Building Deep Tracking Portfolios through deep learning tools: one application case of Autoencoders and Variational Autoencoders

Daniel Aragón Urrego
Oscar Eduardo Reyes Nieto
Sarah Daniella Coral

Project
Mathematics of Machine Learning
Bogotá, Colombia

July 12, 2024

Introduction

i. Markowitz (1952) proposed a model known as Mean-Variance (MV), where a portfolio of n risky assets with expected return $\mu_p = \mathrm{w}'\mu$ and variance $\sigma_p^2 = \mathrm{w}'\Sigma\ \mathrm{w}$, where $\mu \in R^{n1}$, $\Sigma \in R^{nn}$, $\mathrm{w} \in R^{n1}$ and $1 \in R^{n1}$.

$$\min_{\{\mathrm{w}\}}\{\sigma_P^2 : \mathrm{w}'\Sigma\ \mathrm{w}\} \quad s.t.\ \mathrm{w}'\mu = \mu_p;\ \mathrm{w}'1 = 1 \ \text{ and } \mathrm{w} \geq 0 \tag{1}$$

ii. According to the MV approach, passive strategies recognize that the market cannot be overcome, so the best path to take for the portfolio manager is following the market.

Index Tracking (IT)

Index Tracking (IT) strategy has the objective to build a portfolio of stocks, **Tracking Portfolio**, with the purpose of replicating the performance of an index market or benchmark.

The methods for constructing a tracking portfolio can be divided into two groups:

Full Index Tracking (FIT)

This strategy involves taking positions in all the stocks that make up the benchmark, at their respective weights.

- It faithfully reproduces the index.
- Need to invest in all index components.
- Costly in terms of transactions.
- Increased management.

Sparse Index Tracking (SIT)

This approach focuses on constructing a portfolio with a small number (cardinality) of benchmark constituents.

- Lower transaction costs.
- Fewer assets better management.
- Increased risk of tracking.
- It may not perfectly replicate the index.
- Requires sophisticated optimisation techniques.

Numerical example

Suppose that:

- Benchmark (index) $=S\&P500$
- N=503 assets
- k: cardinality.

Tracking Portfolio (SPY500; N=503) $\begin{cases} \text{Full Index Tracking-FIT } (k = N) \\ \text{Sparse Index Tracking-SIT } (k < N) \end{cases}$

Tracking Strategy	**Benchmark**	**N**	**k**	**Number of Portfolios**
Full Index	S&P 500	503	503	1
Sparse Index	S&P 500	503	10	2.611×10^{23}
			20	3.013×10^{38}
			30	1.738×10^{51}
			40	2.879×10^{62}
			50	3.170×10^{72}

Tracking Error

Tracking error (TE) is defined as the difference between the performance of the benchmark and the performance of the tracking portfolio.

$$ETE(w) = \frac{1}{T} \|r_b - Xw\|_2^2 ; \quad w_i \geq 0, w'1 = 1 \tag{2}$$

Where:

- X represents the returns of the N assets.
- w represents the weights (vector is constant) of the assets in the tracking portfolio.
- r_b corresponds to the returns of the index.
- T are the days.

The problem

- Full Index Tracking

  In practice, FIT may not be the most efficient strategy due to the higher transaction costs (large position and frequent rebalancing) and illiquid assets [3], [10].

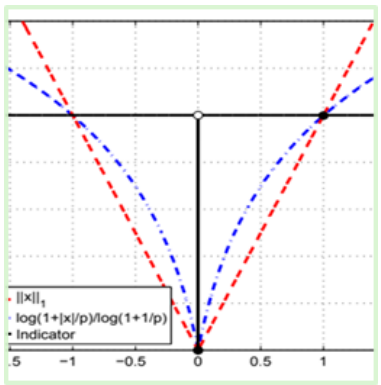
- Sparse Index Tracking

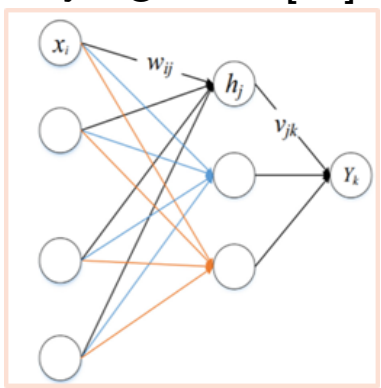
  To limit the number of assets (cardinality constraint) within the tracking portfolio, resulting in an NP-hard constrained optimization problem due to the non-differentiability and non-convexity of the search space (Coleman et al. [5] and Takeda et al. [6]).

  Moreover, implementing this approach is computationally expensive, particularly in high-dimensional environments (Benidis et al. [3] and Shu et al. [4]).

Proposed approaches

Mathematical programming	**Statistical techniques**	**Other approaches: Machine Learning**
Mutunge and Haugland [13]	Benidis et al. [3] Feng and Palomar [14]	Shu et al. [4] Zhang et al. [7] Ouyang et al. [15]

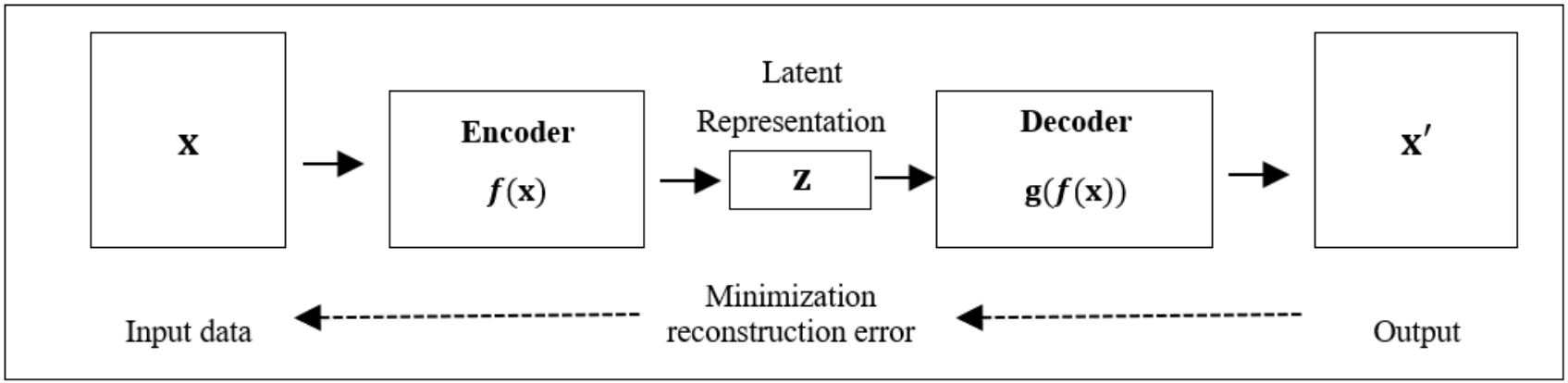




Deep Learning Proposed approach: Autoencoders (AE)

