

Conditional VAE

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

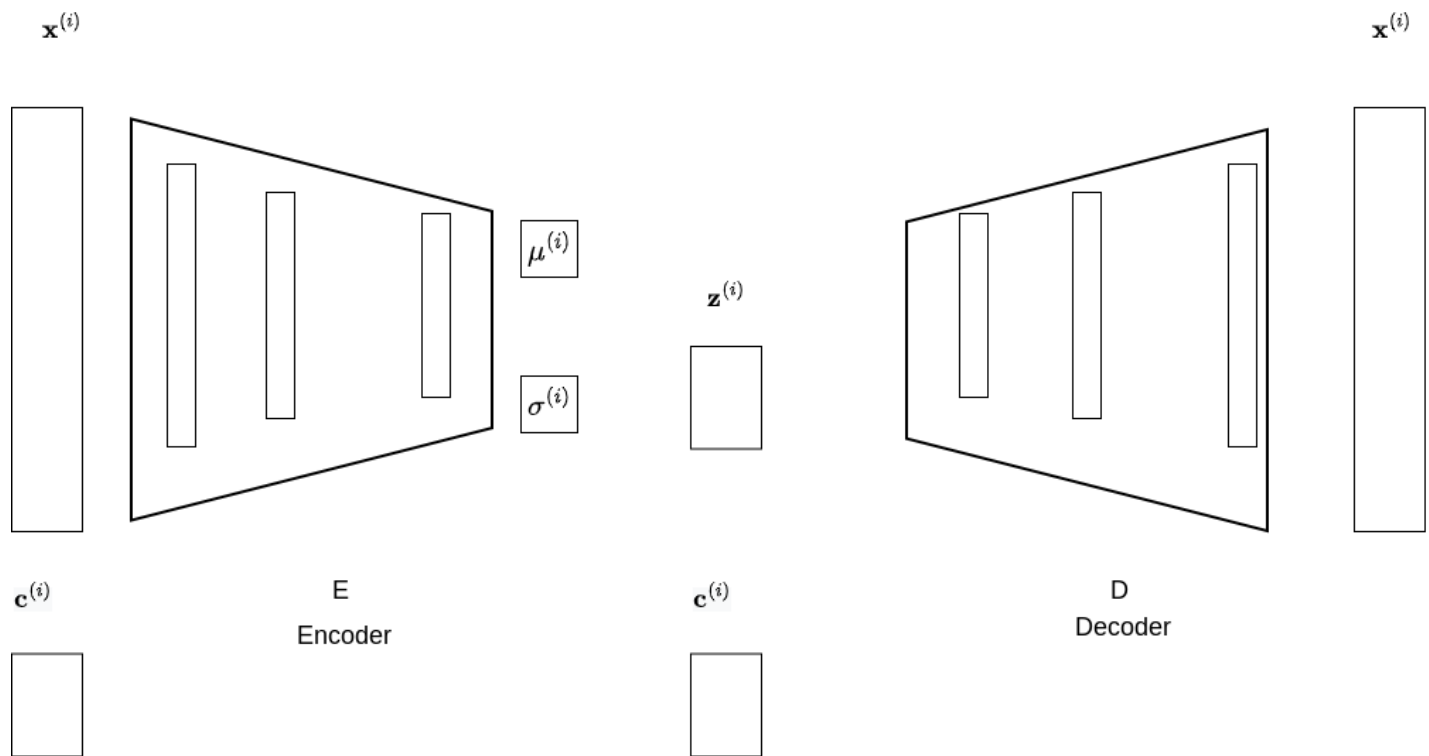
- Can't specify the class of the synthetic feature vectors produced

