UNIL | Université de Lausanne

HEC Lausanne

Université de Lausanne

Faculty of Finance

Deep variational auto-encoder and normalizing flow for stock return prediction

Advanced Data Sciences

Author
Andreas Duquenne

Professor
Simon Scheidegger

October 14, 2024

1 Abstract

This projects aim at exploring the world of portfolio optimization. It mainly consist in the understanding of the forecasting of a stock return. This method consider dividing the modelling in two tasks:

1. Forecasting factor variables that a stock return would rely on.
2. Forecasting the stock return itself using the forecasted factors.

The code base can be found at https://github.com/AnDuquenne/Advanced-Data-Analytics

2 Review of literature

The characterization of a stock return is a widely discussed topic in finance. Accurately determining the influence of these returns could lead to significant wins. From equilibrium theory, we have that the return should be a compensation for a certain risk. However, empirically, returns have proven to be dependent on other factors. Moreover, the risk is a difficult things to determine.

Considering a factor model for excess returns of the form:

$$r_{i,t+1} = \alpha_{i,t} + \beta'_{i,t} f_{t+1} + \epsilon_{i,t+1}, \quad E_t\left[\epsilon_{i,t+1}\right] = 0 \quad (1)$$

By decomposing stock returns into a linear combination of factor exposures, we can achieve dimensionality reduction and can analyze an ever-changing universe of stock. This factor model is the setting for most empirical analysis of expected returns. This factor model is generally addressed by selecting factors considering empirical knowledge about stock returns, treat them as observable, then learn α and β using regression. However, another way to handle this would be to consider these factors as latent and use factor analysis techniques to simultaneously estimate the factors and parameters from the panel of realized returns.

A first method as been introduced by Kelly & al.[1] using instrumented principal components analysis as a factor analysis technique. They provide empirical evidence that these so-called anomaly asset characteristics in fact proxy for unobservable and timevarying exposures to risk factors, and shows that characteristics contain little (if any) anomalous return predictability once their explanatory power for factor exposures has been accounted for. In other words, characteristics appear to predict returns because they help pinpoint compensated aggregate risk exposures. The asset pricing model proposed by KPS assumes that individual returns possess a K-factor structure:

$$r_{i,t} = \beta\left(z_{i,t-1}\right)' f_t + u_{i,t}. \quad (2)$$

- f_t: Factors (treated as latent)
- $\beta\left(z_{i,t-1}\right) \in \Re^{K\times 1}$: The conditional factor exposure
- $z_{i,t-1} \in \Re^{P\times 1}$: Asset characteristics

With $P > K$ and N the individual returns. Kelly & al. make the assumption that the map from P characteristics to K betas is linear, using instrumented principal components analysis as a mapping function.

A further improvement of this model is provided by Gu & al. [2], refuting the linearity assumption between the characteristics and the betas. They consider a non-linear models from the autoencoder family. The autoencoders can be seen as a generalization of the PCA, they remain an unspurvised model, but are typically trained using gradient descent. Neither method, in their standard form, uses information in covariates to guide dimension reduction.

Tepelyan & al.[3] introduce generative models to express the return forecasting problem. Their model aims at learning:

$$p\left(\{r_{i,T+1}\}_{i=1}^{N_{T+1}} \mid \mathcal{F}_T\right) \quad (3)$$

In other words, defining the joint distribution of the N_{T+1} returns $r_{i,T+1} \in \Re$ at time $T+1$ given historical information $\mathcal{F}_T$. Considering factor analysis as in the previously proposed models, the likelihood can be re-expressed as:

$$\int \left(\prod_{i=1}^{N_{T+1}} p\left(r_{i,T+1} \mid \mathbf{F_{T}}_{+1}, \mathcal{F}_T\right) \right) p\left(\mathbf{F_{T}}_{+1} \mid \mathcal{F}_T\right) d\mathbf{F_{T}}_{+1} \quad (4)$$

where $\mathbf{F_{T}}_{+1}$, the factors, is the information at $T+1$. This model will be the basis of this project.

3 Background

This section aim at introducing the technical tools used in order to model the data.

3.1 Least squares linear regression

The model is of the form:

$$p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}\left(y \mid w_0 + \boldsymbol{w}^\top \boldsymbol{x}, \sigma^2\right) \tag{5}$$

where $\boldsymbol{\theta} = \left(w_0, \boldsymbol{w}, \sigma^2\right)$ are the parameters of the model. The model will be evaluated by minimizing the residual sum of squares:

$$\begin{aligned} \mathrm{RSS}(\boldsymbol{w}) &= \frac{1}{2}\sum_{n=1}^{N}\left(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n\right)^2 = \frac{1}{2}\|\mathbf{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 \\ &= \frac{1}{2}(\mathbf{X}\boldsymbol{w} - \boldsymbol{y})^\top(\mathbf{X}\boldsymbol{w} - \boldsymbol{y}) \end{aligned} \tag{6}$$

Minimizing with respect to w:

$$\begin{aligned} &\nabla_{\boldsymbol{w}} \mathrm{RSS}(\boldsymbol{w}) = \mathbf{X}^\top\mathbf{X}\boldsymbol{w} - \mathbf{X}^\top\boldsymbol{y} \\ \iff &\mathbf{X}^\top\mathbf{X}\boldsymbol{w} = \mathbf{X}^\top y \\ \iff &\hat{\boldsymbol{w}} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\boldsymbol{y} \end{aligned} \tag{7}$$

3.2 ARIMA and EMA

3.3 Generative models

The generative model are part of deep unsupervised learning, we try to capture rich patterns in raw data without label. Generative models aim at reconstructing the orignal raw data distribution (not only the distribution of the training set), and then can be used as a simulator of the data. Its purpose is to generate synthetic but realistic high-dimensional data that is as close as possible from the unknown data distribution $p(\mathbf{x})$, but for which we have empirical samples. Thus, we want to build the distribution p from which we could sample values $\mathbf{x}$, and we would like to adjust the parameters of this distribution θ such that the samples that we create are as close as possible to the true original data distribution. By understanding how the data are generated we understand how things work.