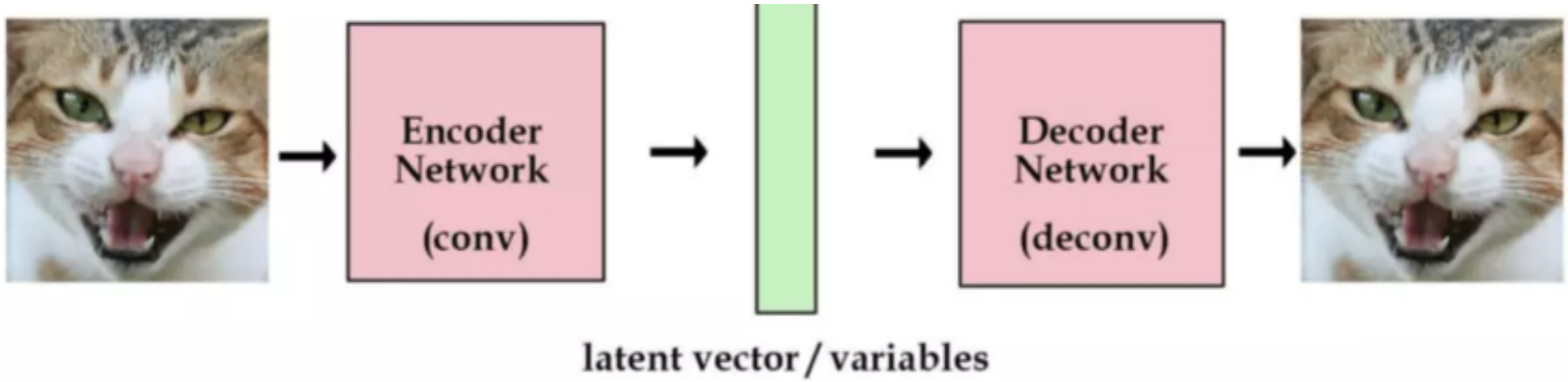
An AE is a neural network designed to learn a feature in an unsupervised manner, by decomposing and reconstructing an original input using a latent representation of its features (Hinton and Salakhutdinov [16]).

Fig 1. General AE structure



Source: Zhang et al. [7]

Fig 2. AE Example



Source: https://shorturl.at/X55pb

Deep Learning Proposed approach: Autoencoders (AE)

A distinctive feature of AEs is that they do not fully replicate the input data, but rather their latent properties.

The AE learning process can be described as the minimization of the reconstruction error, denoted by $L(x, x')$ which is equivalent to the L2-norm:

$$min \sum_{i=1}^{m} (x^{(i)}, x^{`(i)}) = min \sum_{i=1}^{m} ||x^{(i)}, x^{`(i)}||_2 \tag{3}$$

The minimization problem is solved using the backward propagation algorithm, which yields the model weights, w_1 and w_2. These weights are used to obtain:

$$z = f(w_1 x + b_1) \quad and \quad x` = w_2 z + b_2 \tag{4}$$

Deep Learning Proposed approach: Variational Autoencoders (VAE)

Zhang et al. [7] demonstrated that EAs can generate a single value to describe each latent attribute. However, it is preferable to learn from a probability distribution for each latent attribute, rather than from a single value, to produce better generalization.

Additionally, they discovered that this objective could be attained by implementing a generative model known as the variational autoencoder (VAE).

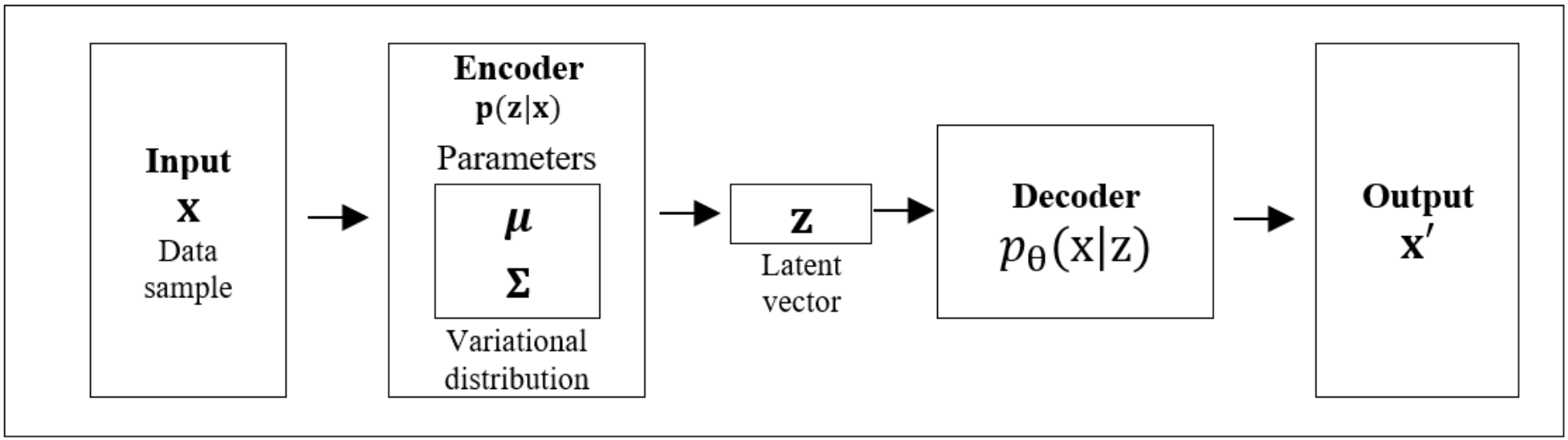
In the VAE, the encoder generates the vector of parameters, θ, which describes a distribution for each dimension of the latent space. This represents the variational posterior estimate, $p(z|x)$. In contrast, the decoder functions as a generative model, representing the probability distribution, $p(x|z)$. In this manner, a joint distribution can be defined as:

$$p(z|x) = p(x)p(x|z) \tag{5}$$

Deep Learning Proposed approach: Variational Autoencoders (VAE)

The VAE learning process can be described as minimizing the reconstruction error of x.

Fig 3. VAE structure



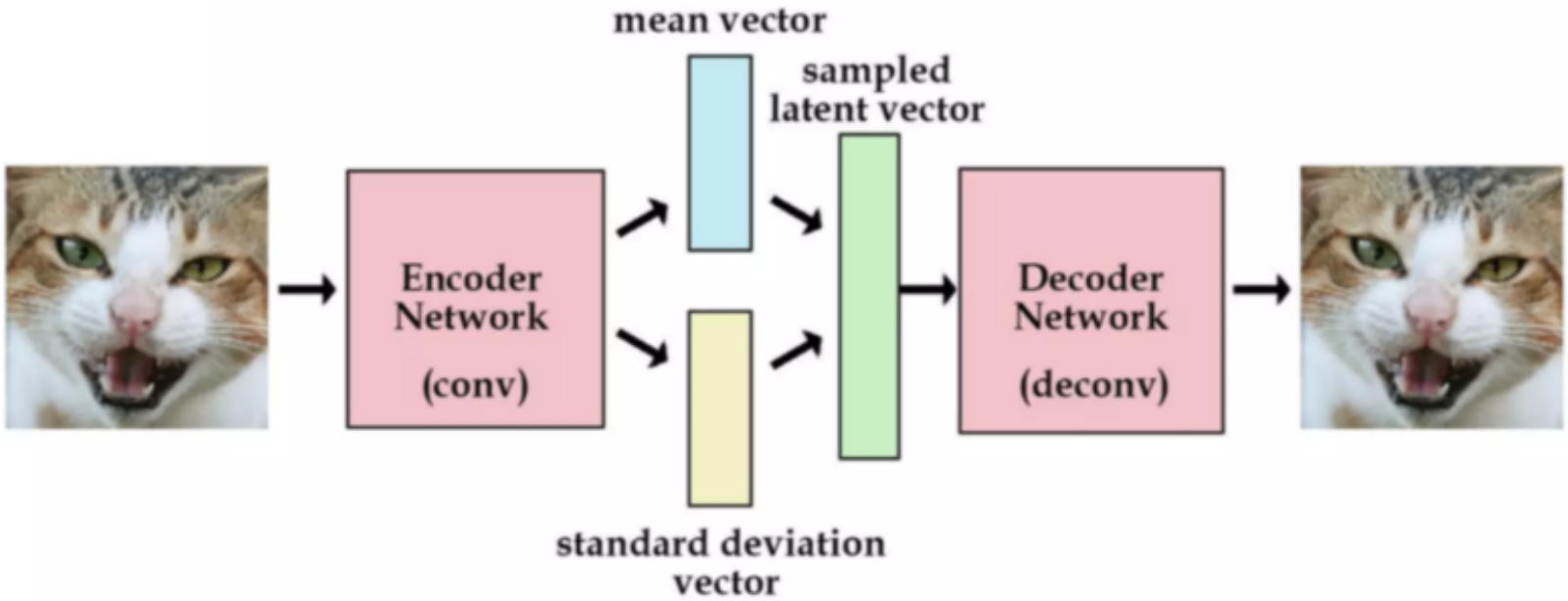
Source: Zhang et al. [7]

