Copulas as High-Dimensional Generative Models: Vine Copula Autoencoders

Natasa Tagasovska, Damien Ackerer and Thibault Vatter.

NeurIPS, 2019.

Ting 2020.12.29

Brief introduction of VCAE

Vine-copula-Autoencoder

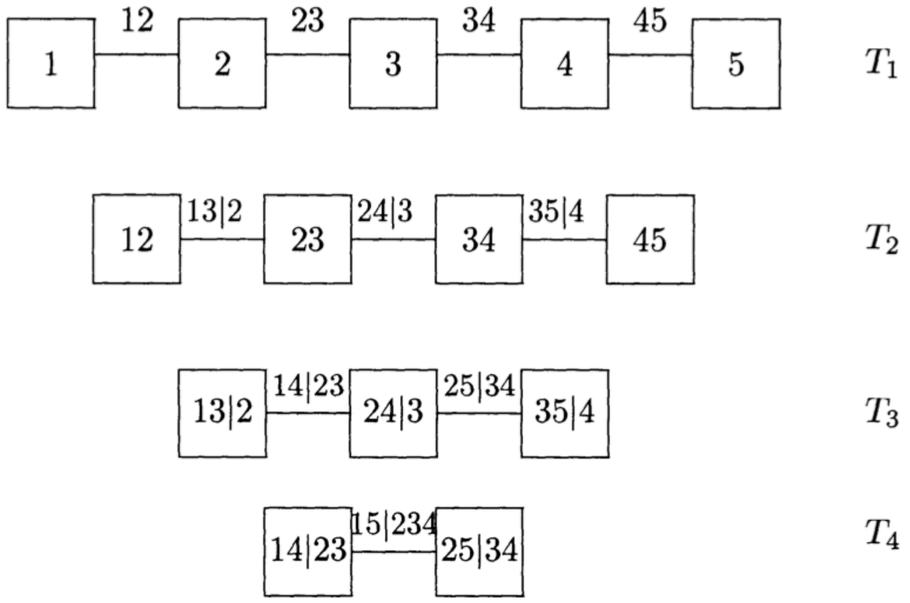


Figure 1. A five-dimensional D-vine, with T_j denoting the jth tree for $j = 1, 2, 3, 4$.

