**Market Scenario Generator Hackathon: From Stability to Storms**

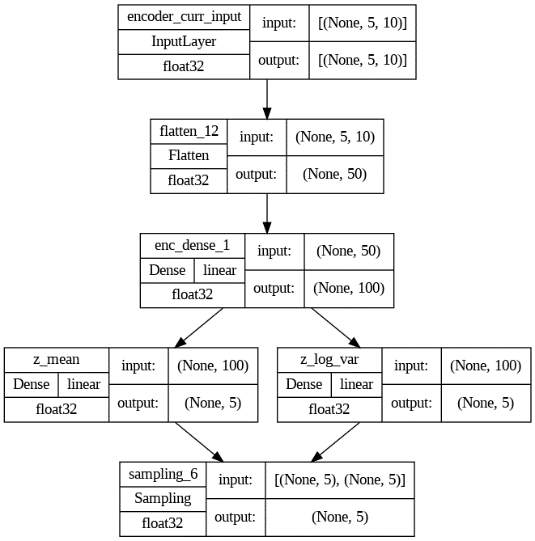
|  |  |
| --- | --- |
| Participant: | tuananhnguyen |
| Number of submissions: | 18 |

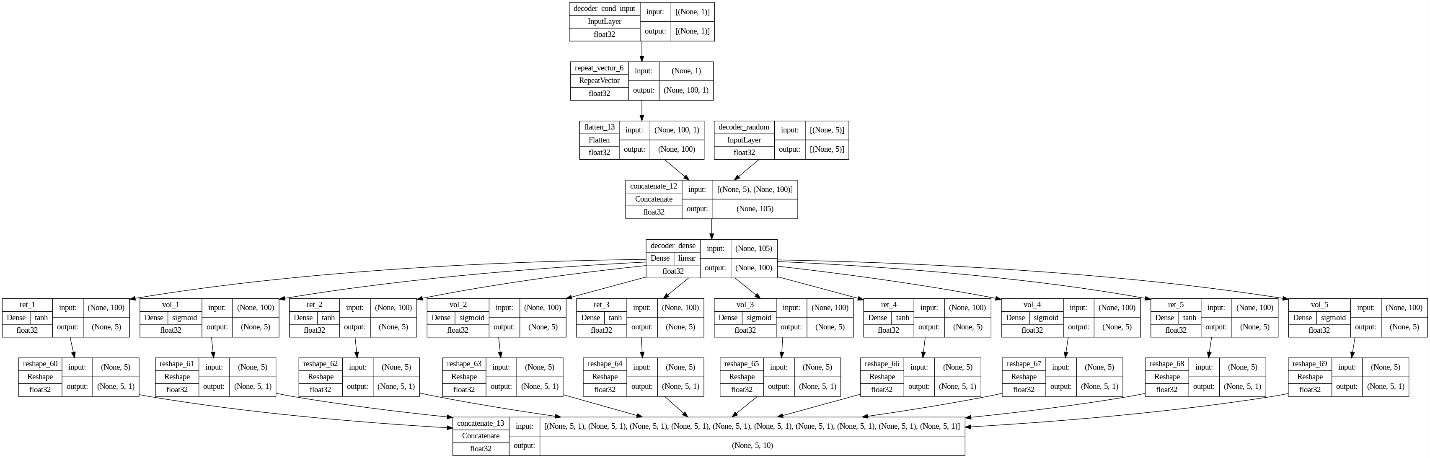
# Overview

This is a report for the submission number 18. The method that I used mostly used Variational AutoEncoder with adjusted output activation functions and loss functions (not TimeVAE though).

# Neural networks

In this section, I will display the summarized structure of my networks (activation functions, layers, …). Like any VAE, my model has an encoder and a decoder with the following structure:





At the end of my decoder, I separate the output since for daily log return, the better activation is ‘tanh’ and as for volatility, softmax or sigmoid are preffered (sigmoid in my case).

# Training step

The details of my training step are actually in the model.py file but I also want to clarify a few things. Usually, a VAE will have 2 loss function: reconstruction loss between generated and true sample (let’s call them tensor), and a KL loss between the mean, variance of the encoder and the standard Gaussian distribution.

In my case, I also have these losses:

* Reconstruction loss 1: 3-part MSE loss including loss between 2 tensors as a whole, across the time axis and across the features axis;
* Reconstruction loss 2: 2-part probability loss between 2 tensors. The first part is just the KL loss between each feature at each time steps (5 timesteps) and the next part is MSE between the correlation matrix between true and generated tensors.
* KL loss: The usual KL loss between the output of encoder with the standard Gaussian distribution.

As I defined the model with the training step function inside, for reproducibility, just use the command model.fit().

# Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Overall | Marginal score | Correlation score | Auto-correlation score | Vale-at-Risk score | Expected shortfall score | AUC |
| 4.833 | 28.3985 | 10.2405 | 0.1354 | 0.0112 | 0.0096 | 0.9592 |

# Possible improvements

I want to point out that, even though I got a pretty nice score for each category, some improvements can still be made. One can consider:

* Substitute the normal ‘Dense’ layer with those recurrent layers (LSTM, RNN, …);
* Adjust the loss function, especially the first part of the reconstruction loss 2. I personally feel that the KL contribute little to the learning process, and it can be substitute by other probability loss such as PSI (a symmetric version of KL) or wassertein loss.