The variable z is set as input to the decoder, assuming a standard normal distribution, which tries to reconstruct x as follows (Zhang et al. [7]):

$$z = g(\epsilon, \theta, x) = \mu + \sigma^2 \epsilon, \quad \textit{where} \quad \epsilon \sim N(0, 1) \tag{6}$$

Deep Learning Proposed approach: Variational Autoencoders (VAE)

Fig 4. VAE Example



Source: https://shorturl.at/X55pb

Deep Learning Proposed approach: Variational Autoencoders (VAE)

The loss function of the VAE is the negative log-likelihood, with a regularizer in the form of the Kullback-Leibler (KL) divergence.

The KL loss is equal to the sum of all the KL divergences between $x_i \sim N(\mu_i, \sigma_i^2)$, and the standard normal distribution.

The evidence lower bound (ELBO) is defined as:

$$\text{ELBO} = \mathbb{E}_{q(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}(p_i(z_i|x)\|p(z_i)) \tag{7}$$

The loss function for the VAE, which we seek to minimize, is the negative ELBO:

$$\text{Loss} = -\text{ELBO} = L(x, x') + \lambda \sum_i D_{KL}\left(p_i(z_i \mid x) \parallel p(z_i)\right) \tag{8}$$

Where: $D_{KL} = \frac{1}{2}(-log\sigma_i^2 + \sigma_i^2 + \mu_i^2 - 1)$

$$L = \sum_{i=1}^{m} \left\| x^{(i)} - x'^{(i)} \right\|_2$$

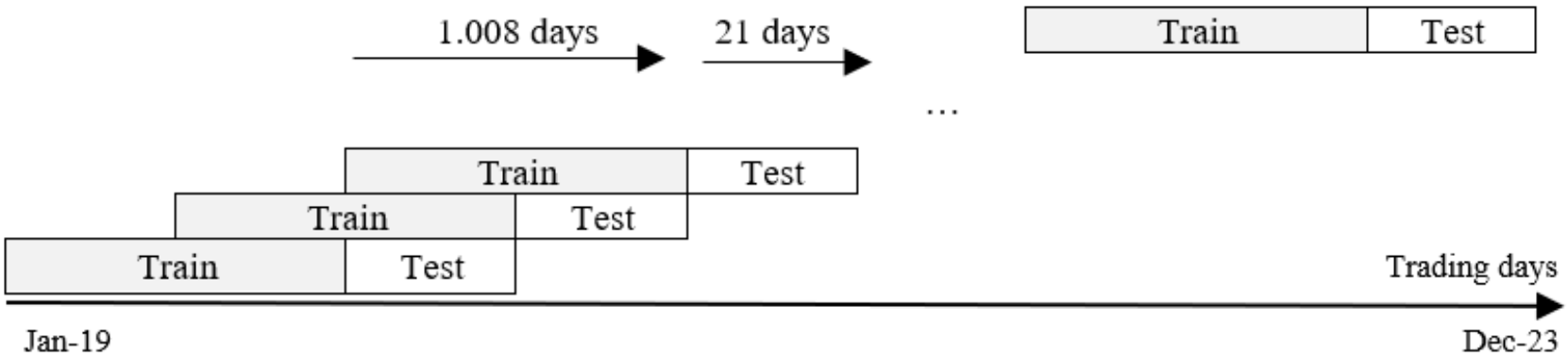
Application

The sparse index tracking problem is implemented for the Nasdaq 100 index (NDX) and its components.

The total analysis period is defined between January 2019 to December 2023. The information is daily and corresponds to the adjusted closing prices, which gives a total of 1258 observations for each variable.

The source of information is Yahoo Finance and the implementation will be done in Python.

Fig 5. Rolling training and testing windows



Source: own elaboration

Experiment 1: AE Arquitecture

Encoder

- Input Layer: 86 neurons (number of available assets).
- First Hidden Layer: 43 neurons, ReLU activation.
- Output Layer: 43 neurons (latent space dimension).

Decoder

- Input Layer: 43 neurons (latent space dimension).
- First Hidden Layer: 43 neurons, ReLU activation.
- Output Layer: 86 neurons, ReLU activation.

Training Parameters

- Optimizer: Adam
- Learning Rate: 0.001 (default in Keras)
- Loss Function: MSE (Mean Squared Error)
- Epochs: 100
- Batch Size: 10

Experiment 1: VAE Arquitecture

Encoder

- Input Layer: 86 neurons (number of available assets)
- First Hidden Layer: 43 neurons, ReLU activation
- Output Layer: z_mean and z_log_var, each with 1 neuron (latent space dimension)

Sampling Layer

- Lambda function for reparameterization to obtain z from z_mean and z_log_var

Decoder

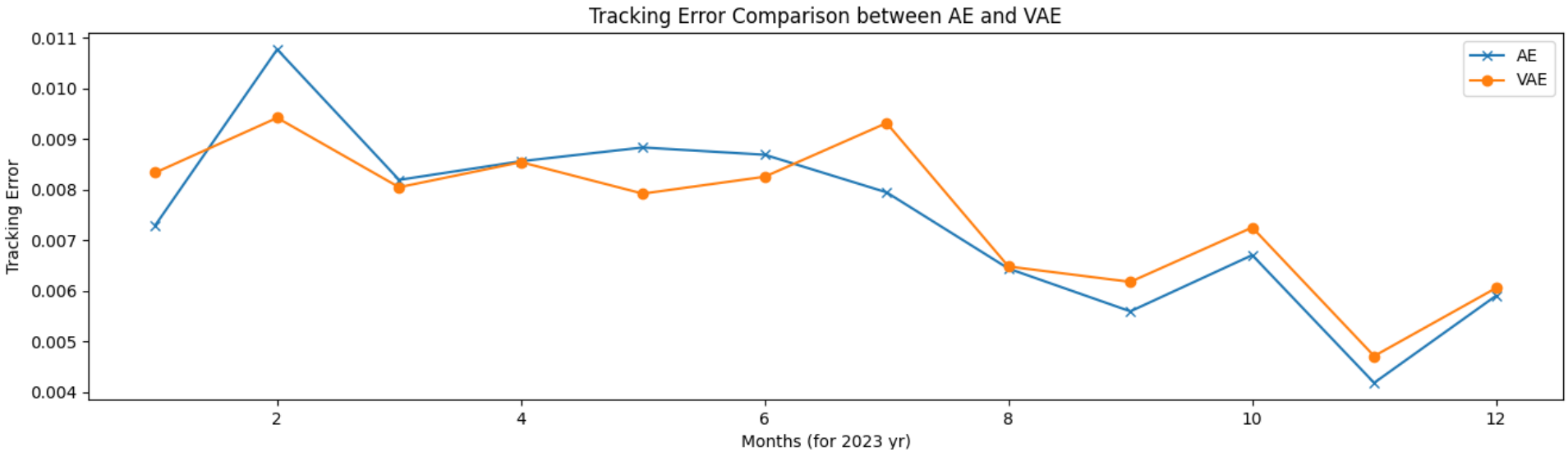
- Input Layer: 2 neurons (latent space dimension)
- First Hidden Layer: 43 neurons, ReLU activation
- Output Layer: 86 neurons, ReLU activation

