Variational autoencoders

Reagan Berhe | CSC 495 | Fall 2020

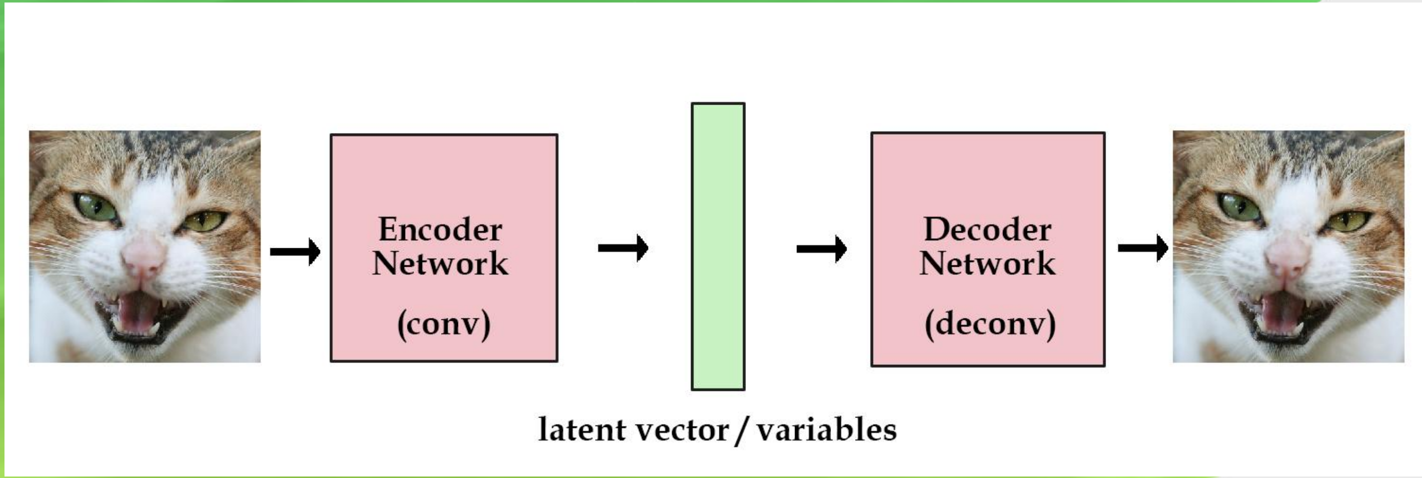
Autoencoder

- Unsupervised Learning
- Designed in neutral networks architecture
- Take data as input and discover some latent state representation of that data.

Autoencoder

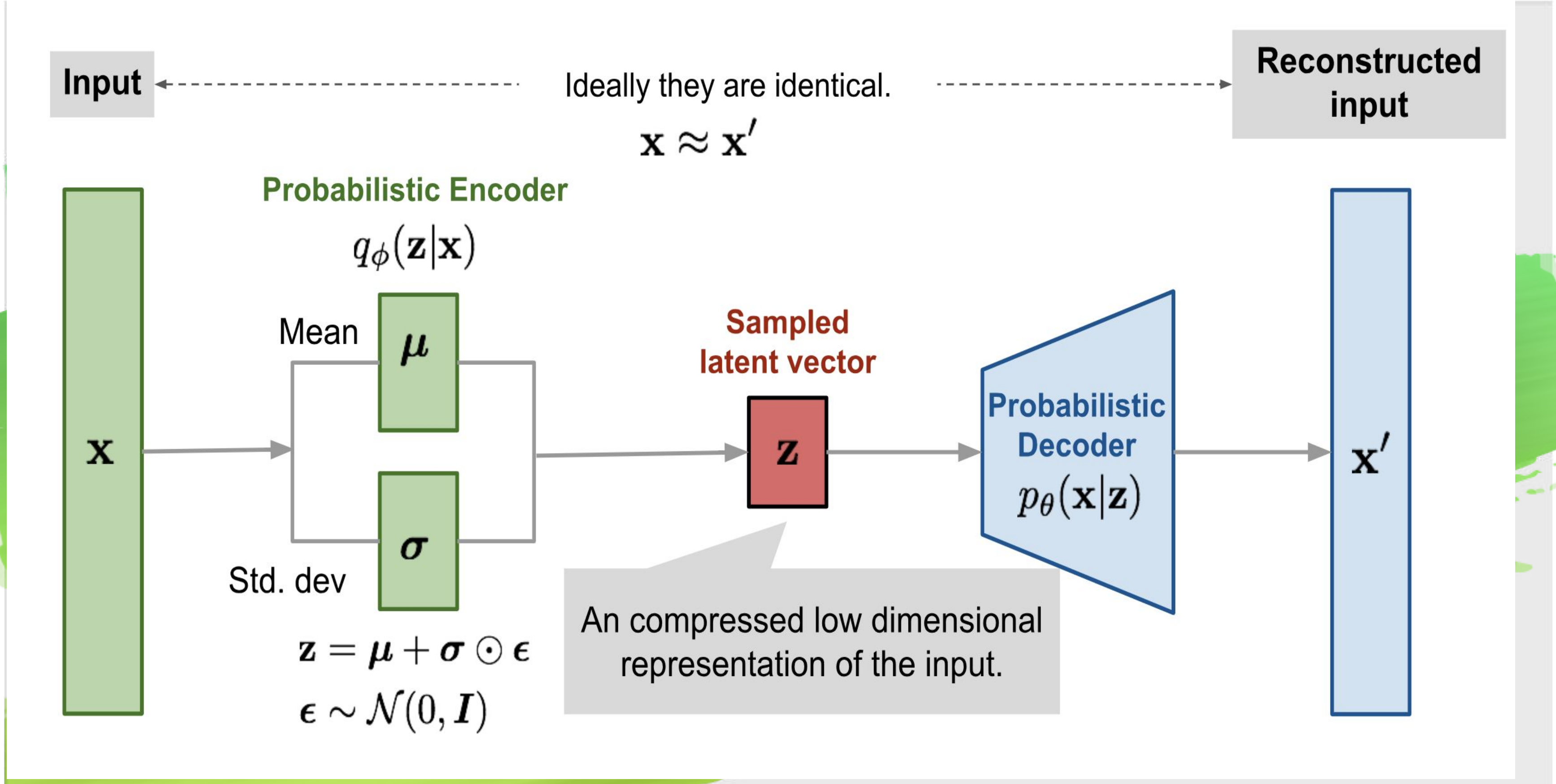
- Input data is converted into an encoding vector where each dimension represents some learned attribute about the data.
- Takes in the original images and encodes them into vectors.