3.3.1 Auto-encoders

The point is to find meaningful degrees of freedom that describe the signal and are of lesser dimension. Said simply, we want to describe the signal with fewer features.

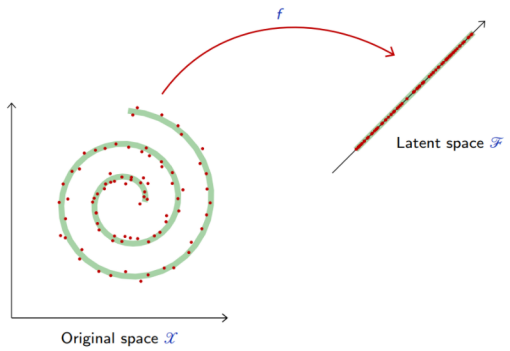

Figure 1

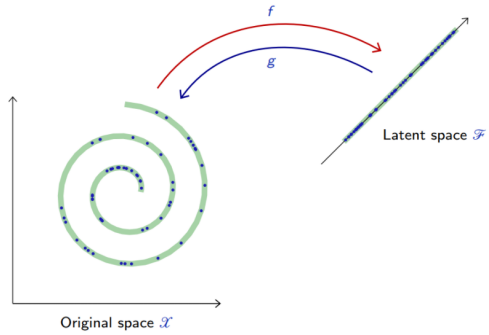

Figure 2

We can project data on a smaller space, called the latent space. Of course we will loose some information about the original data, but we should be able to reconstruct the original data without having lost too much information, as can be seen on *fig.*2. Therefore the f function can be considered as an encoder of the data from an high dimensional space to a smaller dimensional space, and the g function as a decoder that will regenerate the original distribution. We want $g \circ f$ as close as possible as identity on the data. As can also be seen on the figure, the auto-encoder can be used to remove noise as it can generalize the data.

Consider the evaluation of the auto-encoder, in order to train and assess the model. Let $p(\mathbf{x})$ be the data distribution over $\mathcal{X}$. A good auto-encoder could be characterized with the reconstruction loss

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[||\mathbf{x} - g \circ f(\mathbf{x})||^2\right] \approx 0.$$

Given two parameterized mappings $f(\cdot;\theta_f)$ and $g(\cdot;\theta_g)$, training consists of minimizing an empiri-

cal estimate of that loss,

$$\theta_f, \theta_g = \arg\min_{\theta_f, \theta_g} \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}_i - g(f(\mathbf{x}_i, \theta_f), \theta_g)||^2.$$

If we consider a linear auto-encoder (linear combination of the features),

$$f : \mathbf{z} = \mathbf{U}^T \mathbf{x}$$
$$g : \hat{\mathbf{x}} = \mathbf{U}\mathbf{z},$$

with $\mathbf{U} \in \mathbb{R}^{p \times d}$, the reconstruction error reduces to

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[||\mathbf{x} - \mathbf{U}\mathbf{U}^T \mathbf{x}||^2 \right].$$

In this case, an optimal solution is given by PCA. However, in this project we will consider the f and g function to be non-linear, more specifically, fully-connected neural networks.

3.4 Variational inference

Variational inference is a general inference algorithm that we can use for any kind of probabilistic model that we would define.

3.4.1 Latent variable model

We consider a classical latent variable model: a probabilistic model that relates a set of observable variables $x \in \mathcal{X}$, the data, to a set of unobservable variables $z \in \mathcal{Z}$, the representation in the latent space.

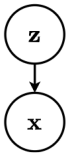

Figure 3: Bayesian network of the distribution $p(x, z)$, oriented acyclic graph. We need to define the prior distribution $p(z)$ and the conditionnal distribution $p(x|z)$

The inference process,

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} \qquad \text{(bayes)}$$
$$= \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{\underbrace{\int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}}_{\text{intractable}}}$$

The integral is the marginal distribution of x: $p(x)$; the computation directly depends on the size of the space $\mathcal{Z}$. The integral is intractable for all interesting problems, we will not be able to, even numerically, evaluate the value of $p(\mathbf{x})$. This is solved by transforming this in an optimization problem.

