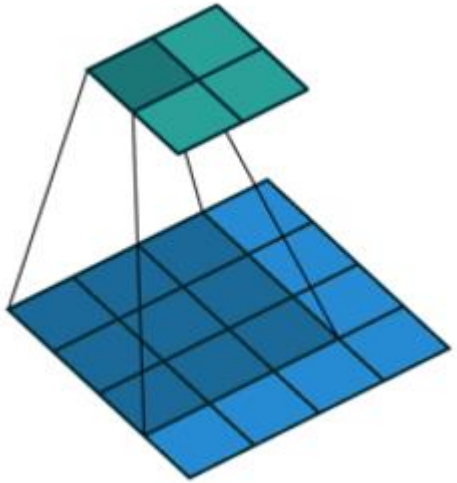
padding = ‘SAME’
strides = [1, 2, 2, 1]

tf.nn.conv2d_transpose

- Decoder
- Stride



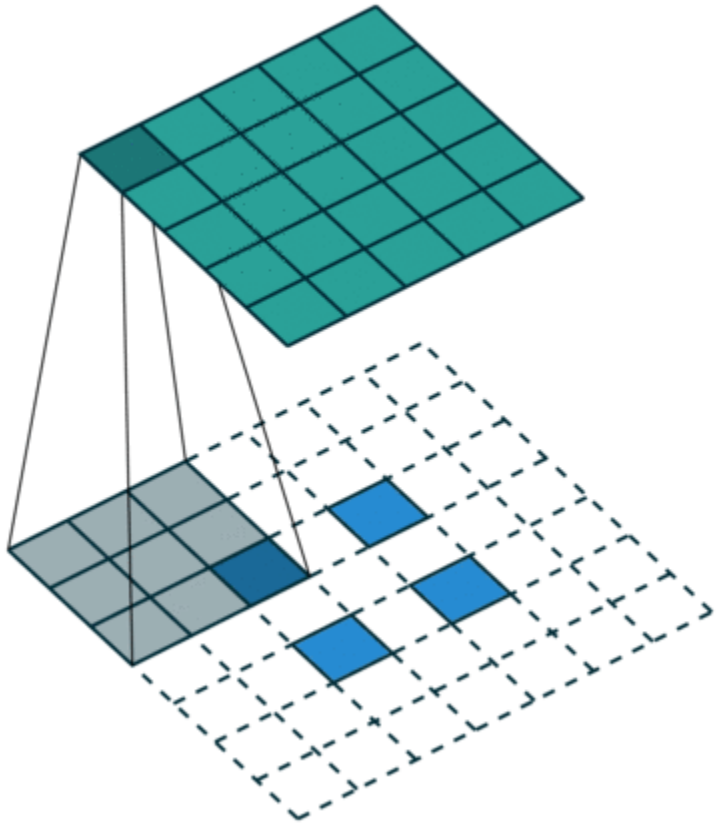
padding = ‘VALID’
strides = (1,1)



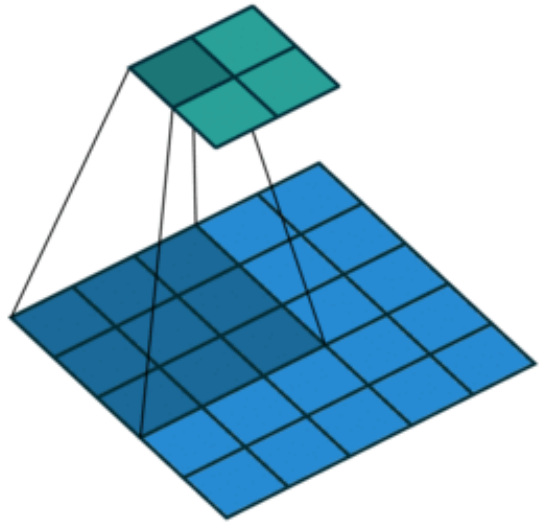
padding = ‘VALID’
strides = (1,1)

