Applying Computer Vision techniques to improve stock market prediction

Adrien Lagesse
École Polytechnique, BNP Paribas

Research internship report - 2022

Department: Applied Mathematics Center
Referent (École Polytechnique): Stefano De Marco
Tutor (BNP Paribas): Romain Rousseau
Internship dates: March, 2022 - September, 2022

Maison Dorée,
20 Boulevard des Italiens, 75009,
Paris, France

Integrity statement on plagiarism

I, the undersigned, Adrien LAGESSE certify on my honour :

1. That the results described in this report are the outcome of my work.
2. That I am the author of this report.
3. I have not used third-party sources or results without clearly citing and referencing them according to the recommended bibliographic rules.

I declare that this work cannot be suspected of plagiarism.

Paris, 26 August 2022 Signature: *Jean Adrien Lagesse*

Acknowledgments

I would like to thank Romain Rousseau and Ilan Dahan, who helped me, guided me in my research and introduced me to the world of algorithmic trading, I learned a lot about the stock market and the different ways to extract *alpha* from the multitude of signals available.

I would also like to thank all the team at BNP Paribas for their precious help. The technical team really helped my to understand the different tools that are available to develop strategies and indicators as well as understanding the different sources of information that are available.

Abstract

Linear models have a long standing in algorithmic trading, they are easy to implement, inference is nearly instantaneous and they rarely over-fit. Nevertheless, these models have one main drawback: they are incapable of finding non-linear relationship without tedious and sub-optimal feature engineering. The expressivity of neural networks makes them prime candidates to solve this problem but they often generalise very poorly to new samples (*i.e* that are not in the training data). During my internship at BNP Paribas, I implemented techniques that were firstly introduced in Computer Vision to solve this problem. On one hand, we show that learning meaningful representations with Variational Auto-Encoders increases the compression rate of information while preserving a highly structured and continuous space (compared to the PCA). On the other hand, using this representation of the market instead of the raw features improves the prediction score of our indicators.

More concretely, we present a method to reduce by 30% the error rate compared to the PCA algorithm while maintaining a latent space of high quality. Moreover, using the deep neural network architecture used here increases the explained variance of the stock market by 19% compared to linear methods.

Résumé

Les modèles linéaires ont une longue réputation dans le trading algorithmique, ils sont faciles à mettre en œuvre, l'inférence est presque instantanée et ils sont rarement surajustés. Néanmoins, ces modèles ont un inconvénient majeur : ils sont incapables de trouver des relations non linéaires sans *feature engineering*; processus fastidieux et sous-optimal. L'expressivité des réseaux neuronaux en fait des candidats de choix pour résoudre ce problème, mais ils se généralisent souvent très mal à de nouveaux échantillons (*i.e* qui ne font pas partie des données d'entraînement). Pendant mon stage chez BNP Paribas, j'ai mis en œuvre des techniques qui ont été introduites pour la première fois dans le domaine de la vision par ordinateur pour résoudre ce problème. D'une part, nous montrons que l'apprentissage de représentations avec des Auto-Encodeurs Variationnels augmente le taux de compression de l'information tout en préservant un espace hautement structuré et continu (par rapport à l'ACP). D'autre part, l'utilisation de cette représentation du marché au lieu des caractéristiques brutes améliore le score de prédiction de nos indicateurs.

Plus concrètement, nous présentons une méthode permettant de réduire de 30% le taux d'erreur par rapport à l'algorithme d'ACP tout en maintenant un espace latent de haute qualité. De plus, l'utilisation de l'architecture de réseau de neurones profonds utilisée ici augmente de 19% la variance expliquée du marché boursier par rapport aux méthodes linéaires.

Contents

Introduction

Predicting stock prices is a notoriously hard problem that banks and hedge-funds are trying to tackle. Traditionally, linear forecasting methods such as the ARIMA model (Mondal et al. [10]) and principle components (Ghorbani et al. [5]) were the most performant statistical techniques. However, more recently, with the recent advances that happened in deep learning, researchers have been successfully applying more expressive methods such as Recurrent Neural Networks, LSTM (Zou et al. [15]) and Deep Q-Learning (Jeong et al. [8]) to predict the stock market. Most of these methods were developed as auto-regressive predictors, using the information contained in the *path* of the time-series to predict its future, nonetheless, another approach consists of using the mutual information between the different features (i.e different stocks) to make a prediction.