3.4.2 Variational inference

Instead of trying to compute directly the real posterior $p(\mathbf{z}|\mathbf{x})$ we will approximate that distribution. We will postulate some parametric family of distribution, called a variational family, denoted $q(z; \nu)$, and we will try to identify parameters to this distribution such that this one will we as close as possible to the posterior distribution that we would like to evaluate. In this project, the variational family considered will be the family of Gaussian distributions. We will start from some initial values of the parameters ν_{init}, an then try to optimize the values of the parameters such that the q distribution, our approximation, is as close as possible to the original posterior. We reach the optimal parameter values $\nu*$ when we minimize some loss, which is defined between the two distribution. Here we will use the KL divergence as a loss.

$$\mathrm{KL}(p(x)||q(x)) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} dx$$

So the KL will be equal to 0 when the two distributions are identical. The optimization problem becomes:

$$\begin{aligned}
\nu* &= \arg\min_{\nu} \mathrm{KL}(q(\mathbf{z}; \nu)||p(\mathbf{z}|\mathbf{x})) \\
&= \arg\min_{\nu} \mathbb{E}_{q(\mathbf{z};\nu)} \left[\log \frac{q(\mathbf{z}; \nu)}{p(\mathbf{z}|\mathbf{x})} \right] \\
&= \arg\min_{\nu} \mathbb{E}_{q(\mathbf{z};\nu)} \left[\log q(\mathbf{z}; \nu) - \log \underbrace{p(\mathbf{z}|\mathbf{x})}_{= \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{x})}} \right] \\
&= \arg\min_{\nu} \mathbb{E}_{q(\mathbf{z};\nu)} \left[\log q(\mathbf{z}; \nu) - \log p(\mathbf{x}, \mathbf{z}) \right] \\
&\qquad + \underbrace{\log p(\mathbf{x})}_{\text{does not depend on } \mathbf{z}} .
\end{aligned}$$

That already simplify the problem a bit because we are only interested in the optimal parameter values the solution to this optimization problem will be the same as just minimizing the expectation without taking care of $p(\mathbf{x})$. The original problem

becomes equivalent to,

$$\begin{aligned}\nu* &= \arg\min_{\nu} \mathbb{E}_{q(\mathbf{z};\nu)}\left[\log q(\mathbf{z};\nu) - \log p(\mathbf{x},\mathbf{z})\right] \\ &= \arg\max_{\nu} \underbrace{\mathbb{E}_{q(\mathbf{z};\nu)}\left[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\nu)\right]}_{\text{ELBO}(\mathbf{x};\nu)\text{: evidence lower bound objective}}\end{aligned}$$

The ELBO does not depend on $\mathbf{z}$ because the z are marginals out of the expectation. Considering a bayesian network we are able to evaluate the joint $p(\mathbf{x},\mathbf{z})$. By definition:

$$\begin{aligned}&\text{ELBO}(\mathbf{x};\nu) \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\nu)}\left[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\nu)\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\nu)}\left[\log p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) - \log q(\mathbf{z};\nu)\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\nu)}\left[\log p(\mathbf{x}|\mathbf{z})\right] - \text{KL}(q(\mathbf{z};\nu)||p(\mathbf{z}))\end{aligned}$$

So what we will try to do is to identify parameters value ν such that the $\mathbf{z}$ that we draw from this configuration (q parametrized by ν) explain well the x values from our data.

Therefore, maximizing the ELBO:

- encourages distributions to place their mass on configurations of latent variables that explain the observed data (first term);
- encourages distributions close to the prior (second term).

Provided that we can evaluate the gradient of the ELBO on the parameters we can proceed by gradient descent.

3.4.3 Variational auto-encoders

Variational auto-encoders are auto-encoders just as before except that we equip both the $\mathbf{x}$ and the $\mathbf{z}$ space with a probability distribution, and we will not directly define a mapping from a single point x to a single point z, but a probabilistic mapping that will correspond to $p(\mathbf{z}|\mathbf{x})$, and vice versa.

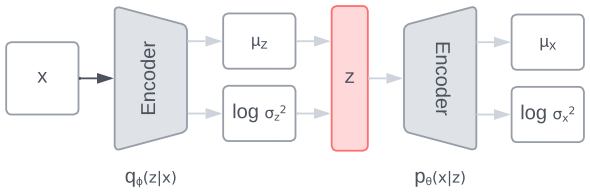

Figure 4

- The prior $p(\mathbf{z})$ is prescribed, chosen to be Gaussian.
- The likelihood $p(\mathbf{x}|\mathbf{z};\theta)$ is parameterized with a generative network NN_θ (or decoder) that takes as input $\mathbf{z}$ and outputs parameters $\varphi = \text{NN}_\theta(\mathbf{z})$ to the data distribution. E.g.,

$$\begin{aligned}\mu, \sigma &= \text{NN}_\theta(\mathbf{z}) \\ p(\mathbf{x}|\mathbf{z};\theta) &= \mathcal{N}(\mathbf{x};\mu,\sigma^2\mathbf{I})\end{aligned}$$

- The approximate posterior $q(\mathbf{z}|\mathbf{x};\varphi)$ is parameterized with an inference network NN_φ (or encoder) that takes as input $\mathbf{x}$ and outputs parameters $\nu = \text{NN}_\varphi(\mathbf{x})$ to the approximate posterior. E.g.,

$$\begin{aligned}\mu, \sigma &= \text{NN}_\varphi(\mathbf{x}) \\ q(\mathbf{z}|\mathbf{x};\varphi) &= \mathcal{N}(\mathbf{z};\mu,\sigma^2\mathbf{I})\end{aligned}$$

Now instead of having a regular auto-encoder which have a direct mapping from one point to the other, we have a mapping from a point in $\mathcal{X}$ to a full distribution in $\mathcal{Z}$ (the posterior $q(\mathbf{z}|\mathbf{x})$, and in the opposite way the likelihood $p(\mathbf{x}|\mathbf{z})$ is a mapping from a single point in the latent space $\mathcal{Z}$ that produce a full distribution of values in the original space. And these f and g are parametrized by neural networks.