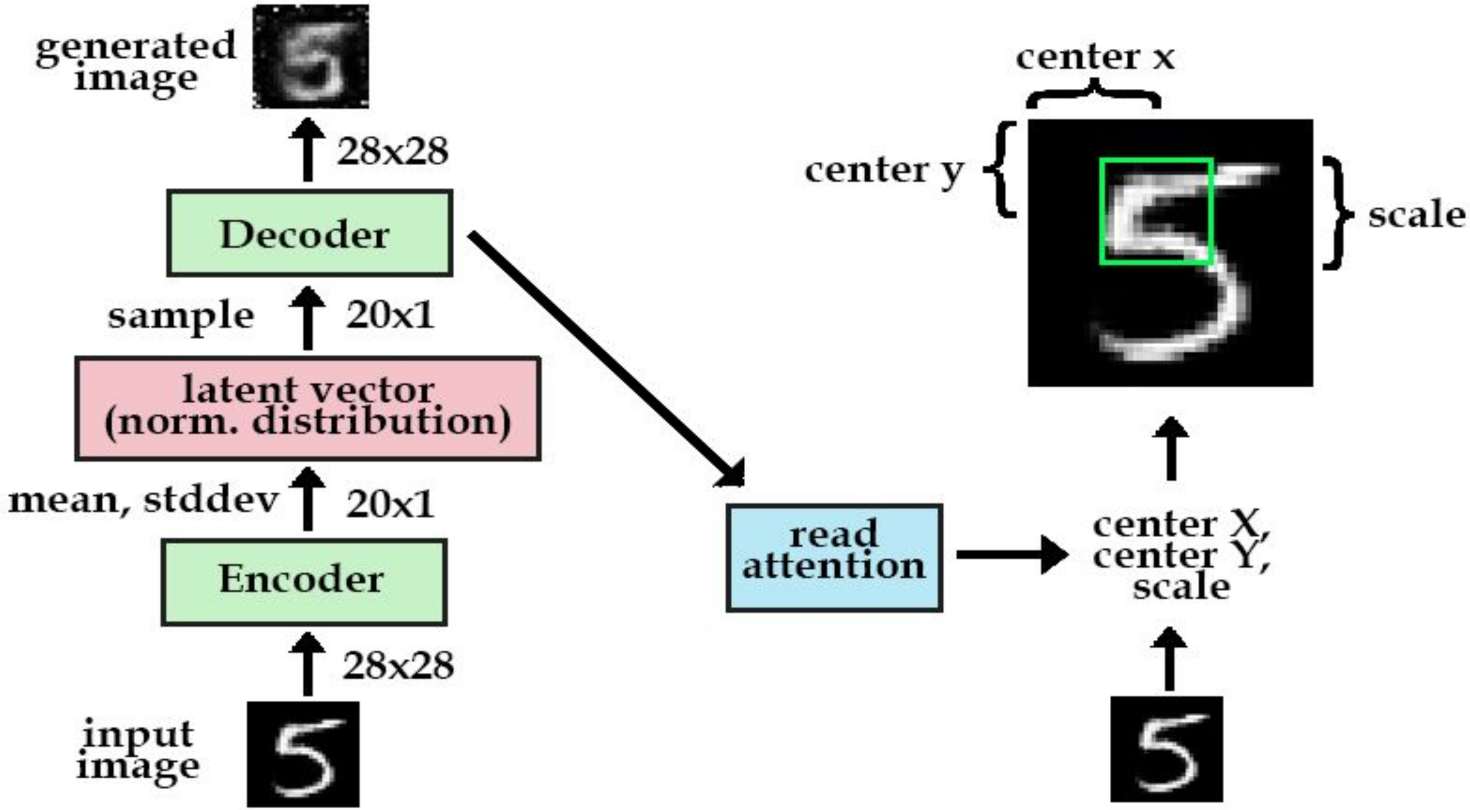
Autoencoder



What is a variational autoencoder?

- VAE provides a probabilistic manner for describing an observation in latent space.
- Our encoder network is outputting a single value for each encoding dimension.


Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{x}$
Mean
$\boldsymbol{\mu}$
$\boldsymbol{\sigma}$
Std. dev
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$


generated image
28x28
Decoder
sample
20x1
latent vector (norm. distribution)
mean, stddev
20x1
Encoder
28x28
input image
center x
center y
scale
read attention
center X, center Y, scale