Training Parameters

- Optimizer: Adam
- Learning Rate: Default (0.001 in Keras)
- Loss Function: MSE for reconstruction loss + KL Divergence loss
- Epochs: 100
- Batch Size: 10

Experiment 1: Results

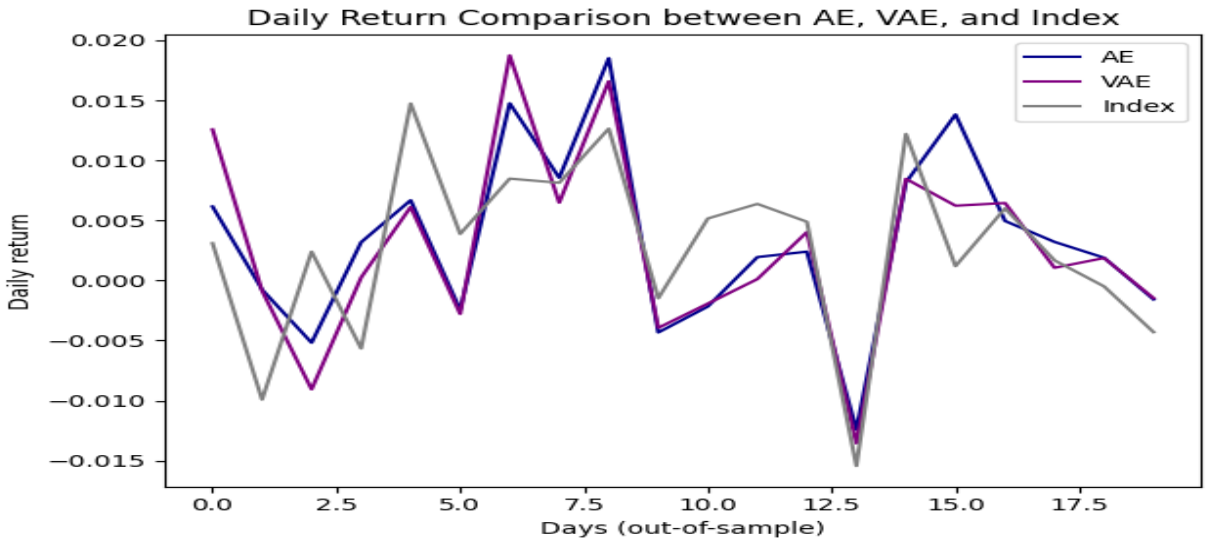
Fig 6. Comparison of ETE results in the out-of-sample period

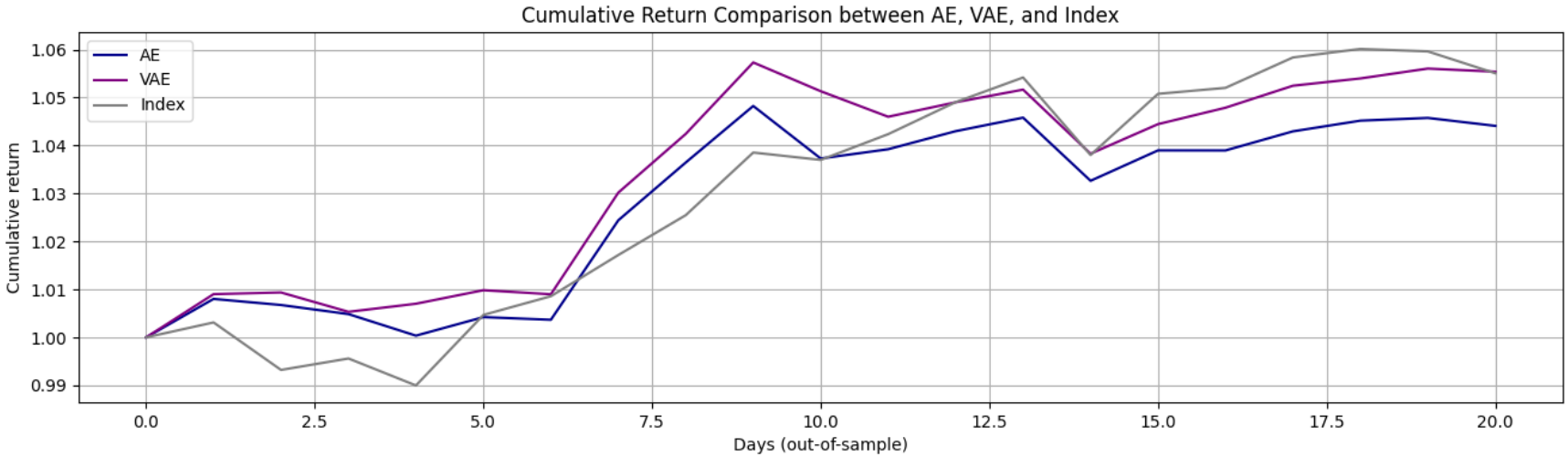


Source: own elaboration

Experiment 1: Results

Fig 7. Comparison of daily (a) and cumulative returns (b) in the out-of-sample period.