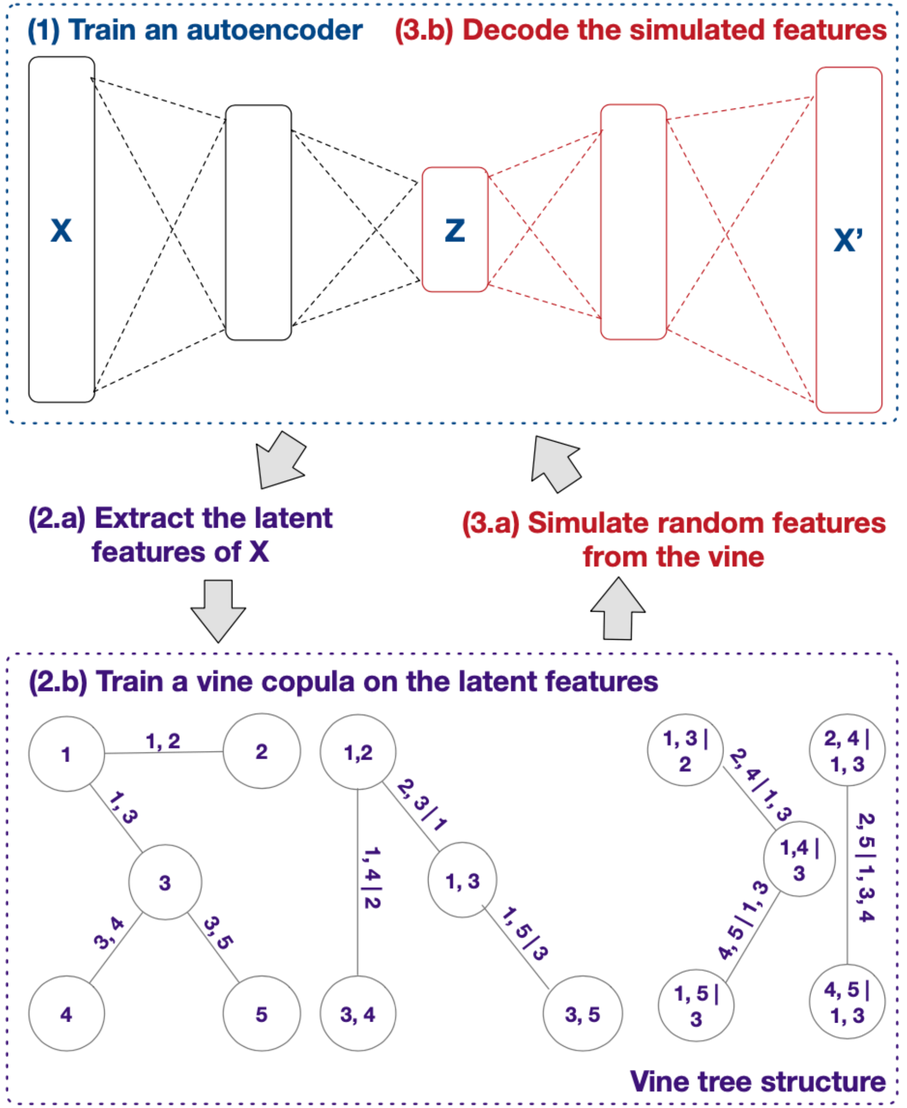


Figure 1: Conceptual illustration of a VCAE.

VCAE algorithm

Algorithm 1 Vine Copula Autoencoder

Input: train set X of $\{x_1, x_2, ...x_n\}$ images.
1. Train autoencoder component with X:
$f \leftarrow encoder$
$g \leftarrow decoder$

2. Encode train set with f :
$\phi(X) \leftarrow f(X)$

3. Fit a vine copula c using encoded features:
$c \leftarrow \{\phi_1, \phi_2, ...\phi_n\}$ (as described in Sec 2.2 and 2.3).

4. Sample random observations form c:
$\phi' \leftarrow c(\phi)$ (as in Sec 2.4)

5. Decode the random features:
$X' \leftarrow g(\phi')$

Output: generated images X'.

- n=5: truncated at 5th tree
- Using nonparametric copula
- Using the ‘Rvinecopulib’ package in R
- Combining R package with Python code by ‘rpy2’ package in python

CNN for Decoder and Encoder

- Encoder:

$$\begin{aligned} x \in R^{32\times 32} &\to Conv_{32} \to BN \to ReLU \\ &\to Conv_{64} \to BN \to ReLU \\ &\to Conv_{128} \to BN \to ReLU \\ &\to FC_{10} \end{aligned}$$

- Decoder:

$$\begin{aligned} z \in R^{10} \to FC_{100} &\to ConvT_{128} \to BN \to ReLU \\ &\to ConvT_{64} \to BN \to ReLU \\ &\to ConvT_{128} \to BN \to ReLU \\ &\to FC_{1} \end{aligned}$$

Experiments in paper

MNIST, Street View House Numbers, and one large scale - CelebA.