Implementation Goal

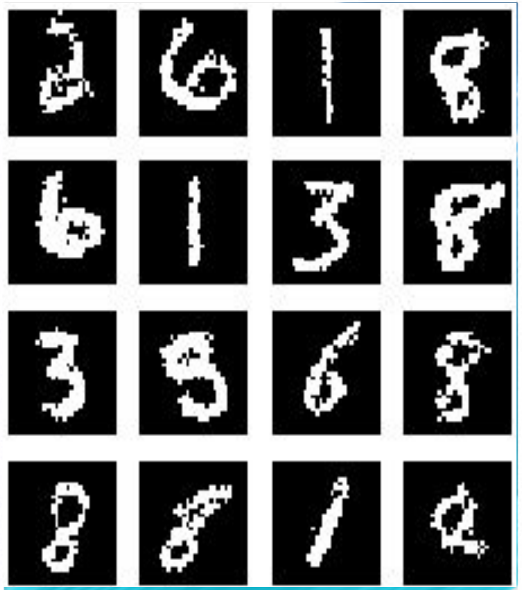
(a) Reconstruction of the input digit images as close as possible

(b) Generation of new digit images that look realistic, using random samples from the prior distribution (rather than samples from the posterior, conditional on data) as the input to the decoder

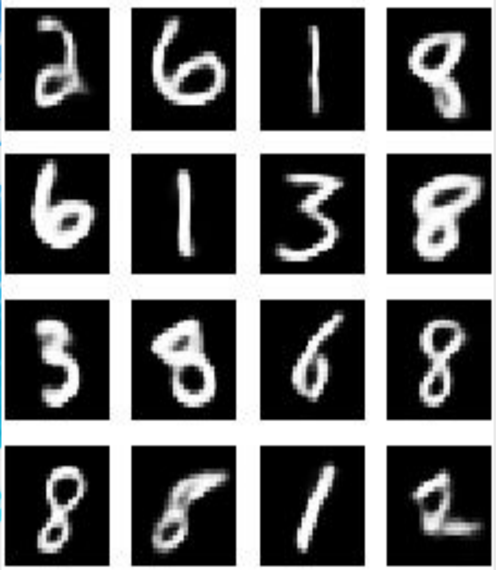
Implementation Goal

- This module applies:
  - MNIST dataset with handwritten digits
  - Images of shape (28,28,1)
  - Normalizing the dataset to be between 0 and 1, discretized the values to be either 0 or 1, using 0.5 as threshold