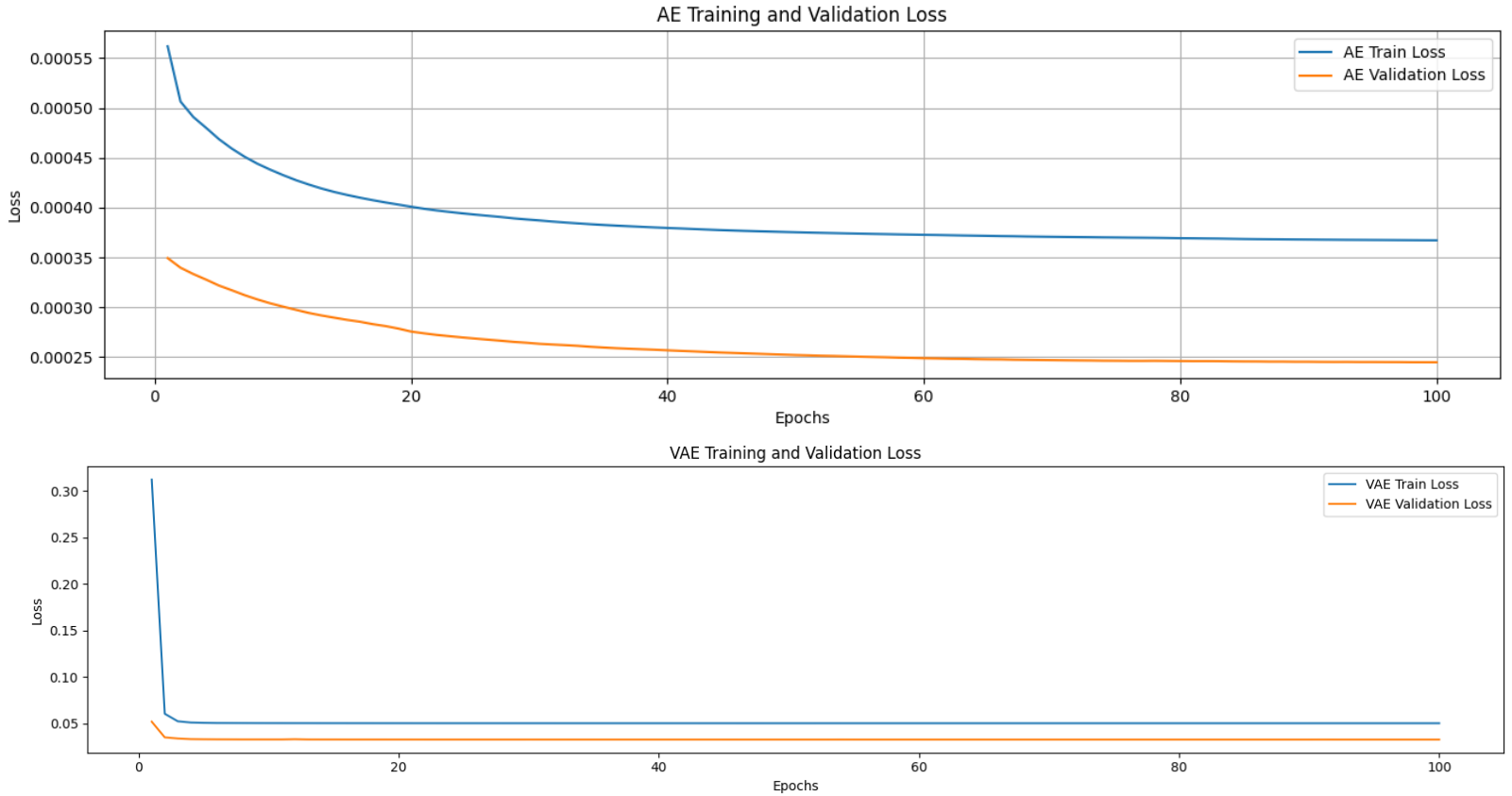




Source: own elaboration

Experiment 1: Results

Fig 8. AE (a) and VAE (b) Loss.



Source: own elaboration

The plot of the loss functions from experiment 1 shows overfitting in the models. Accordingly, two additional experiments are proposed, namely: incorporating dropout (experiment 2) and L2 regularisation (experiment 3).

Experiment 2: AE Arquitecture

Encoder

- Input Layer: 86 neurons (number of available assets).
- First Hidden Layer: 43 neurons, ReLU activation. Dropout rate of 0.2.
- Output Layer: 43 neurons (latent space dimension).

Decoder

- Input Layer: 43 neurons (latent space dimension).
- First Hidden Layer: 43 neurons, ReLU activation. Dropout rate of 0.2.
- Output Layer: 86 neurons, ReLU activation.

Training Parameters

- Optimizer: Adam
- Learning Rate: 0.001
- Loss Function: MSE (Mean Squared Error)
- Epochs: 30
- Batch Size: 10

Experiment 2: VAE Arquitecture

Encoder

- Input Layer: 86 neurons (number of available assets)
- First Hidden Layer: 43 neurons, ReLU activation. Dropout rate of 0.2.
- Output Layer: z_mean and z_log_var, each with 2 neuron (latent space dimension)

Sampling Layer

- Lambda function for reparameterization to obtain z from z_mean and z_log_var

Decoder

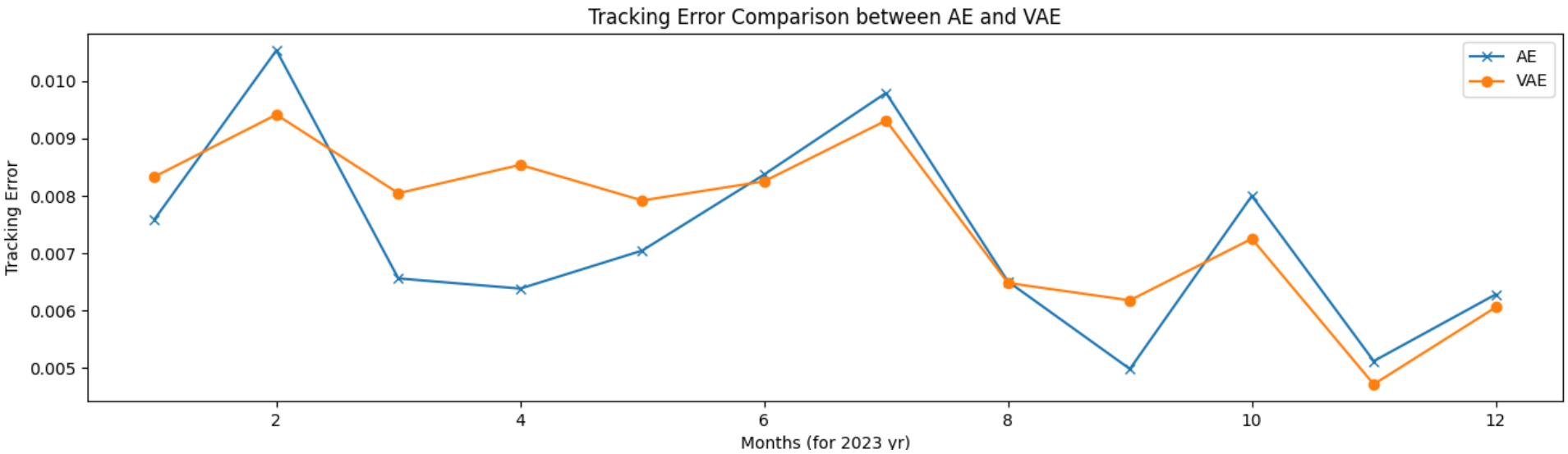
- Input Layer: 2 neurons (latent space dimension)
- First Hidden Layer: 43 neurons, ReLU activation. Dropout rate of 0.2.
- Output Layer: 86 neurons, ReLU activation

Training Parameters

- Optimizer: Adam
- Learning Rate: 0.001
- Loss Function: MSE for reconstruction loss + KL Divergence loss
- Epochs: 30
- Batch Size: 10

Experiment 2: Results

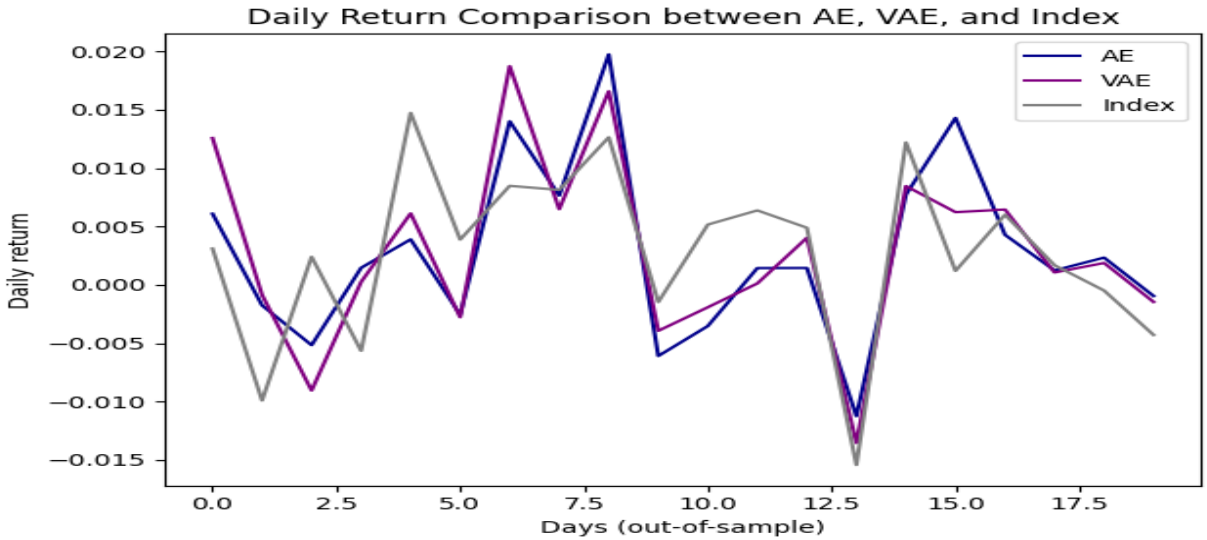
Fig 9. Comparison of ETE results in the out-of-sample period