In this paper, predictors are based on the mutual information between stocks. We use a deep neural network approach and bring techniques inspired by the recent breakthroughs

that happened in Computer Vision and Representation Learning to predict stock returns more accurately. Indeed, in the last few years, Deep Neural Networks became the state-of-the-art of nearly all Computer Vision tasks. These breakthroughs were made thanks to new neural network architectures such as Convolutional Neural Network (Bengio et al. [3]) to detect local features in images to the invention of Variational Auto-Encoder (Kingma et al. [9]) to create representations in a continuous latent space. The same architectures are adapted in this paper to increase the performance of our predictors.

Modelisation of the problem

To model our problem we will use a probabilistic point of view as it is usually used in modern theories. Let $\mathcal{M} = (R_1(t), ..., R_N(t))_{0 \leq t \leq T}$ be a stationary stochastic process on the probability space $(\Omega = \mathbb{R}^N, \mathcal{B}(\mathbb{R})^N, \mathbb{P})$. Each R_i is a stochastic process representing the return of the stock $i \in \{0, ..., N\}$. The canonical filtration of $\mathcal{M}$ is defined as $\mathcal{F} = \sigma(\mathcal{M}_{t'},\ 0 \leq t' \leq t)_{0 \leq t \leq T}$. Moreover, to simplify the problem, we will also consider that $\mathcal{M}$ is a Markov process, hence, $\mathcal{F}_t = \mathcal{M}_t$ for $0 \leq t \leq T$. This hypothesis enable us to *forget* the history of the stock price, we consider that all the information of the market is contained in its current state. Finally, the stationarity of $\mathcal{M}$ implies that $(\mathcal{M}_t, \mathcal{M}_{t+\delta t})$ is identically distributed for $0 \leq t \leq T$, hence, two random variables $\widetilde{\mathcal{M}}$ and $\widetilde{\mathcal{M}}_\delta$ exist such that $(\mathcal{M}_t, \mathcal{M}_{t+\delta t}) \sim (\widetilde{\mathcal{M}}, \widetilde{\mathcal{M}}_\delta)$ for every t.

With these notations, the random variable $\widetilde{\mathcal{M}}_\delta | \widetilde{\mathcal{M}}$ contains all the information needed, it is however intractable to compute its probability law. As a first approximation, we are going to compute $\mathbb{E}(\widetilde{\mathcal{M}}_\delta | \widetilde{\mathcal{M}})$ which is the naive Bayes regressor. Considering the class of parametric functions $\{f_\theta \quad | \quad \theta \in \Theta\}$, this problem can be seen as solving the two following equivalent propositions:

$$\begin{aligned} &\underset{\theta \in \Theta}{\operatorname{argmin}} \quad \mathbb{E}\left[(f_\theta(\widetilde{\mathcal{M}}) - \mathbb{E}(\widetilde{\mathcal{M}}_\delta \mid \widetilde{\mathcal{M}}))^2 \right] \\ &\underset{\theta \in \Theta}{\operatorname{argmin}} \quad \mathbb{E}\left[(f_\theta(\widetilde{\mathcal{M}}) - \widetilde{\mathcal{M}}_\delta)^2 \right] \end{aligned} \tag{$\star$}$$

This model can be slightly modified to add information about the history of stock prices but it can only work on a fixed size time window. If we consider that $\mathcal{F}_t = \sigma(\mathcal{M}_t, \mathcal{M}_{t-\delta}, \ldots, \mathcal{M}_{t-(r-1)\times\delta})$ where $r \geq 1$ is the length of the time window, the stationary property of $\mathcal{M}$ assures us that there exists $\widetilde{\mathcal{M}}_{-(r-1)\times\delta}, \ldots, \widetilde{\mathcal{M}}$ and $\widetilde{\mathcal{M}}_\delta$ exist such that $(\mathcal{M}_{t-(r-1)\times\delta}, \ldots, \mathcal{M}_t, \mathcal{M}_{t+\delta t}) \sim (\widetilde{\mathcal{M}}_{-(r-1)\times\delta}, \ldots, \widetilde{\mathcal{M}}, \widetilde{\mathcal{M}}_\delta)$ for every t. In this case, the optimisation problem is:

$$\begin{aligned} &\underset{\theta \in \Theta}{\operatorname{argmin}} \quad \mathbb{E}\left[(f_\theta(\widetilde{\mathcal{M}}_{-(r-1)\times\delta}, ..., \widetilde{\mathcal{M}}) - \mathbb{E}(\widetilde{\mathcal{M}}_\delta \mid \widetilde{\mathcal{M}}))^2 \right] \\ &\underset{\theta \in \Theta}{\operatorname{argmin}} \quad \mathbb{E}\left[(f_\theta(\widetilde{\mathcal{M}}_{-(r-1)\times\delta}, ..., \widetilde{\mathcal{M}}) - \widetilde{\mathcal{M}}_\delta)^2 \right] \end{aligned} \tag{$\star\star$}$$

