Conditional VAE

One of the drawbacks of the plain Autoencoder and VAE when used to generate synthetic examples

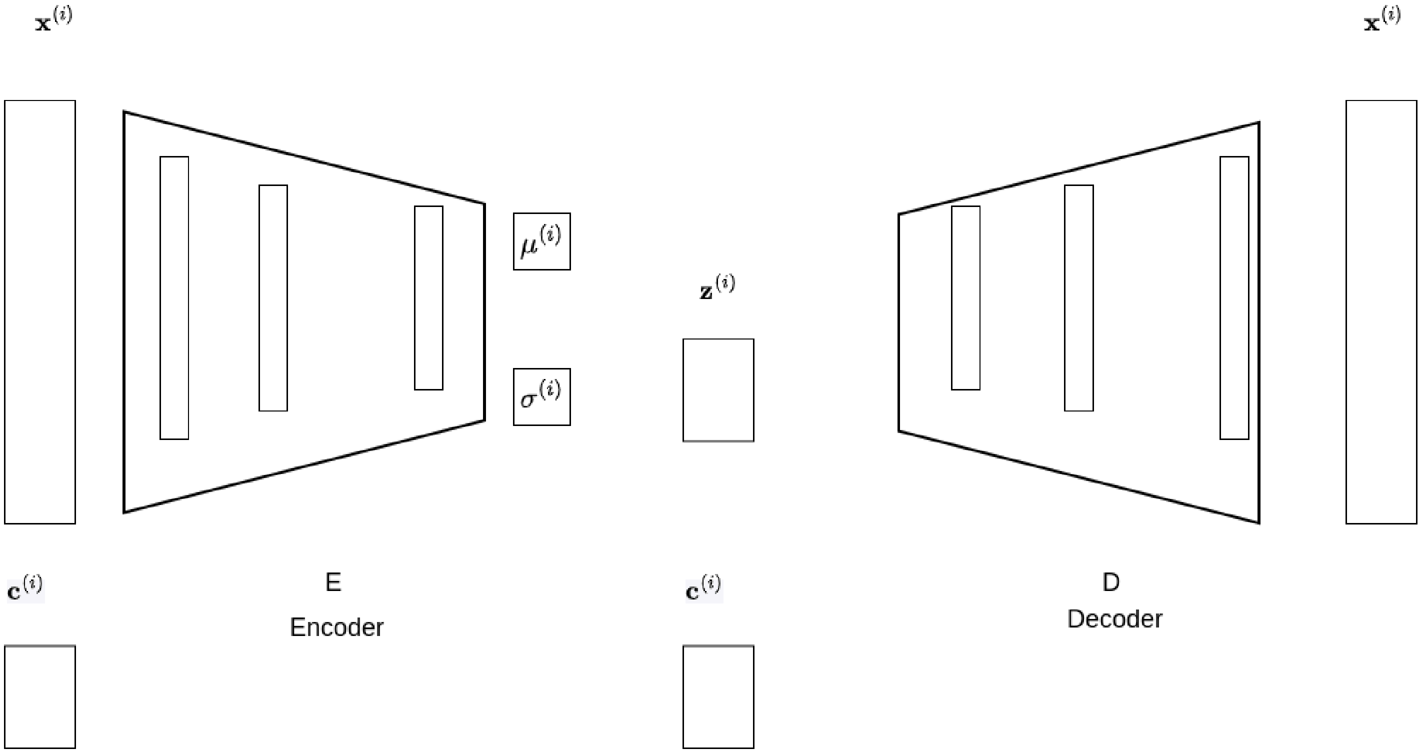
- Can't specify the class of the synthetic feature vectors produced

The Conditional VAE addresses this problem.

The Encoder and Decoder components of the Conditional VAE

- Each take a second input: the class label of the example
- The Encoder merely passes this class label directly to its output
    - where, in training, it is used as the class label input of the Decoder
- The Encoder is otherwise identical to the one in the "unconditional" VAE
- To generate a synthetic feature vector
    - pass the class label desired along with a latent to the Decoder

Conditional VAE: Components



Note that the class label does not affect the Encoder at all

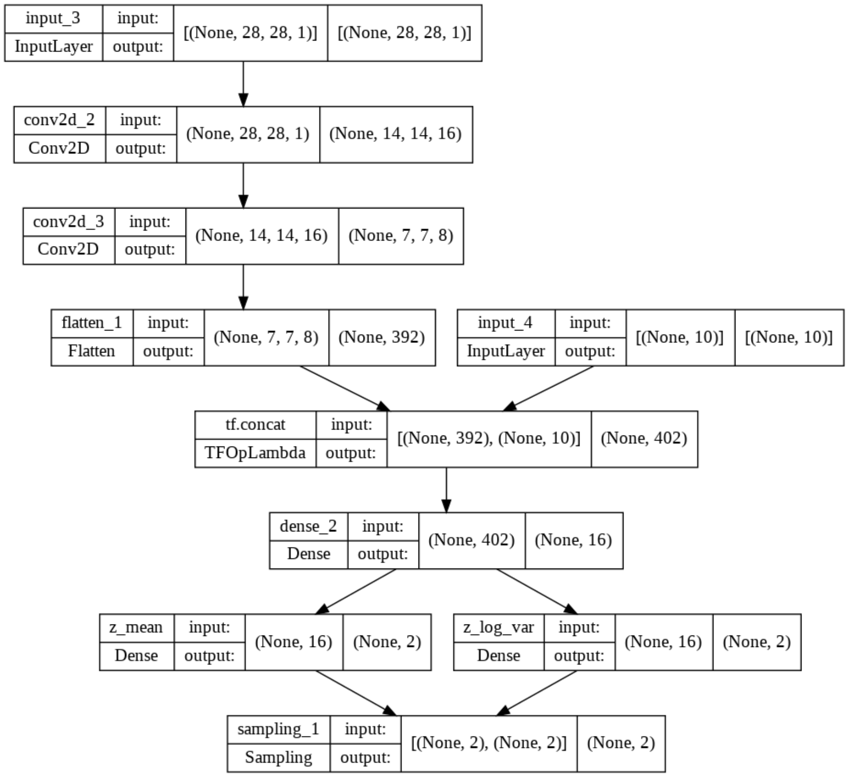
- latent representation does not depend on it