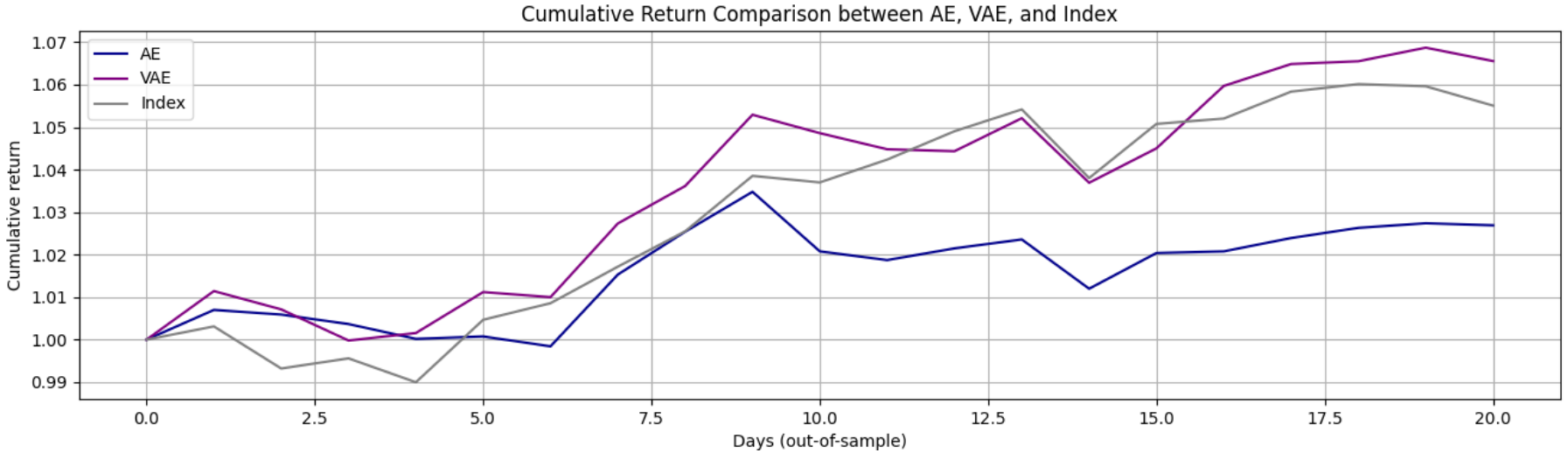


Source: own elaboration

Experiment 2: Results

Fig 10. Comparison of daily (a) and cumulative returns (b) in the out-of-sample period.

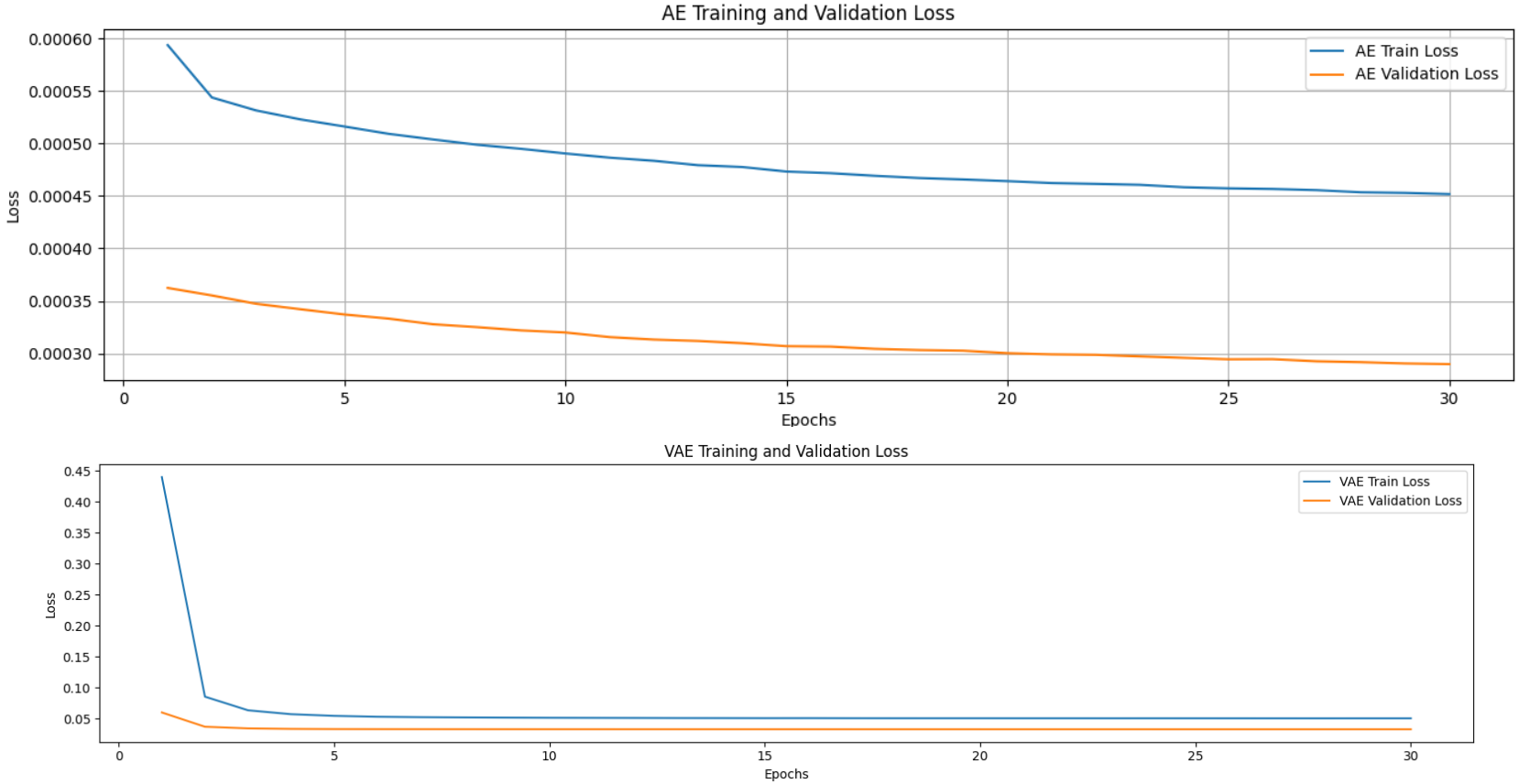




Source: own elaboration

Experiment 2: Results

Fig 11. AE (a) and VAE (b) Loss.



Source: own elaboration

Experiment 3: AE Arquitecture

Encoder

- Input Layer: 86 neurons (number of available assets).
- First Hidden Layer: 43 neurons, ReLU activation, L2 regularization (penalty factor 0.01).
- Output Layer: 43 neurons (latent space dimension).

Decoder

- Input Layer: 43 neurons (latent space dimension).
- First Hidden Layer: 43 neurons, ReLU activation, L2 regularization (penalty factor 0.01).
- Output Layer: 86 neurons, ReLU activation.