Chapter 1

Linear models, regularisation and dimension reduction

Linear models are a simple yet very practical to extract relevant information from the market. Even if the class of linear functions is sparse, in very high dimensions it can easily over-fit the data and in this chapter, we test several variants of the classical linear regression algorithm to establish a benchmark on our dataset. We also analyse the linear relationships between stocks to have a deeper understanding of the market and quantify its sensibility to some key stocks. This primary work gives a good intuition of the statistical structure of the market that will be necessary when working with complex non-linear models.

1.1 Linear regression and over-fitting

Let $\mathcal{C}_{linear} = \{f_\theta : x \mapsto \theta x | \theta \in \Theta\}$ where $\Theta = \mathcal{M}_{N\times N}(\mathbb{R})$ be the class of linear predictors. The solution of $(\star)$ on the class $\mathcal{C}_{linear}$ is the best linear approximation of $\widetilde{\mathcal{M}}_\delta$ given $\widetilde{\mathcal{M}}_\delta$. In practice, we don't have access to the laws of $\widetilde{\mathcal{M}}$ and $\widetilde{\mathcal{M}}_\delta$, instead we have M samples $(\widetilde{\mathcal{M}}^1, \widetilde{\mathcal{M}}^1_\delta), \dots, (\widetilde{\mathcal{M}}^M, \widetilde{\mathcal{M}}^M_\delta)$. An approximate solution of $(\star)$ is given by:

$$\underset{\theta\in\Theta}{\operatorname{argmin}} \quad \sum_{i=1}^{M} (f_\theta(\widetilde{\mathcal{M}}^i) - \widetilde{\mathcal{M}}^i_\delta)^2 \qquad (\star\star\star)$$

In the case of linear predictors, solving $(\star\star\star)$ can be done analytically, however, this method is very rigid. We will rather use gradient descent techniques,they are more adaptable and give a better insight on how the model learns the existing relationships in the data. Moreover, even if in the general case there is no guaranty, because $(\star\star\star)$ is a convex problem, we are assured that gradient descent will converge to the unique solution.

However, even a sparse class like $\mathcal{C}_{linear}$ can produce a solution that over-fit on M samples. This phenomenon is a direct consequence of the *curse of dimensionality* (Bellman [2]), this term describes the fact that as the dimension of the feature of a random variable grows, the number of points to accurately describe this random variable exponentially explodes. In the case of stock prediction, it is very important to work with hundreds of stocks because each stock only brings a very small amount of information.

When working on the European stock market, several hundreds stocks are traded on a daily basis on many exchanges. When fitting a linear regression to predict the price of stocks in the next minute, we are confronted to the problem cited above: the regressor explains very well the training data but fails to generalise to new data points.

In the following experiments (all experiments in the paper), the dataset is composed of 472 stocks, with historical one minute returns. 140 days trading days are used as the training dataset and 60 days are used in the validation dataset. There is $\approx 100,000$ samples in the dataset depicting a complete picture of the European stock market between 2019, 2020 and 2021.