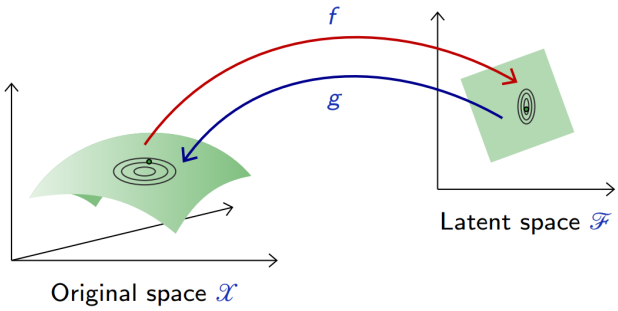

Figure 5

We can use variational inference to optimize the parameters, except that we will optimize jointly the parameters of the likelihood model and of the posterior model. We will now have an *ELBO* which depends on both the parameters of the likelihood θ, and of the parameters the approximate posterior φ. We want,

$$\begin{aligned}\theta^*, \varphi^* &= \arg\max_{\theta,\varphi} \text{ELBO}(\mathbf{x};\theta,\varphi) \\ &= \arg\max_{\theta,\varphi} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x},\mathbf{z};\theta) - \log q(\mathbf{z}|\mathbf{x};\varphi)\right] \\ &= \arg\max_{\theta,\varphi} \underbrace{\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x}|\mathbf{z};\theta)\right]}_{1} - \underbrace{\text{KL}(q(\mathbf{z}|\mathbf{x};\varphi)||p(\mathbf{z}))}_{2}.\end{aligned}$$

1. Given some fixed generative network θ (so if the generative process from z to x is fixed), we

want to put the mass of the latent variables, by adjusting φ, such that they explain the observed data, while remaining close to the prior.

2. Given some inference network φ, we want to put the mass of the observed variables, by adjusting θ, such that they are well explained by the latent variables.

3.5 Normalizing flow

The normalizing flow has been presented by Papamakarios & al.[7]. Normalizing flows consist in a series of bijective functions. From the *fig.*6, it can be seen that the normalizing flows transform a complex data to a simple Gaussian distribution.

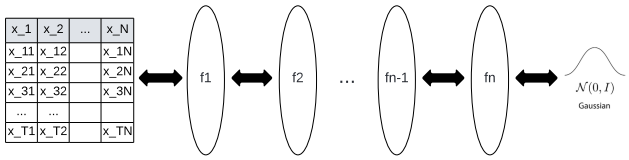

Figure 6: Normalizing flow

The result of this approach is a mechanism to construct new families of distributions by choosing an initial density and then chaining together some number of parameterized, invertible and differentiable transformations. The new density can be sampled from (by sampling from the initial density and applying the transformations) and the density at a sample (i.e., the likelihood) can be computed as above.

Mathematically, let a random variable $\mathbf{Z} \in \Re^D$ be a random variable with a known pdf of the form:

$$p_{\mathbf{Z}} : \Re^D \longrightarrow \Re$$

And let $g(.)$ be an invertible function such that:

$$\mathbf{Y} = g(\mathbf{Z})$$

Then using change of variable we can compute the pdf of the random variable $\mathbf{Y}$:

$$\begin{aligned} p_{\mathbf{Y}}(\mathbf{y}) &= p_{\mathbf{Z}}(\mathbf{f}(\mathbf{y}))|\det \mathrm{D}\mathbf{f}(\mathbf{y})| \\ &= p_{\mathbf{Z}}(\mathbf{f}(\mathbf{y}))|\det \mathrm{D}\mathrm{g}(\mathbf{f}(\mathbf{y}))|^{-1} \end{aligned}$$

where f is the inverse of g, and,

$$\mathrm{Df}(\mathbf{y}) = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}$$

$$\mathrm{Dg}(\mathbf{z}) = \frac{\partial \mathbf{g}}{\partial \mathbf{z}}$$

are the Jacobian of respectively f and g.

Let $g_1, g_2, \ldots, g_N$ be a set of N bijective functions, and $\mathrm{g} = \mathrm{g}_N \circ \mathrm{g}_{N-1} \circ \cdots \circ \mathrm{g}$ the composition of the functions. Then g, the composition, is also bijective, and its inverse is given by:

$$\mathbf{f} = \mathbf{f}_1 \circ \cdots \circ \mathbf{f}_{N-1} \circ \mathbf{f}_N \tag{8}$$

And the determinant of the Jacobian is,

$$\det \boldsymbol{D}\boldsymbol{f}(\mathbf{y}) = \prod_{i=1}^{N} \det D\mathbf{f}_i\left(\mathbf{x}_i\right)$$

with,

$$\mathrm{D}\mathbf{f}_i(\mathbf{y}) = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}}$$

is the Jacobian of $\mathbf{f}_i$.