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

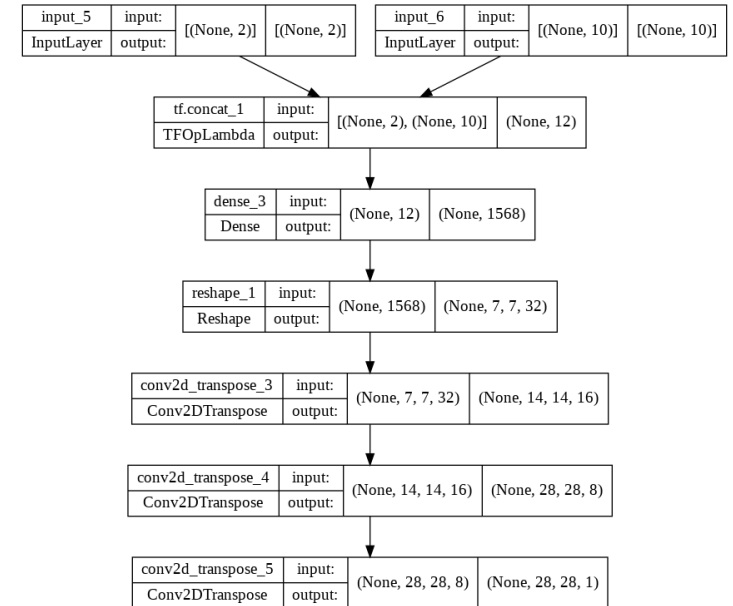
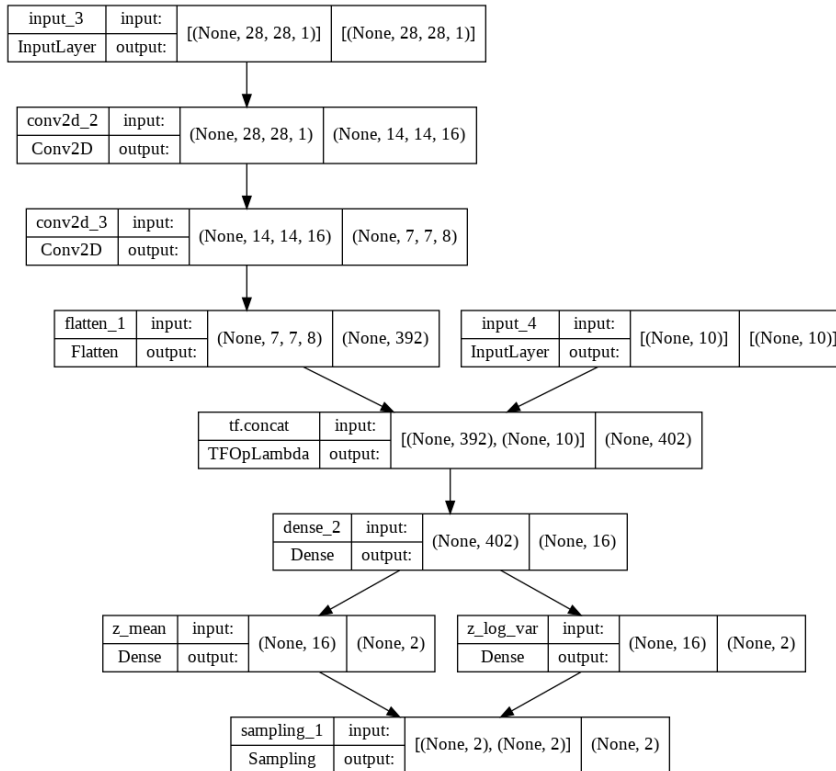
- there is no pre-determined manner in which the class label needs to influence the architecture
 - we choose to concatenate the class label (as a OHE vector) to the output of an intermediate layer of the "non-conditional" Decoder
- it merely conditions the Decoder on both the class label and the latent