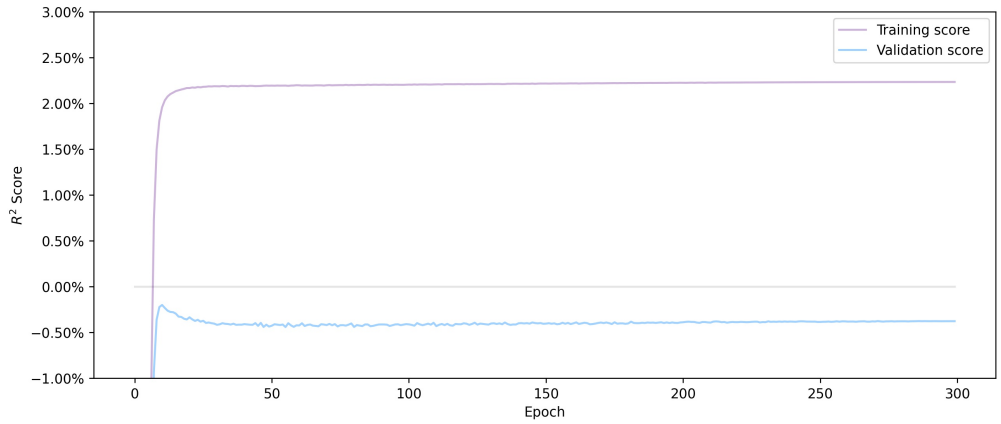


Figure 1.1: As seen in this figure, during training, the linear regression learns to model the training dataset very well but fails to understand general statistical relationships in the stock market. In the beginning, the linear regressor learns things that are true in the validation set but then it prioritize noisy information in the training dataset that hurts the performance in the validation dataset.

1.2 A variant of the linear regression model: Elastic-Net

To solve this over-fitting problem, Elastic-net (Zou et al. [14]) was introduced. It incorporates ideas of the Lasso and Ridge regression to regularize the model and to select important variables only. The parametric class remains unchanged, but, the optimisation problem is slightly modified. Two new terms are added. Let $\lambda_1 > 0$ and $\lambda_2 > 0$ in the modified $(\star)$ problem:

$$\underset{\theta \in \Theta}{\operatorname{argmin}} \, \mathbb{E}\left[(f_\theta(\widetilde{\mathcal{M}}) - \widetilde{\mathcal{M}}_\delta)^2 \right] + \lambda_1 ||\theta||_1 + \lambda_2 ||\theta||_2 \tag{1.1}$$

This forces the parameters to be smaller and the level of that force is set by λ_1 and λ_2. The effect of each one of those parameters are to reduce over-fitting but they act in a different way. The L^1 norm makes the model choose a few features while "forgetting" the others resulting in a sparser model. The L^2 norm finds relationships between all the features. Applying

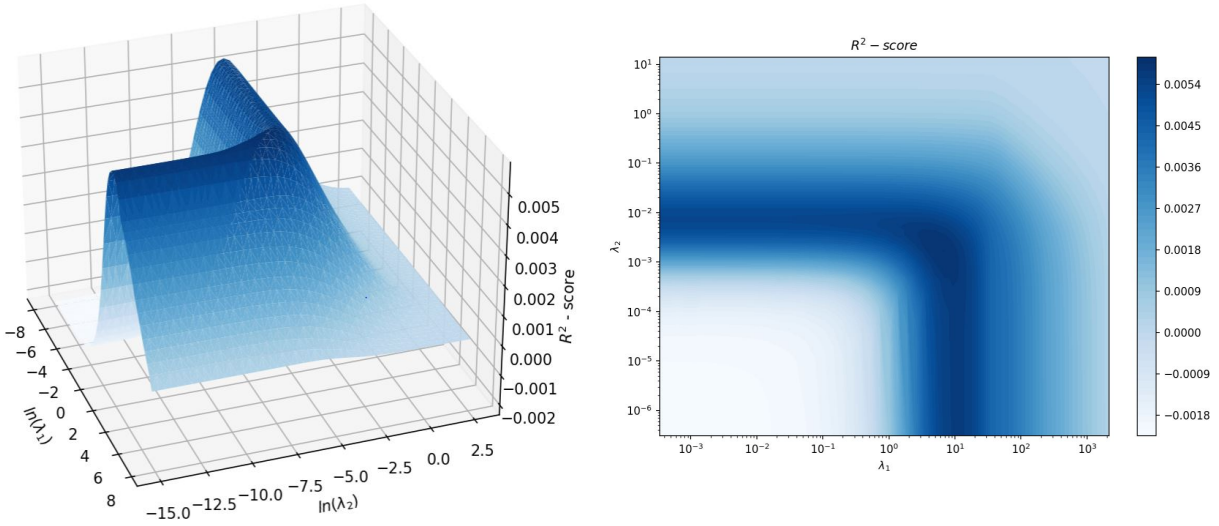


Figure 1.2: R^2-score of Elastic-Net with different regularization parameters

those modifications have a huge effect on the results of the linear regression. The resulting predictor is generalising very well to new data-points, meaning that meaningful linear relationships are found in the data. However, the choice of λ_1 and λ_2 is very important and have a big impact on the performances as shown in 1.2. If these hyper-parameters are to small the model will over-fit, however, if they are to big, the model will only not be able to find all the relationships in the data (minimizing the L^1 and L^2 loss will be more important than minimizing the prediction loss).