Training

Given a set of observable data,

$$\mathcal{D} = \left\{\mathbf{y}^{(i)}\right\}_{i=1}^{M},$$

the flow $\mathbf{g}(.;\theta)$, and $p_{\mathbf{Z}}(.;\phi)$. we can perform likelihood estimation of the parameters:

$$\begin{aligned} \log p(\mathcal{D} \mid \theta, \phi) &= \sum_{i=1}^{M} \log p_{\mathbf{Y}}\left(\mathbf{y}^{(i)} \mid \theta, \phi\right) \\ &= \sum_{i=1}^{M} \log p_{\mathbf{Z}}\left(\mathbf{f}\left(\mathbf{y}^{(i)} \mid \theta\right) \mid \phi\right) \\ &\quad + \log \left|\det \mathrm{Df}\left(\mathbf{y}^{(i)} \mid \theta\right)\right| \end{aligned}$$

During training, the parameters of the flow (θ) and of the base distribution (ϕ) are adjusted to maximize the log-likelihood.

3.5.1 Linear flows

Linear mappings can express correlation between dimensions:

$$\mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$$

where $\mathbf{A}$ is an invertible matrix. Linear flows maps a family distribution to its own family, in other words, using linear flow on a Gaussian distribution outputs a Gaussian distribution. We consider the base distribution:

$$p_{\mathbf{Z}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}, \mu, \Sigma)$$

After applying the linear flow we obtain:

$$p_{\mathbf{Y}} = \mathcal{N}\left(\mathbf{y}, \mathbf{A}\mu + \mathbf{b}, \mathbf{A}^T \Sigma \mathbf{A}\right)$$

And the determinant of the Jacobian is given by $\det(A)$.

4 Data

The model being divided in a factor model and a stock model, two dataset were retrieved. The first one containing the factors, the second one a list of stocks in the S&P500. These data are provided on a daily basis. The factors are variables that should summarize the "state of the world".

1. **CFE VIX**: The VIX index, also known as the "fear gauge," measures market expectations of near-term volatility conveyed by S&P 500 stock index option prices.
2. **CBOE SKEW**: Measures the perceived risk of extreme negative market moves, reflecting tail risk in S&P 500 options.
3. **ML MOVE**: Tracks the implied volatility of U.S. Treasury bonds, indicating expected fluctuations in bond prices.
4. **RUSSELL 3000**: Represents the performance of the 3,000 largest U.S. companies, encompassing approximately 98% of the investable U.S. equity market.
5. **RUSSELL 3000 GROWTH**: Measures the performance of the growth segment within the Russell 3000, focusing on companies with higher growth potential.
6. **ISHARE SMALL CAP**: Tracks the performance of small-capitalization U.S. stocks, often represented by the iShares Russell 2000 ETF.
7. **ISHARE MID CAP**: Tracks the performance of mid-capitalization U.S. stocks, often represented by the iShares Russell Mid-Cap ETF.
8. **International baselines**: A list of national currencies against the American dollar, As we predict return of S&P500 companies.
9. **Commodities baselines**: A list of commodities etf, such as copper, gold, gas or agriculture index.

5 model

This project will follow the work of [3] and consider generative models to make return predictions. We have:

$$\int \left(\prod_{i=1}^{N_{T+1}} p\left(r_{i,T+1} \mid \mathbf{F_{T+1}}, \mathcal{F}_T\right) \right) p\left(\mathbf{F_{T+1}} \mid \mathcal{F}_T\right) d\mathbf{F_{T+1}} \tag{9}$$

where, coming from the factor analysis assumption, we have two conditionnal probabilities to evaluate.

Factor model The factor model aims at describing the likelihood of the factors at time $T+1$ given information up to T. The authors propose to ease the learning of the factor model by using the PCA in order to reduce the dimensionality. The mapping between the factors and their reduced representation is done using an auto-encoder. This auto-encoder is simply defined as a succession of feed forward networks reducing the dimension of the input to a latent dimension, then a feed-forward networks extending it back to the original dimension. In particular, the feed-forward is composed of multi-layer perceptron with:

$$z = \text{Sigmoid}(f(x; \theta_f))x = g(y; \theta_g)$$

where f ins the encoder defined by the parameters θ_f and passed through a Sigmoid function to ensure that the latent representation is $\in [0, 1]$. This is convenient as it will serve as input for the variational auto-encoder.

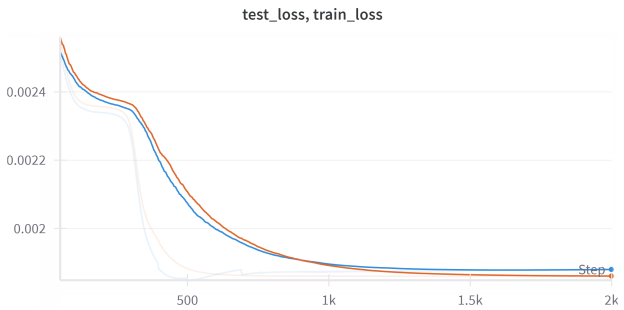

Figure 7: Auto-encoder loss

Moreover, the factor model uses the historical factors in order to predict $T+1$ factor values. Therefore, the model is learning:

$$p\left(\hat{\mathbf{F}}_{\mathrm{T}+1} \mid \left\{\hat{\mathbf{F}}_{\mathrm{t}}\right\}_{t=1}^{T}\right) \tag{10}$$

The network used to model the data:

1. The historical data $\{\hat{\mathbf{F}}_{\mathrm{t}}\}_{t=1}^{T}$ is passed through a LSTM network. The network is composed of a one-layer multi-layer perceptron, and two LSTM layers.