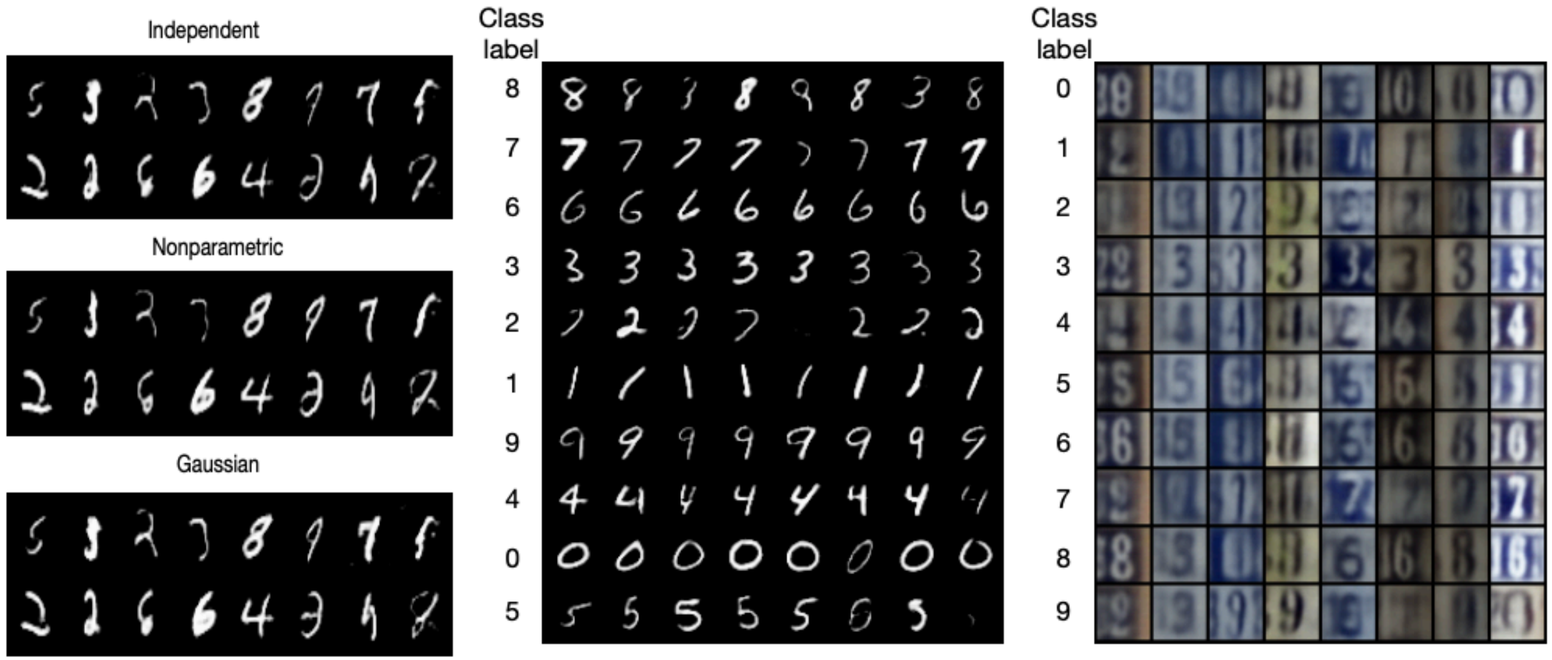


Figure 4: Left - impact of copula family selection in image sampling. Right - Random samples of Conditional *VCAE* on **MNIST** and **SVHN**

VCAE as generative models

Comparing with DCGAN(deep convolutional GAN) and variational autoencoders (VAEs)

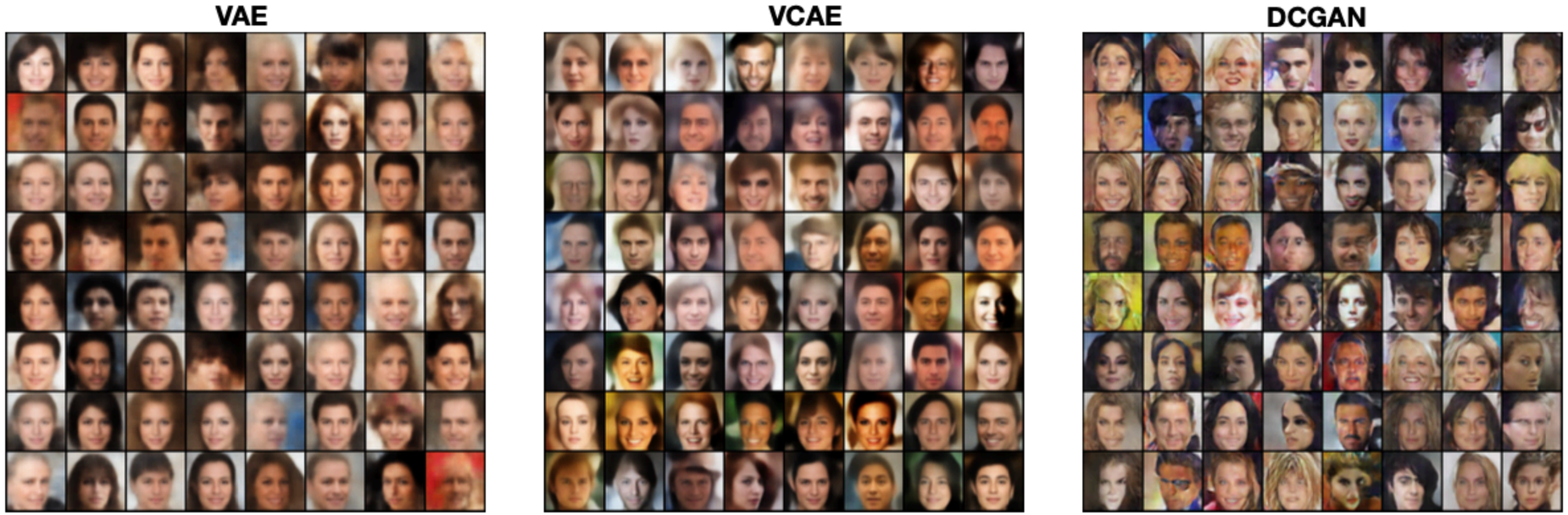


Figure 7: Random samples from left - *VAE*, - *VCAE* on **CelebA**, both trained for 200 epochs. - *DCGAN* best results at 30 epochs.