For the Decoder

- there is no pre-determined manner in which the class label needs to influence the architecture
    - we choose to concatenate the class label (as a OHE vector) to the output of an intermediate layer of the "non-conditional" Decoder
- it merely conditions the Decoder on both the class label and the latent
$$\tilde{\mathbf{x}} \in p_\Theta(\mathbf{x} \mid \mathbf{z}, \mathbf{c})$$
    - rather than just the latent

Code

Conditional VAE: Components



input_3 | InputLayer | input: [(None, 28, 28, 1)] | output: [(None, 28, 28, 1)]

conv2d_2 | Conv2D | input: (None, 28, 28, 1) | output: (None, 14, 14, 16)

conv2d_3 | Conv2D | input: (None, 14, 14, 16) | output: (None, 7, 7, 8)

flatten_1 | Flatten | input: (None, 7, 7, 8) | output: (None, 392)

input_4 | InputLayer | input: [(None, 10)] | output: [(None, 10)]

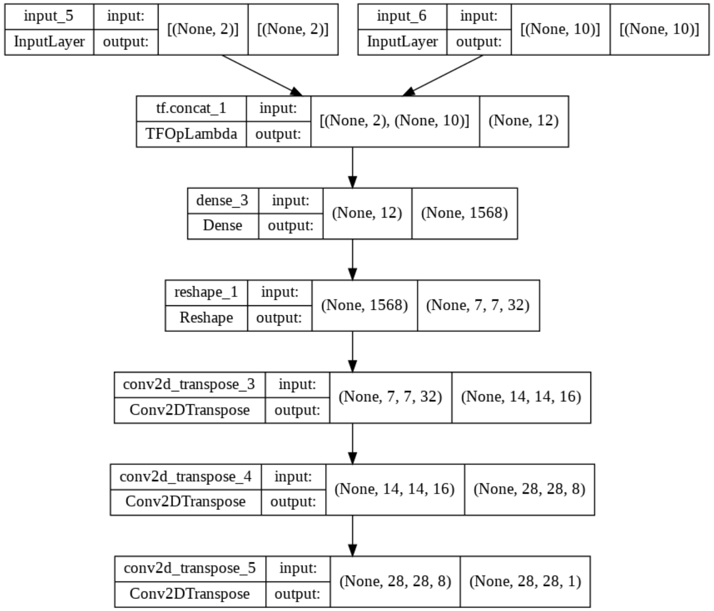
tf.concat | TFOpLambda | input: [(None, 392), (None, 10)] | output: (None, 402)

dense_2 | Dense | input: (None, 402) | output: (None, 16)

z_mean | Dense | input: (None, 16) | output: (None, 2)

z_log_var | Dense | input: (None, 16) | output: (None, 2)

sampling_1 | Sampling | input: [(None, 2), (None, 2)] | output: (None, 2)



input_5 | InputLayer | input: [(None, 2)] | output: [(None, 2)]

input_6 | InputLayer | input: [(None, 10)] | output: [(None, 10)]

tf.concat_1 | TFOpLambda | input: [(None, 2), (None, 10)] | output: (None, 12)

dense_3 | Dense | input: (None, 12) | output: (None, 1568)

reshape_1 | Reshape | input: (None, 1568) | output: (None, 7, 7, 32)

conv2d_transpose_3 | Conv2DTranspose | input: (None, 7, 7, 32) | output: (None, 14, 14, 16)

conv2d_transpose_4 | Conv2DTranspose | input: (None, 14, 14, 16) | output: (None, 28, 28, 8)

conv2d_transpose_5 | Conv2DTranspose | input: (None, 28, 28, 8) | output: (None, 28, 28, 1)

Observe in the code diagram above

- both the Encoder and Decoder have 2 `InputLayers`
    - one for the class label

The class label arguments

- are implemented as One Hot Encoded vectors
    - for both the Encoder and Decoder
    - look for the `InputLayer` of shape `(None, 10)`
        - this is the OHE of 10 possible classes (digits 0 through 9)

Synthetic MNIST examples from a Conditional VAE: vary the 2 components of a 2D latent $\mathbf{z}$; fixed label

In [ ]:

```
print("Done")
```