$$\tilde{\mathbf{x}} \in p_{\Theta}(\mathbf{x} \mid \mathbf{z}, \mathbf{c})$$

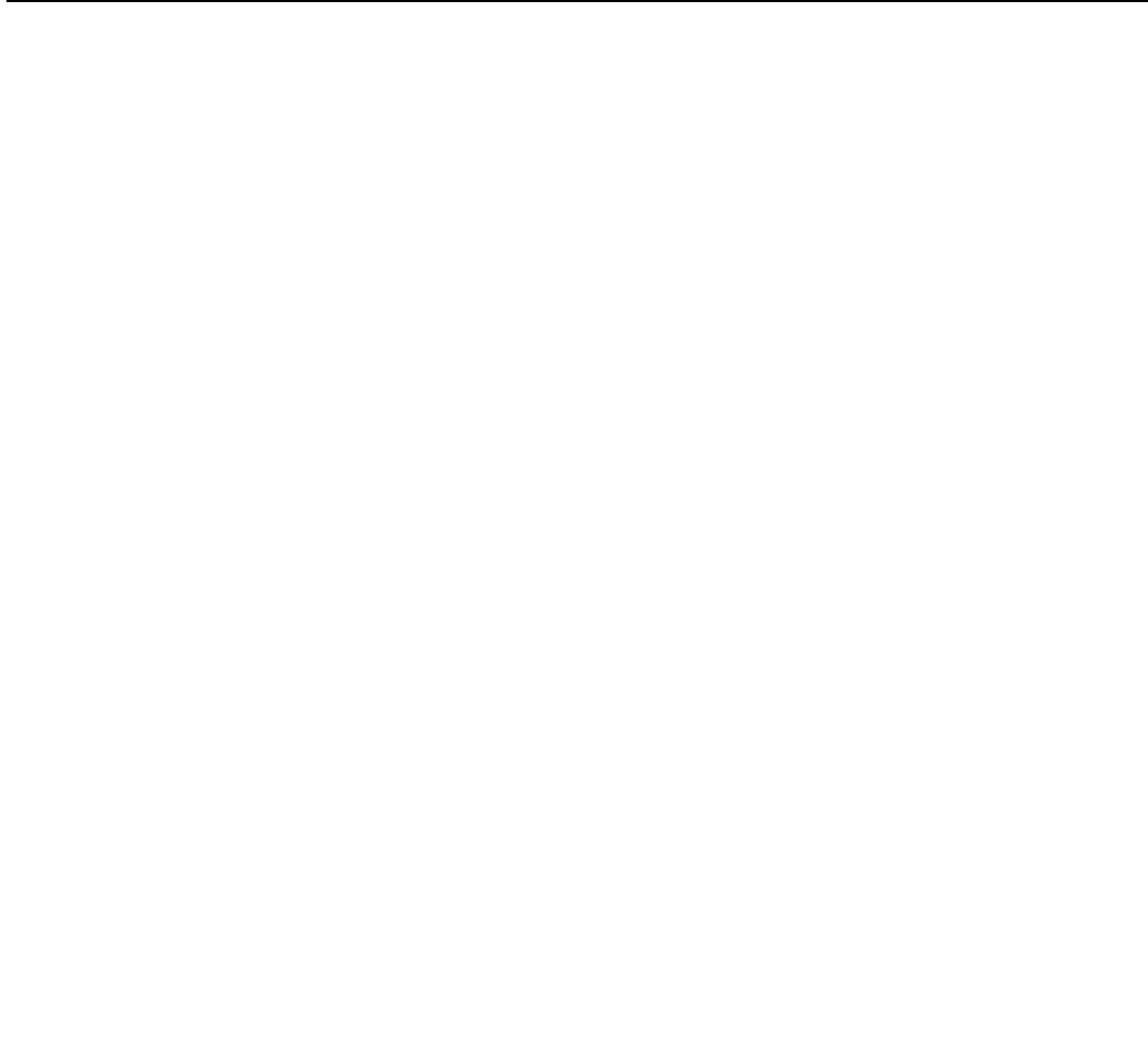
- rather than just the latent

Code

Conditional VAE: Components



Synthetic MNIST examples from a Conditional VAE: vary the 2 components of a 2D latent z ; fixed label



```
In [ ]: print("Done")
```