tf.nn.conv2d_transpose

- Decoder
- Stride



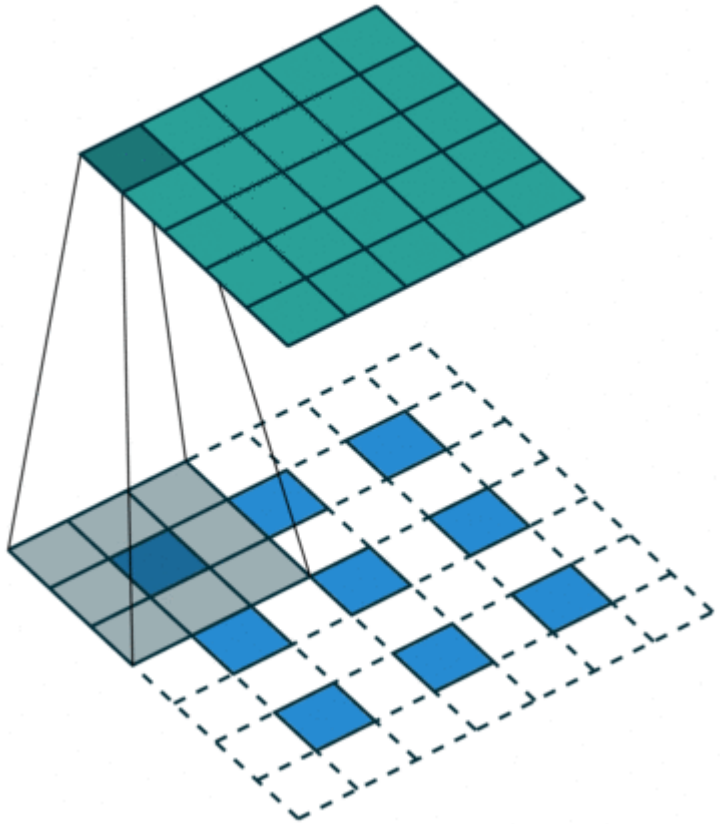
padding = ‘VALID’
strides = (2,2)



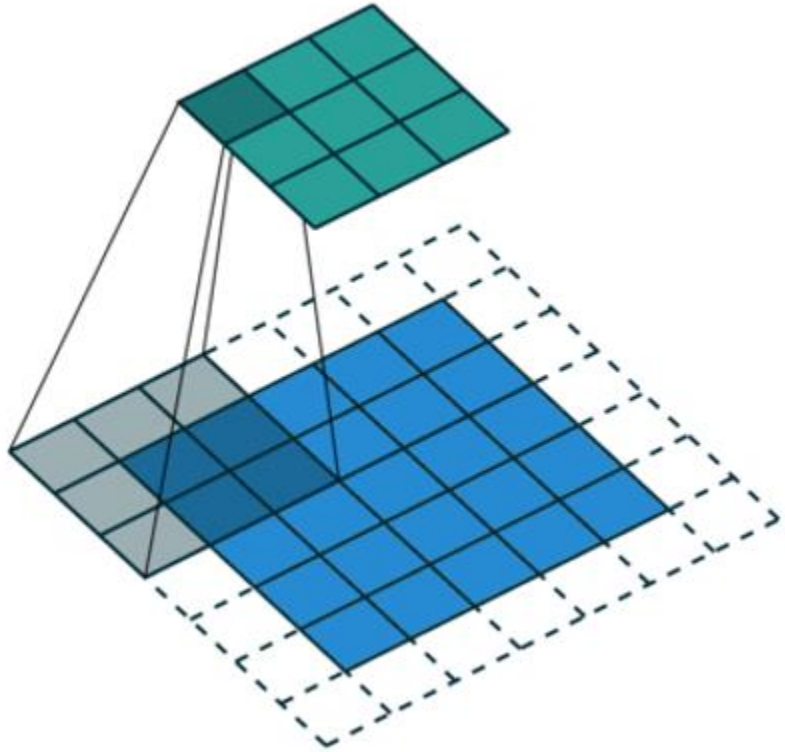
padding = ‘VALID’
strides = (2,2)