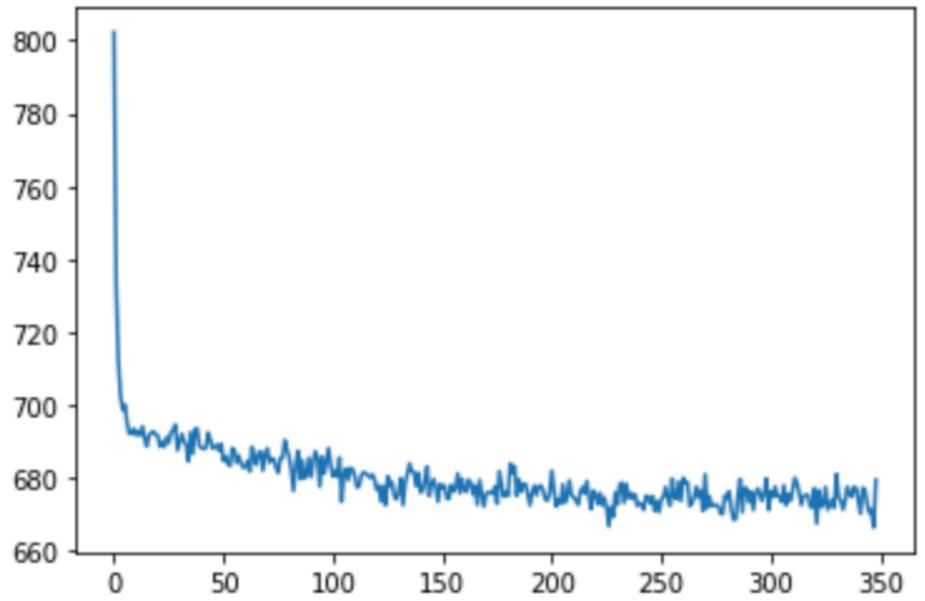
Our Aim: perform VCAE on MedMNIST data

organmnist_axial

- Epoch=4
- Loss figure

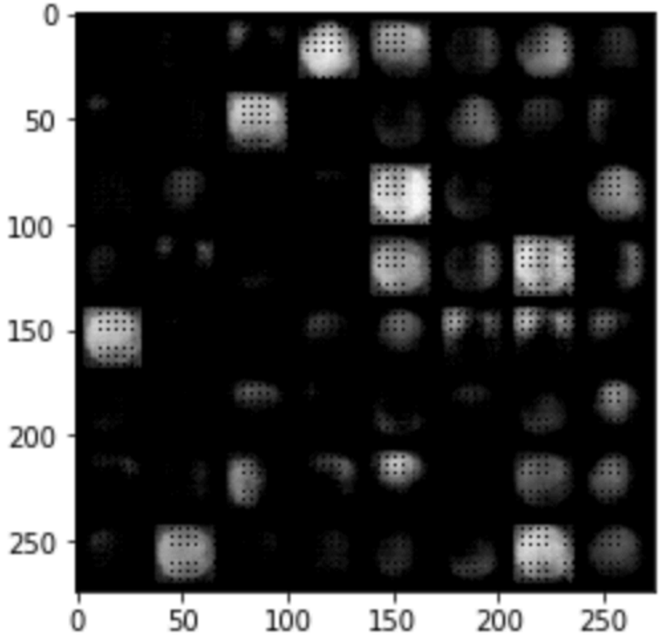
```
In [14]: import matplotlib.pyplot as plt
         plt.plot(lossseq)

Out[14]: [<matplotlib.lines.Line2D at 0x7fece1ce1350>]
```