$$\{\mathbf{h}_{1,t}\}_{t=1}^{T} = NN_{FC}\left(\{\hat{\mathbf{F}}_{\mathrm{t}}\}_{t=1}^{T}\right)$$
$$\mathbf{h}_{2,T} = NN_{LSTM}\left(\{\mathbf{h}_{1,t}\}_{t=1}^{T}\right)$$

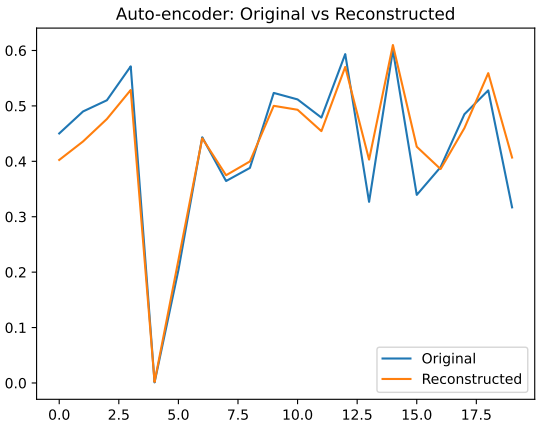

Figure 8: Example of reconstruction using the trained auto-encoder. The original input is compressed to a latent space $\Re^3 \in [0, 1]$

2. Historical information passed through the LSTM and the $T + 1$ information are used to parametrize the variational auto-encoder $q(\mathbf{z} \mid \hat{\mathbf{F}}_{\mathbf{T+1}}, \{\hat{\mathbf{F}}_{\mathbf{t}}\}_{t=1}^{T})$. The encoder is defined by a function f of the parameters θ, and is a neural network composed of two successive multi-layer perceptron, and two distinct multi-layer perceptron two output the mean and the variance of the variational family.

$$\mathbf{h}_{3,\mathbf{t}} = f_\phi(\hat{\mathbf{F}}_{\mathbf{T+1}} \| \underbrace{\mathbf{h}_{2,T}}_{\text{History}})$$
$$\mu_\eta = NN_\eta(\mathbf{h}_{3,T})$$
$$\sigma_\zeta = SoftPlus(NN_\zeta(\mathbf{h}_{3,T}))$$

We will then be able to sample data from the distribution learnt.

$$z \sim \mathcal{N}(\mu_\eta, \sigma_\zeta)$$

3. The decoder is define by a neural networks as well, parametrized by parameters θ. It takes as input z the sample from the learnt distribution, and the output of the LSTM network. Processing the same way as for the encoder, it will produce samples for the original distribution.

Figure 9: Factor VAE Loss

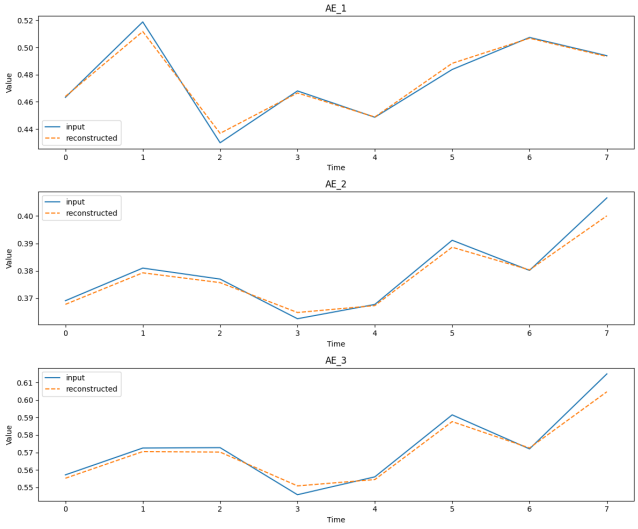

Figure 10: Example of encoding of one test data sample using the trained model. X-axis is the look-back period. The last x being the prediction, blue for true value, orange for prediction.

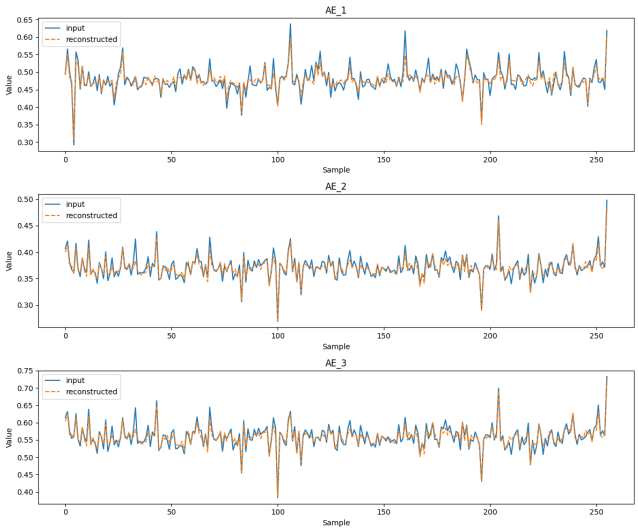

Figure 11: Example of prediction of the factors for the next periods for a series of samples, on the train data.