tf.nn.conv2d_transpose

- Decoder
- Stride



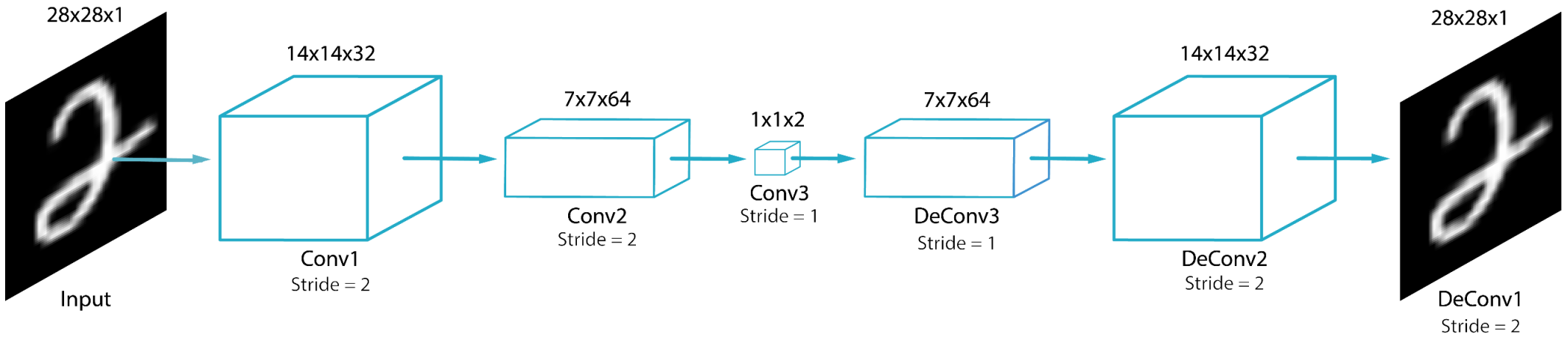
padding = ‘SAME’
strides = (2,2)



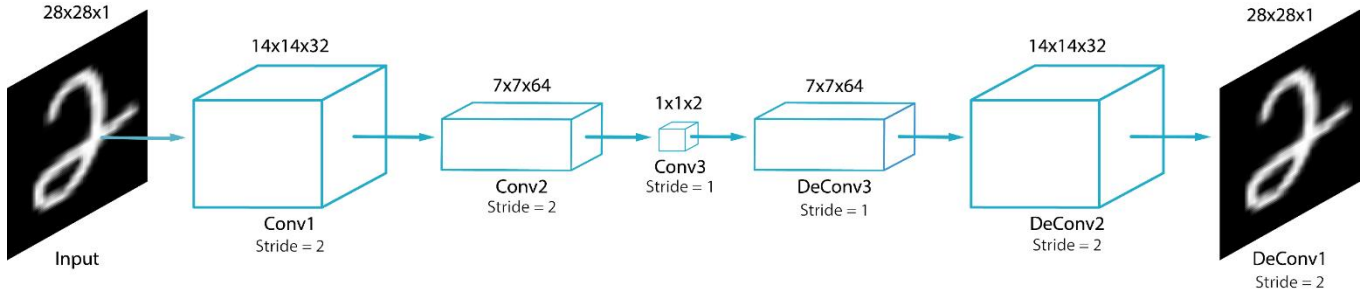
padding = ‘SAME’
strides = (2,2)

CAE Implementation

- Fully convolutional
- Note that no dense layer is used



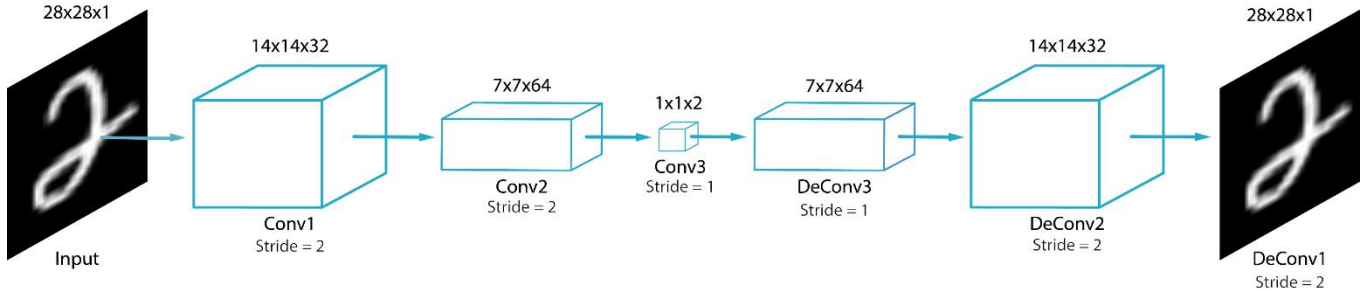
CAE Implementation




```
# Input
input_h = 28
input_w = 28
input_ch = 1
# (None, 28, 28, 1)

## First convolution layer
k1_h = 3
k1_w = 3
k1_ch = 32
p1_h = 2
p1_w = 2
# (None, 14, 14, 32)

## Second convolution layer
k2_h = 3
k2_w = 3
k2_ch = 64
p2_h = 2
p2_w = 2
# (None, 7, 7, 64)

k3_h = 7
k3_w = 7
k3_ch = 2
# (None, 1, 1, 2)
```

```
dk3_h = 7
dk3_w = 7
dk3_ch = 64
# (None, 7, 7, 64)

## Deconvolution layer
dk2_h = 3
dk2_w = 3
dk2_ch = 32
s2_h = 2
s2_w = 2
# (None, 14, 14, 32)

## Deconvolution layer
dk1_h = 3
dk1_w = 3
dk1_ch = 1
s1_h = 2
s1_w = 2
# (None, 28, 28, 1)

## Output
output_h = 28
output_w = 28
output_ch = 1
```