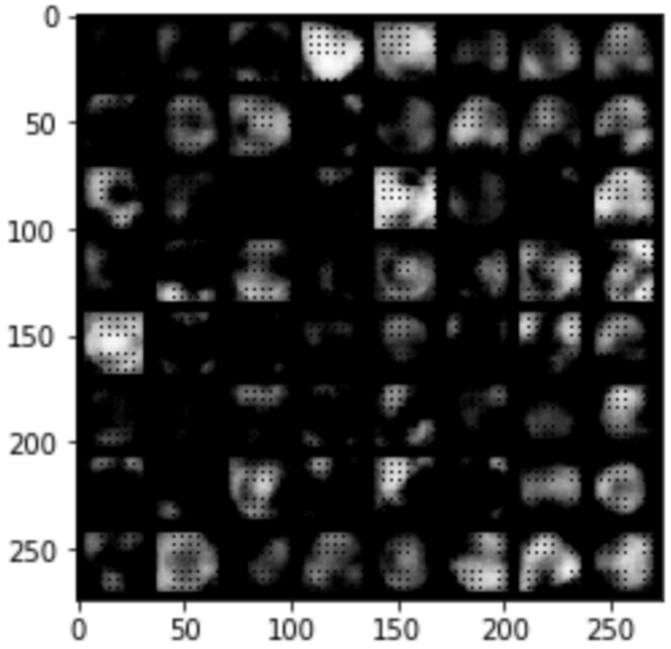
Results

```
<matplotlib.image.AxesImage at 0x7fec7ba06c10>
```



epoch=1

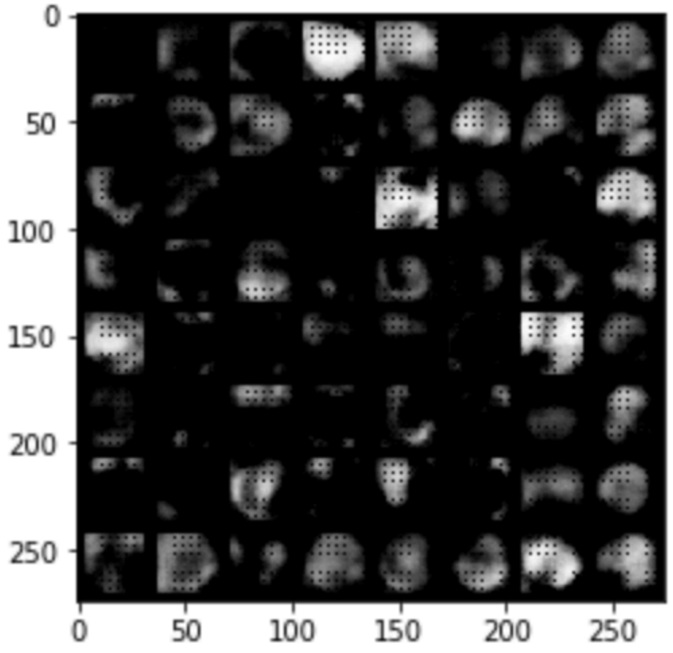
```
<matplotlib.image.AxesImage at 0x7fec7b07ed50>
```



epoch=2

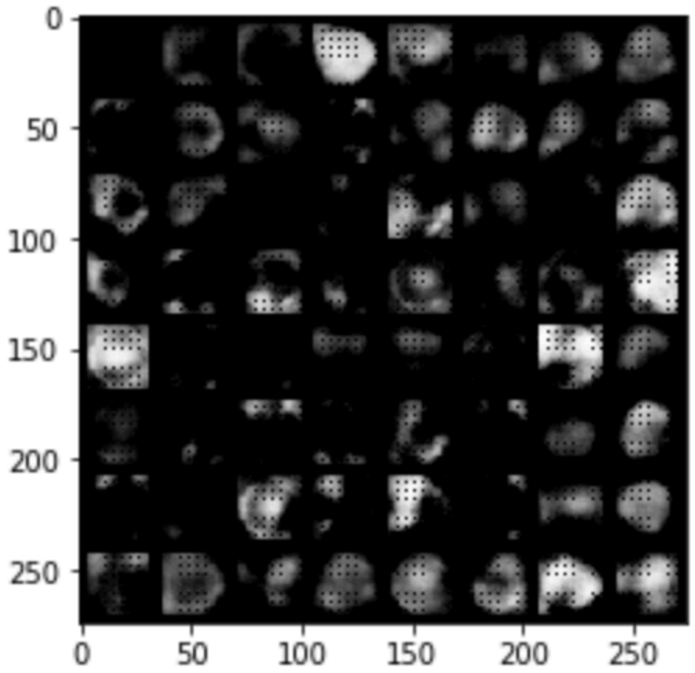
Results

```
<matplotlib.image.AxesImage at 0x7fec7a699e90>
```



epoch=3

```
<matplotlib.image.AxesImage at 0x7fec7ab21f90>
```



epoch=4