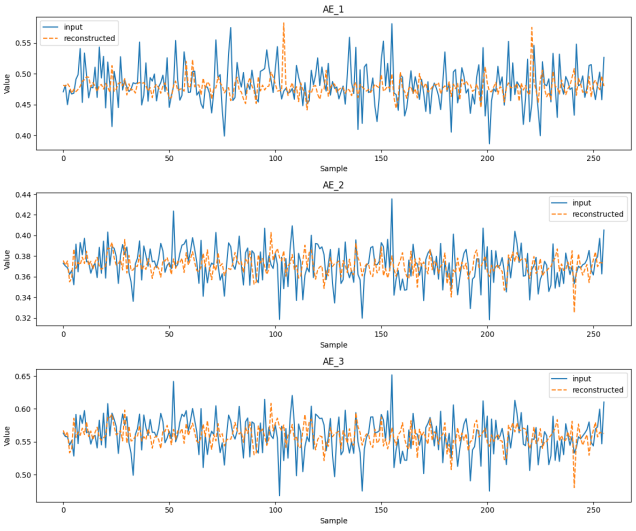

Figure 12: Example of prediction of the factors for the next periods for a series of samples, on the test data.

Has can be seen on *fig.*11 and *fig.*12, the model tends to capture the training data distribution pretty well but have more difficulties generalizing to estimate the original data distribution, providing worse results on the test set.

Stock model The stock model is defined as:

$$p\left(r_{\mathbf{i},\mathbf{T}+1} \mid \hat{\mathbf{F}}_{\mathrm{T}+1}, \{r_{i,t}\}_{t=1}^{T}\right)$$

In other word, we are computing the probability of the stock return a $T+1$ given our previous prediction using the factor model and the historical returns. The model is composed of 15 layers of auto-regressive transforms and a Gaussian prior distribution.

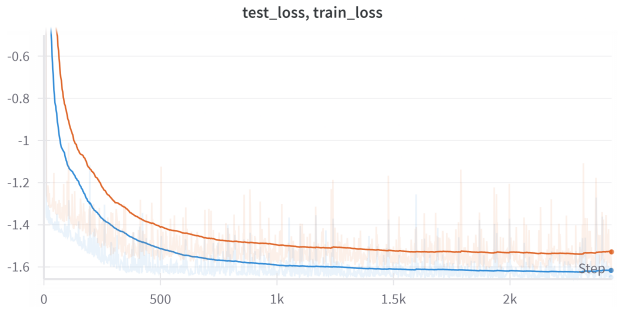

Figure 13: StockFlow loss

As we parametrize a probability density function given a context, we are able to make prediction by computing the mean of the distribution given this particular context, as well as a confidence interval, computed using the standard deviation.

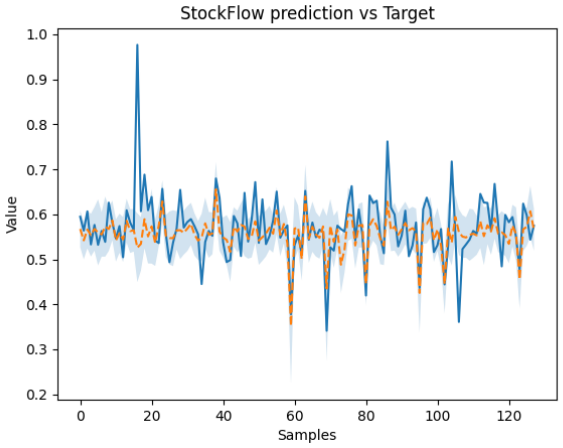

Figure 14: Prediction of a stock return for 128 samples on the training set, given its context and stock return history

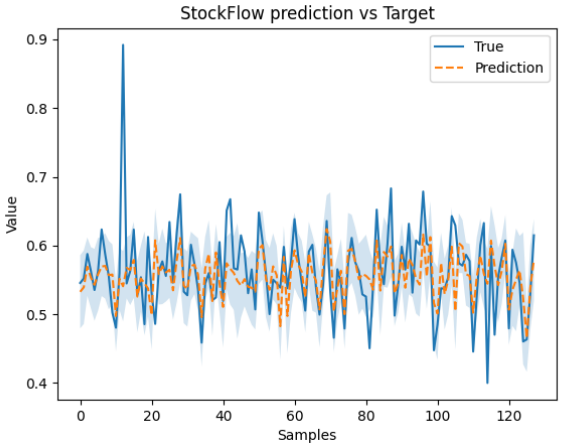

Figure 15: Prediction of a stock return for 128 samples on the test set, given its context and stock return history

Usage We can now make predictions using historical factors returns and stock returns.

1. Sample $\hat{\mathbf{F}}'_{\mathbf{T}+1}$ from the factor model
2. Sample returns for stocks $\left\{r'_{i,T+1}\right\}_{i=1}^{N_{T+1}}$, given the previously computed sample

Therefore, this model allows to make predictions for a future horizon of one period. Longer horizon predictions can be achieved by making auto-regressive prediction up to the horizon of interest.