$\longrightarrow$ *The results of Elastic-Net will be used as a benchmark in the rest of the paper. When using non-linear methods such as Neural Networks, just using L^1 and L^2 regularisation will not work and the rest of the paper will present ways to circumvent this problem.*

1.3 Dimension reduction and PCA

As the results of Elastic-Net prove, predictive linear relationships exist on the stock market. Yet, the amount of information is very small compared to the noise. Another easier problem, gives useful insights on the stock market to build a good intuition and simplify the prediction problem. Instead of analyzing the predictive relationships, we can learn the static information between different stocks. With the Principal Components Analysis algorithm (Abdi et al. [1]), it is possible to learns these linear relationships existing in the stock market. This algorithm can then be used in several different ways: to reduce the dimension of the feature space, to create a different representation of the data or to help us understand some key elements of the market structure.

Theorem 1.1 (Principal Component Analysis). — *Let $X = (X_1, ..., X_N)$ be a random variable on $\mathbb{R}^N$, there exist a bijective orthogonal matrix P such that $XP = (\tilde{X}_1, ..., \tilde{X}_N)$ where $\mathbb{C}\text{ov}(\tilde{X}_i, \tilde{X}_j) = 0$ for all $i \neq j$ and $\mathbb{V}\text{ar}(\tilde{X}_1) \geq \cdots \geq \mathbb{V}\text{ar}(\tilde{X}_N)$. The columns of P are*

the principal components of X.

As an orthogonal matrix, P can be seen as a change of basis. The new *space* have useful properties: the coordinate are linearly independent and the information contained in each coordinate is decreasing (the first coordinates explain well X). This structure on the new space gives a canonical way of reducing the dimension of the feature space, choosing only the first $N^* < N$ coordinates assure us that the most information is retained and that only the less meaningful coordinates are dropped.

In practice, it is very easy to calculate P for relatively small datasets. An analytical method (Abdi et al. [1]) exists to compute the matrix P and also the variance of $\tilde{X}_i$ for all i. Let $X = (X_{i,j})_{i,j \in \{1,..,M\} \times \{1,...,N\}} \in \mathbb{M}^{M \times N}$ represents M observations of N features.The empirical covariance matrix is equal to $C = \frac{1}{M-1} X^T X$, this matrix is a symmetric matrix so it can be decomposed as $C = P^T D P = P^{-1} D P$ where P is an orthogonal matrix and D is a diagonal and positive matrix. The columns of P are the principal components and $\mathbb{V}\text{ar}(\tilde{X}_i) = D_{i,i}$.

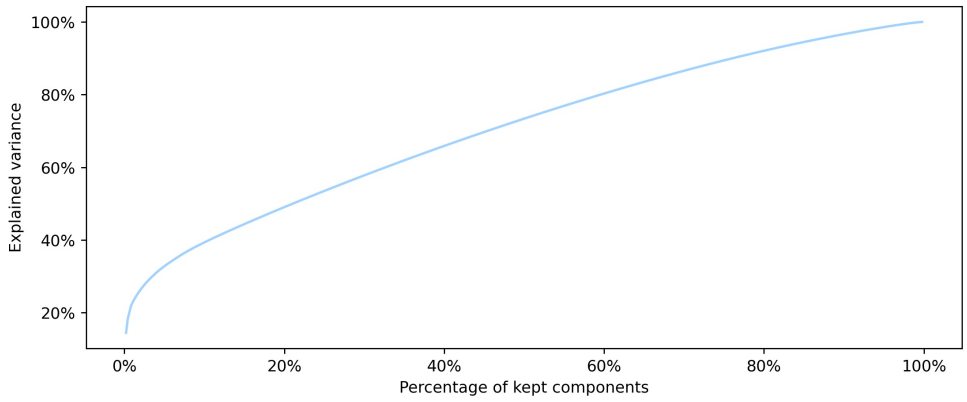


Figure 1.3: Cumulative explained variance of the PCA on the stock market.

When applying PCA on the stock market, a few principal components are able to explain a lot of the information but if we want retain a lot of information, we will still need many principal components. In 1.3, we can see that 5% of the components explains 30% of the variance but if we want to retain 80% of the information we need 75% of the components. This is a consequence of the high amount of noise in the stock market, most of the variance is due to uncorrelated noise.