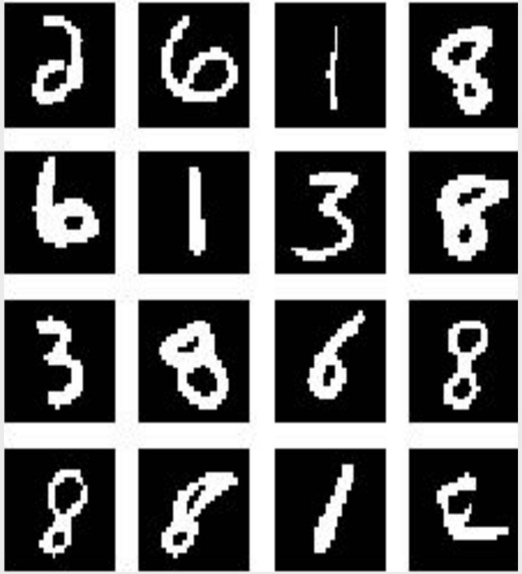
```
def preprocess_images(images):
  images = images.reshape((images.shape[0], 28, 28, 1)) / 255.
  return np.where(images > .5, 1.0, 0.0).astype('float32')

train_images = preprocess_images(train_images)
test_images = preprocess_images(test_images)
```



one sample from decoder distribution

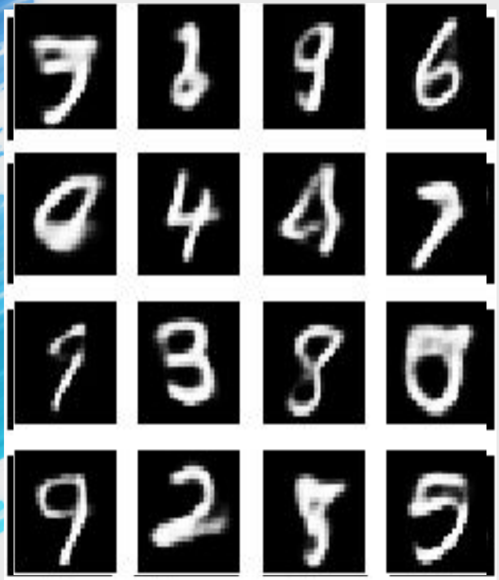


mean of decoder from distribution

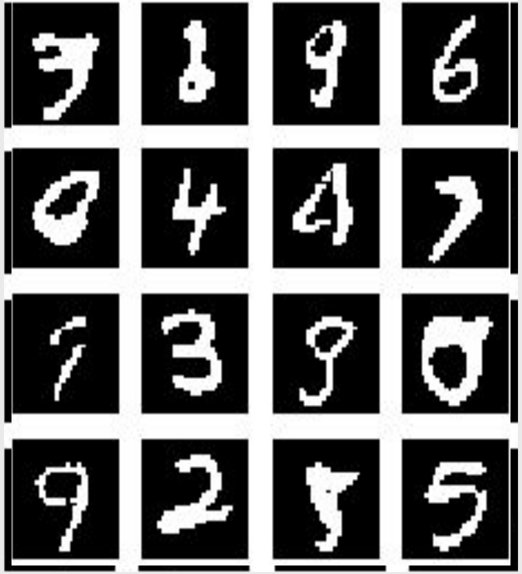


mode of decoder from distribution

one sample from
decoder distribution



mean of decoder
distribution



mode of decoder
distribution

Focus on Implementing VAE

- By combining TensorFlow Probability (TFP) with Keras API of TensorFlow 2.0 (TF2)
- In this version, the output of both the encoder and the decoder are objects from

```
8 import tensorflow_probability as tfp
```

Focus on Implementing VAE

- For the original VAE, the decoder output is deterministic, therefore after sampling z, the decoder output is set.
- However, with the output being a distribution, we can call `mean`, `mode`, or `sample` method to output a `tf.Tensor`

Observation

- For the original VAE, the decoder output is deterministic, therefore after sampling z, the decoder output is set.
- However, with the output being a distribution, we can call `mean`, `mode`, or `sample` method to output a `tf.Tensor`

Observation

- Since the mean of a Bernoulli distribution is a value between 0 and 1, while the mode is either 0 (when parameter is less than 0.5) or 1 (otherwise)

Reference

[1] Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes (2013), Proceedings of the 2nd International Conference on Learning Representations (ICLR)

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu and Daan Wierstra, Weight Uncertainty in Neural Networks (2015), Proceedings of the 32nd International Conference on Machine Learning (ICML)

[3] Making new Layers & Models via subclassing (2020), TensorFlow Guide

[4] Convolutional Variational Autoencoder (2020), TensorFlow Tutorial

[5] Ian Fischer, Alex Alemi, Joshua V. Dillon, and the TFP Team, Variational Autoencoders with Tensorflow Probability Layers (2019), TensorFlow on Medium

Questions?