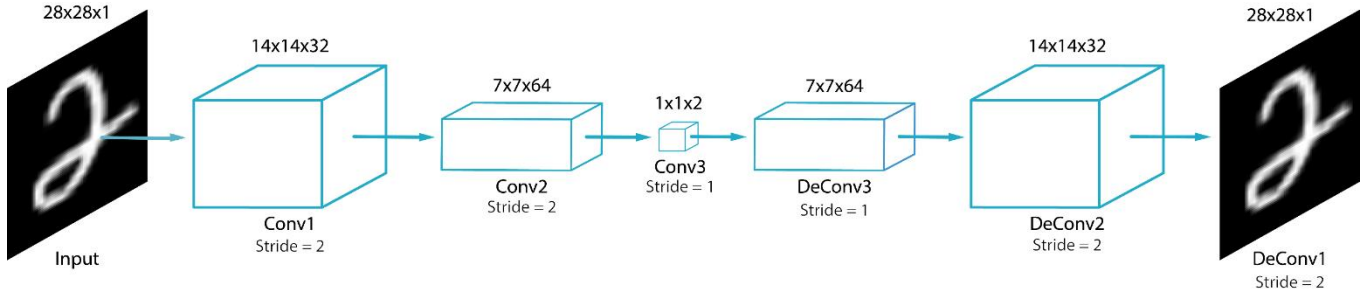
CAE Implementation



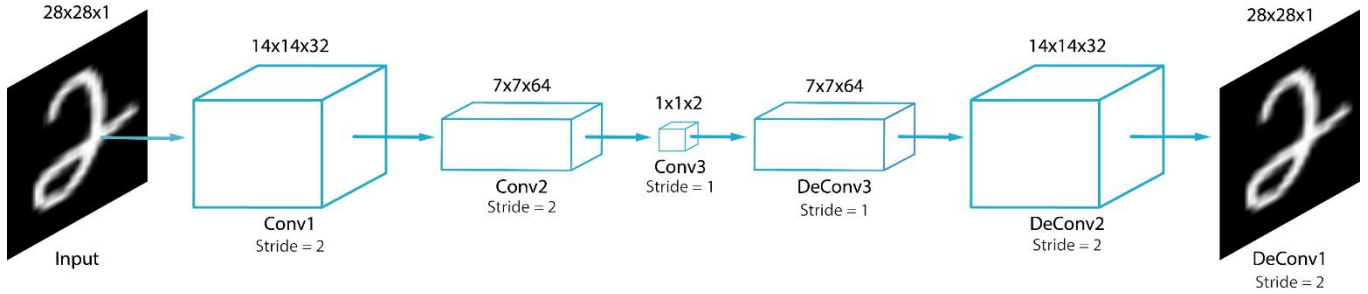

```
weights = {
    'conv1' : tf.Variable(tf.random_normal([k1_h, k1_w, input_ch, k1_ch], stddev = 0.1)),
    'conv2' : tf.Variable(tf.random_normal([k2_h, k2_w, k1_ch, k2_ch], stddev = 0.1)),
    'conv3' : tf.Variable(tf.random_normal([k3_h, k3_w, k2_ch, k3_ch], stddev = 0.1)),
    'deconv3' : tf.Variable(tf.random_normal([dk3_h, dk3_w, dk3_ch, k3_ch], stddev = 0.1)),
    'deconv2' : tf.Variable(tf.random_normal([dk2_h, dk2_w, dk2_ch, dk3_ch], stddev = 0.1)),
    'deconv1' : tf.Variable(tf.random_normal([dk1_h, dk1_w, dk1_ch, dk2_ch], stddev = 0.1))
}

biases = {
    'conv1' : tf.Variable(tf.random_normal([k1_ch], stddev = 0.1)),
    'conv2' : tf.Variable(tf.random_normal([k2_ch], stddev = 0.1)),
    'conv3' : tf.Variable(tf.random_normal([k3_ch], stddev = 0.1)),
    'deconv3' : tf.Variable(tf.random_normal([dk3_ch], stddev = 0.1)),
    'deconv2' : tf.Variable(tf.random_normal([dk2_ch], stddev = 0.1)),
    'deconv1' : tf.Variable(tf.random_normal([dk1_ch], stddev = 0.1)),
}

x = tf.placeholder(tf.float32, [None, input_h, input_w, input_ch])
y = tf.placeholder(tf.float32, [None, output_h, output_w, output_ch])
```