Training Parameters

- Optimizer: Adam
- Learning Rate: 0.001
- Loss Function: MSE (Mean Squared Error)
- Epochs: 30
- Batch Size: 10

Experiment 3: VAE Arquitecture

Encoder

- Input Layer: 86 neurons (number of available assets)
- First Hidden Layer: 43 neurons, ReLU activation, L2 regularization (penalty factor 0.01).
- Output Layer: z_mean and z_log_var, each with 2 neurons (latent dimension), L2 regularization (penalty factor 0.01)

Sampling Layer

- Lambda function for reparameterization to obtain z from z_mean and z_log_var

Decoder

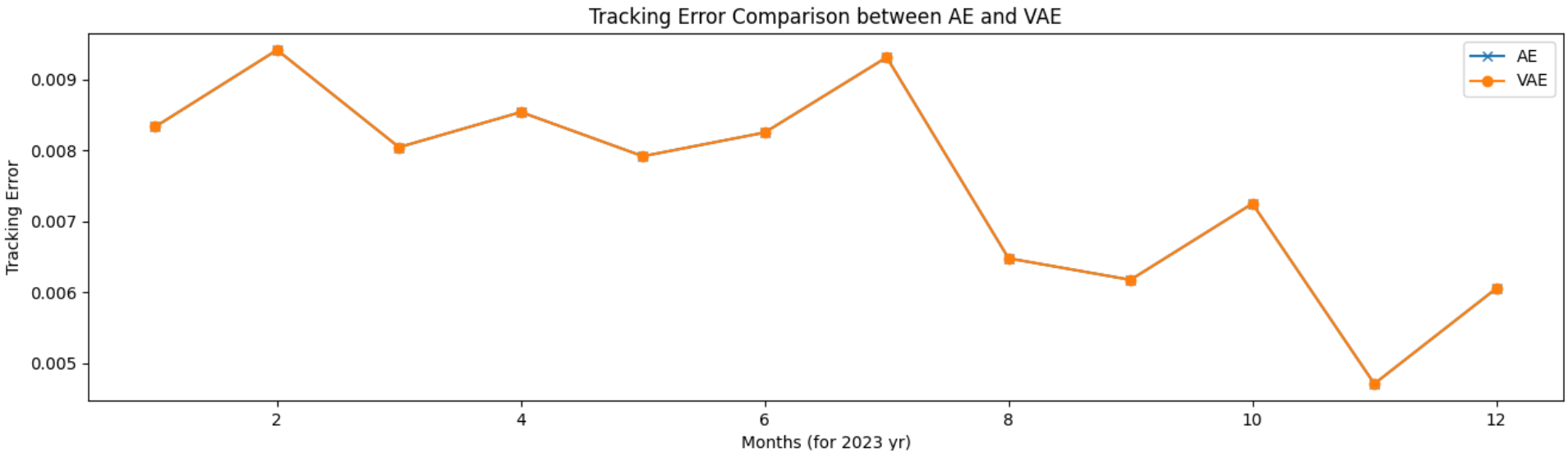
- Input Layer: 2 neurons (latent space dimension)
- First Hidden Layer: 43 neurons, ReLU activation, L2 regularization (penalty factor 0.01).
- Output Layer: 86 neurons, ReLU activation, L2 regularization (penalty factor 0.01).

Training Parameters

- Optimizer: Adam
- Learning Rate: 0.001
- Loss Function: MSE for reconstruction loss + KL Divergence loss
- Epochs: 30
- Batch Size: 10

Experiment 3: Results

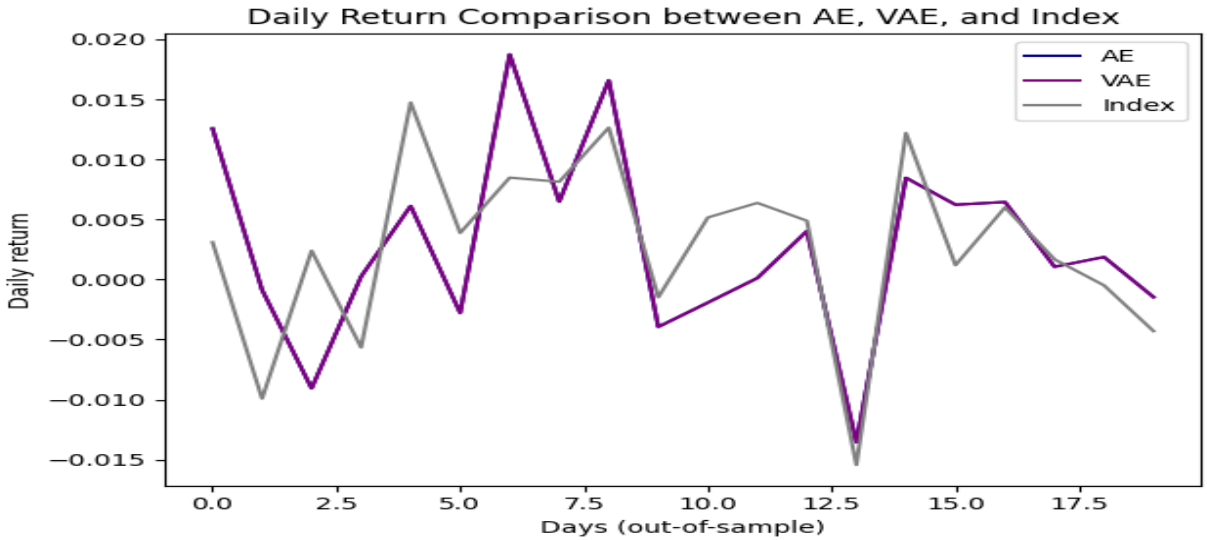
Fig 12. Comparison of ETE results in the out-of-sample period

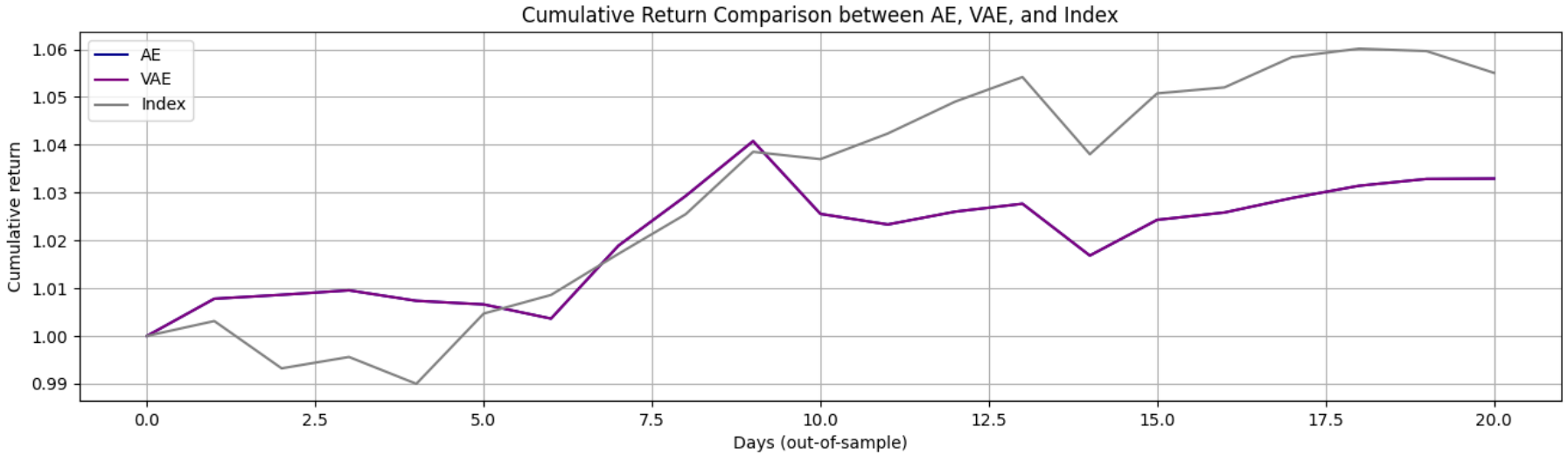


Source: own elaboration

Experiment 3: Results

Fig 13. Comparison of daily (a) and cumulative returns (b) in the out-of-sample period.