5.1 Baselines models

Several baselines models have been implemented to compare the performances of our model. The three models used are straightforward models that are often used to predict mean return in order to apply a mean-variance portfolio optimization: linear regression, exponential moving average, decision tree regressor. The different models provide these results on an equivalent sample:

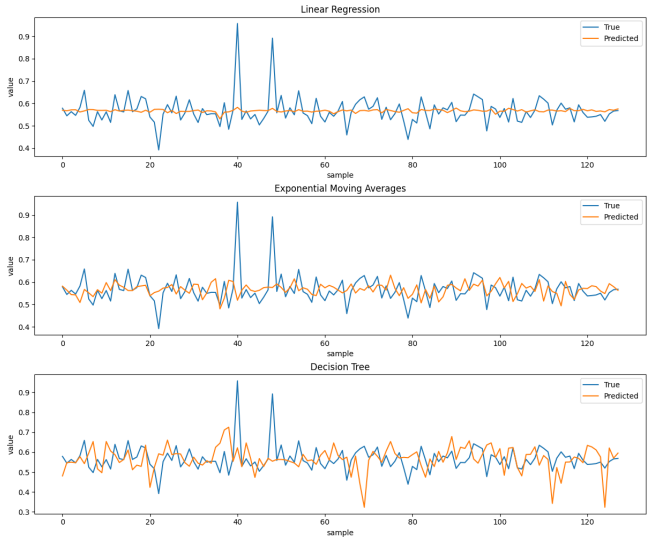

Figure 16: Example of predictions using the baselines models on 128 samples.

6 Architecture & Code handling

The project architecture consist in three mains files, one for the auto-encoder, one for the factor variational auto-encoder, and one for the stock normalizing flow. Each of these files contain all the necessary classes to model, train and load the data. Moreover, auxiliary files are used to define parameters for the different models and preprocess the data.

7 Conclusion

Various assessment measure have been used to asses the model performances by comparing to the baselines models. The coefficient of correlation, or R^2, is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{11}$$

measures the proportion of the variance in the dependent variables. The mean square error is given by:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{12}$$

And the mean absolute error:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{13}$$

The model provide the following results compared to the other baselines models:

model	R^2	MSE	MAE
Linear regression	-0.0005	0.004	0.044
Exp. moving average	-0.179	0.005	0.048
Decision tree reg.	-1.056	0.008	0.064
StockFlow	**0.3052**	**0.002**	**0.033**

Table 1: Comparison of models evaluation metrics

Therefore, the *StockFlow* model provides better results and propose an improvement to the baselines models.
Moreover, it is possible to explore a lot a different ways to use and expend them:

- This model with the found hyper-parameters have shown to have no difficulties to capture the training distribution but provide less efficient results on test data. Thus, some generalization technique could be tried: adding dropout, reducing the complexity of the model, etc..
- Other networks type could be implemented within each model, such as attention layers or recurrent layers inside the VAE.
- Explore different generative models such as diffusion models.
- Changing the context given to the conditionnal probability distribution, such as changing the factors, and/or adding security specific context like the firm financial data.
- Compare auto-regressive versus multi-period forecasting

Copilot and ChatGPT

These auxiliary tools were employed for coding, each in its unique way. Copilot was utilized for end-of-line completion to boost coding speed, while ChatGPT was used for debugging and improve prose.

References

[1] Bryan T. Kelly, Seth Pruitt, and Yinan Su. "Characteristics are covariances: A unified model of risk and return". In: *Journal of Financial Economics* 134.3 (2019), pp. 501–524. ISSN: 0304-405X (page 1).

[2] Shihao Gu, Bryan Kelly, and Dacheng Xiu. "Autoencoder asset pricing models". In: *Journal of Econometrics* 222.1, Part B (2021). Annals Issue: Financial Econometrics in the Age of the Digital Economy, pp. 429–450. ISSN: 0304-4076 (page 1).

[3] Ruslan Tepelyan and Achintya Gopal. "Generative Machine Learning for Multivariate Equity Returns". In: *4th ACM International Conference on AI in Finance*. ICAIF '23. ACM, Nov. 2023. DOI: 10.1145/3604237.3626884. URL: http://dx.doi.org/10.1145/3604237.3626884 (pages 1, 6).

[4] Gilles Louppe. "INFO8010 Deep Learning, Auto-encoders and generative models". In: (2023). URL: https://glouppe.github.io/teaching/.

[5] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. "Dive into Deep Learning". In: *CoRR* abs/2106.11342 (2021). arXiv: 2106.11342. URL: https://arxiv.org/abs/2106.11342.

[6] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11 (Nov. 2021), pp. 3964–3979. ISSN: 1939-3539. DOI: 10.1109/tpami.2020.2992934. URL: http://dx.doi.org/10.1109/TPAMI.2020.2992934.

[7] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. *Normalizing Flows for Probabilistic Modeling and Inference*. 2021. arXiv: 1912.02762 [stat.ML] (page 5).

[8] Ruslan Salakhutdinov Yuri Burda Roger Grosse. "Importance Weighted Autoencoders". In: (2016). URL: https://arxiv.org/abs/1509.00519.

[9] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. *nflows: normalizing flows in PyTorch*. Version v0.14. Nov. 2020. DOI: 10.5281/zenodo.4296287. URL: https://doi.org/10.5281/zenodo.4296287.

[10] Ricky T. Q. Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. *Residual Flows for Invertible Generative Modeling*. 2020. arXiv: 1906.02735 [stat.ML].

8 Appendices

8.1 Change of variable

Suppose X is continuous with probability density function $f_X(x)$. Let $y = h(x)$ with h a strictly increasing continuously differentiable function with inverse $x = g(y)$. Then $Y = h(X)$ is continuous with probability density function $f_Y(y)$ given by:

$$f_Y(y) = f_X(g(y))g'(y)$$