CAE Implementation

```python
def encoder(x, weights, biases):
    ## First convolution layer
    conv1 = tf.nn.conv2d(x,
                         weights['conv1'],
                         strides = [1, 1, 1, 1],
                         padding = 'SAME')
    conv1 = tf.nn.relu(tf.add(conv1, biases['conv1']))
    maxp1 = tf.nn.max_pool(conv1,
                           ksize = [1, p1_h, p1_w, 1],
                           strides = [1, p1_h, p1_w, 1],
                           padding = 'VALID')

    ## Second convolution layer
    conv2 = tf.nn.conv2d(maxp1,
                         weights['conv2'],
                         strides = [1, 1, 1, 1],
                         padding = 'SAME')
    conv2 = tf.nn.relu(tf.add(conv2, biases['conv2']))
    maxp2 = tf.nn.max_pool(conv2,
                           ksize = [1, p2_h, p2_w, 1],
                           strides = [1, p2_h, p2_w, 1],
                           padding = 'VALID')

    conv3 = tf.nn.conv2d(maxp2,
                         weights['conv3'],
                         strides = [1, 1, 1, 1],
                         padding = 'VALID')
    conv3 = tf.add(conv3, biases['conv3'])

    return conv3
```



CAE Implementation

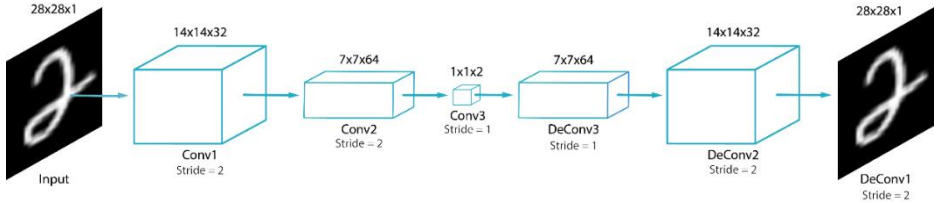



```
def decoder(latent, weights, biases):
    deconv3 = tf.nn.conv2d_transpose(latent,
                                     weights['deconv3'],
                                     output_shape = [tf.shape(latent)[0], 7, 7, 64],
                                     strides = [1, 1, 1, 1],
                                     padding = 'VALID')
    deconv3 = tf.nn.relu(tf.add(deconv3, biases['deconv3']))

    ## First deconvolution layer
    deconv2 = tf.nn.conv2d_transpose(deconv3,
                                     weights['deconv2'],
                                     output_shape = [tf.shape(deconv3)[0], 14, 14, 32],
                                     strides = [1, s2_h, s2_w, 1],
                                     padding = 'SAME')
    deconv2 = tf.nn.relu(tf.add(deconv2, biases['deconv2']))

    ## Second deconvolution layer
    deconv1 = tf.nn.conv2d_transpose(deconv2,
                                     weights['deconv1'],
                                     output_shape = [tf.shape(deconv2)[0], 28, 28, 1],
                                     strides = [1, s1_h, s1_w, 1],
                                     padding = 'SAME')
    deconv1 = tf.add(deconv1, biases['deconv1'])

    return deconv1
```

CAE Implementation with tf.layers

```python
def encoder(x):
    ## First convolution layer
    conv1 = tf.layers.conv2d(inputs = x,
                             filters = 32,
                             kernel_size = [3, 3],
                             strides = [1, 1],
                             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")

    return conv3
```

```python
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 = [3, 3],
                                         strides = (2, 2),
                                         padding = 'SAME',
                                         activation = tf.nn.relu)

    ## Second deconvolution layer
    deconv1 = tf.layers.conv2d_transpose(inputs = deconv2,
                                         filters = 1,
                                         kernel_size = [3, 3],
                                         strides = (2, 2),
                                         padding = 'SAME')

    return deconv1
```



Reconstruction Result