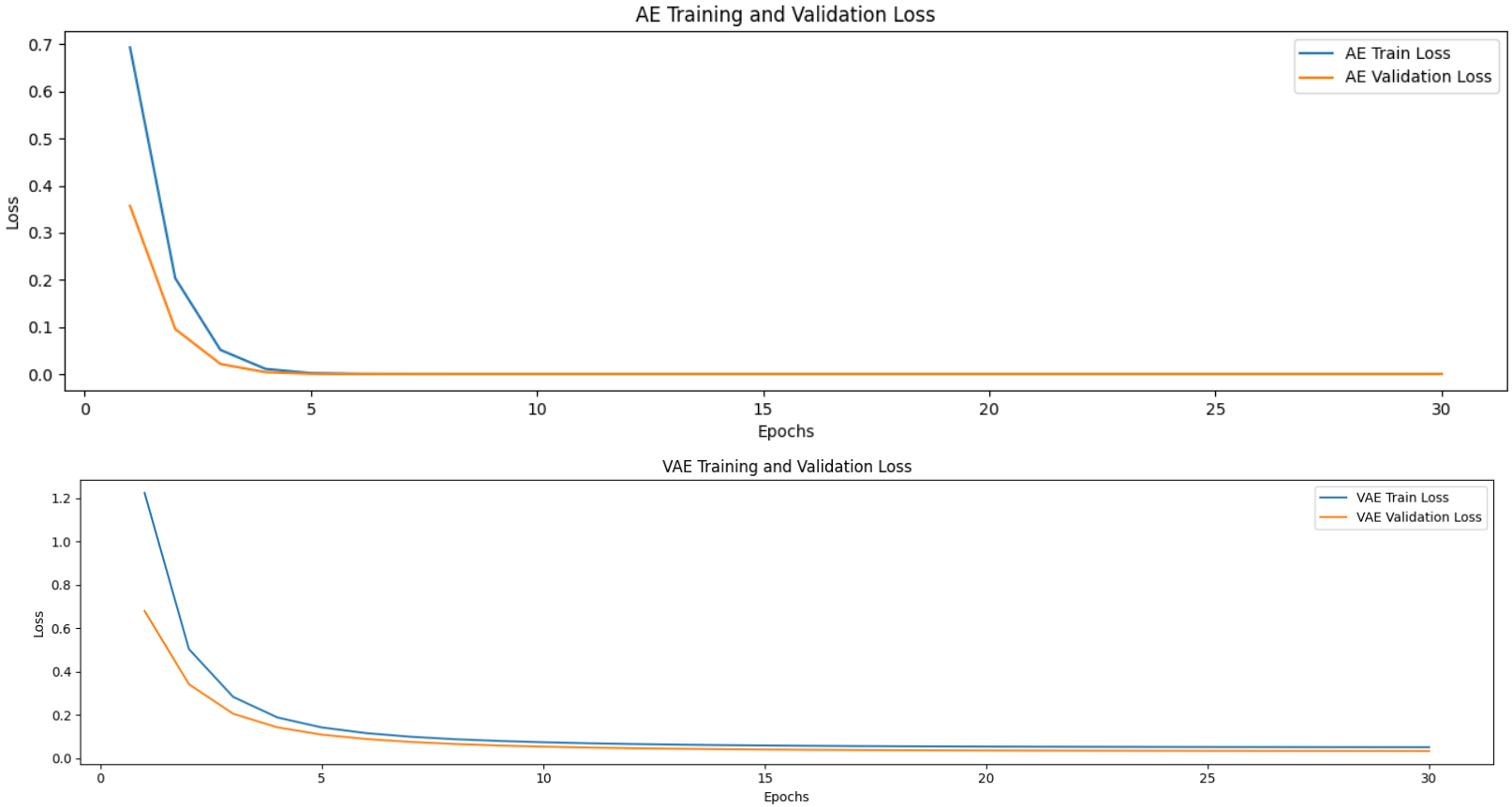




Source: own elaboration

Experiment 3: Results

Fig 14. AE (a) and VAE (b) Loss.



Source: own elaboration

References

1. Roll, R.: A mean/variance analysis of tracking error. The Journal of Portfolio Management, 18(4), 13-22 (1992). doi: 10.3905/jpm.1992.701922
2. Jorion, P.: Portfolio Optimization with Tracking-Error Constraints. Financial Analysts Jour-nal, 59, 70-82 (2003). doi: 10.2469/faj.v59.n5.2565
3. Benidis, K., Feng, Y. and Palomar, D.: Sparse Portfolios for High-Dimensional Financial Index Tracking, IEEE Trans. on Signal Processing, 66(1), 155-170 (2018). doi: 10.1109/TSP.2017.2762286
4. Shu, L., Shi, F., Tian, G.: High-dimensional index tracking based on the adaptive elastic net. Quantitative Finance, 20(9), 1513-1530 (2020). doi: 10.1080/14697688.2020.1737328
5. Coleman, T. F., Li, Y., Henniger, J.: Minimizing tracking error while restricting the num-ber of assets. Journal of Risk, 8(4), 33 (2006).
6. Takeda, A., Niranjan, M., Gotoh, J. Y., Kawahara, Y.: Simultaneous pursuit of out-of-sample performance and sparsity in index tracking portfolios. Computational Management Science, 10, 21-49 (2013). doi: 10.1007/s10287-012-0158-y
7. Zhang C, Liang S, Lyu F and Fang L.: Stock-Index Tracking Optimization Using Auto-Encoders. Front. Phys. 8:388 (2020). doi: 10.3389/fphy.2020.00388
8. Markowitz, H.: Portfolio selection. J. Fin., (7)1, 77-91 (1952).
9. Markowitz, H.: Portfolio selection: Efficient diversification of investments, New Haven, Yale University Press (1959)

References

10. Strub, O., Baumann, P.: Optimal Construction and Rebalancing of Index-tracking Portfolios. European Journal of Operational Research, 264(1), 370-387 (2018). doi: 10.1016/j.ejor.2017.06.055
11. Jansen, R., Van Dijk, R. (2002). Optimal benchmark tracking with small portfoli-os. Journal of Portfolio Management, 28(2), 33-39. doi:10.3905/jpm.2002.319830
12. Silva, J. C. S., de Almeida Filho, A. T. (2024). A systematic literature review on solution approaches for the index tracking problem. IMA Journal of Management Mathemat-ics, 35(2), 163-196. doi: 10.1093/imaman/dpad007
13. Mutunge, P., Haugland, D. (2018). Minimizing the tracking error of cardinality con-strained portfolios. Computers Operations Research, 90, 33-41. doi: 10.1016/j.cor.2017.09.002
14. Feng, Y., Palomar, D. P. (2016). A signal processing perspective on financial engineer-ing. Foundations and Trends® in Signal Processing, 9(1–2), 1-231.
15. Ouyang, H., Zhang, X., Yan, H.: Index tracking based on deep neural network. Cognitive Systems Research, 57, 107-114 (2019). doi: 10.1016/j.cogsys.2018.10.022
16. Kingma, D. P., Welling, M. (2019). An introduction to variational autoenco-ders. Foundations and Trends® in Machine Learning, 12(4), 307-392.
17. Hinton, G. E. Salakhutdinov, R.: Reducing the dimensionality of data with neural net-works, Science 313(5786), 504–507, (2006). doi: 10.1126/science.1127647

Thanks!