Looking at the composition of the principal components gives an understanding of the statistical structure of the stock market. The first principal components encode a lot of information and looking at their composition is also very useful to understand what stocks drive the market. On one hand, the stocks that have a high weight in the firsts principal components will contain information that is useful to describe all the market, on the other hand, stocks with less weight in the firsts components and have high weights in the last components are very well explained by other stocks but doesn't bring a lot of information on other stocks.

The PCA matrix 1.4 shows that there are two kinds of principal components, the first columns contain a lot of information and are composed of significantly many stocks, we can suppose that they act as an *index*, representing a general state of the market. The rest of the matrix is stratified, each component contains information on a few stocks only. As stated before, the order in which the stocks appear is linked to the amount of information that they explain.

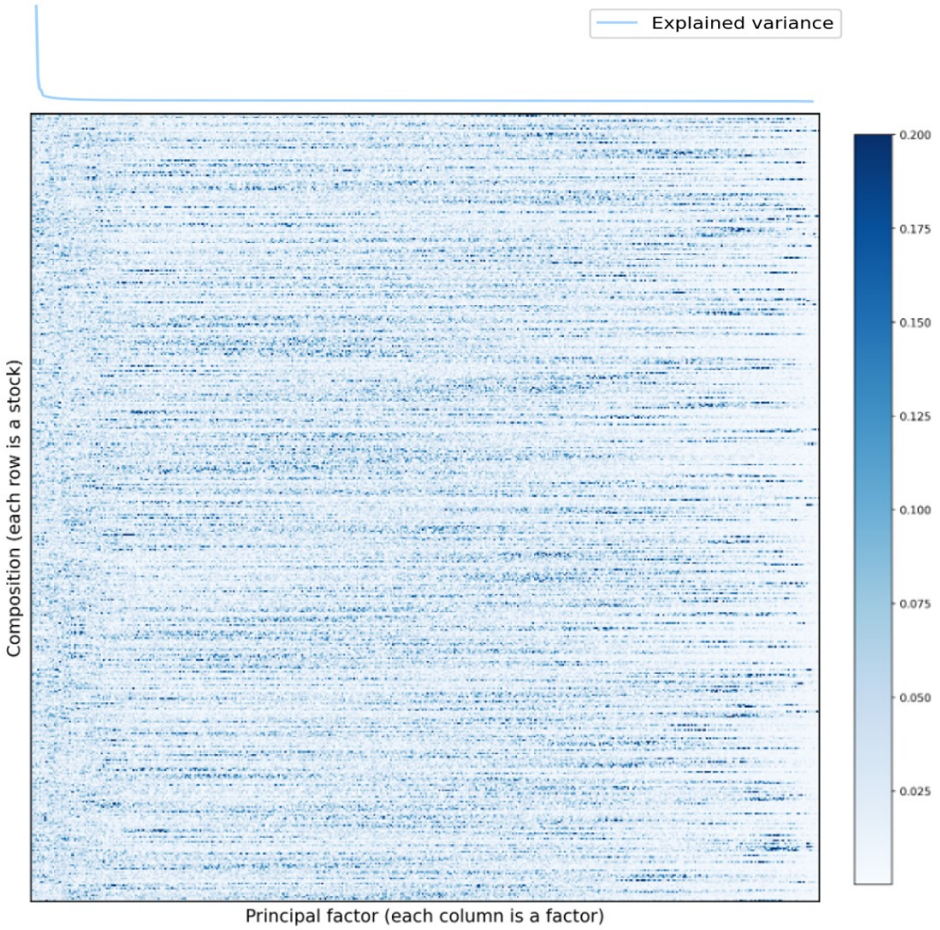


Figure 1.4: Visualisation of the matrix P, each column of this matrix represents the composition of a principal component with on top the explained variance of this component.

In the case of the European stock market, 1.5 summarises the most important stocks in the first components. We can see that many of these stocks are from big companies such as *BNP Paribas*, *GSK*, *Société Générale* and *Diageo*. Moreover, the banking sector is very well represented, and the countries that are represented are mainly the UK and France. Hence, we can assume that the banking sector represents well the global trend of the market and that some countries have a bigger impact than others on the market.
Because the PCA have this property of sorting the useful information and gives us a way to reduce the dimension of the problem, it can be seen as a representation learning technique and a manifold learning method to. This means that the PCA is able to extract a new representation of the data by projecting it on a lower dimension linear plan. The power of the manifold learning lies in the fact that even if the data lives in a N-dimensional space, it is possible that the data have a lower intrinsic dimension N^*. The projection on the manifold

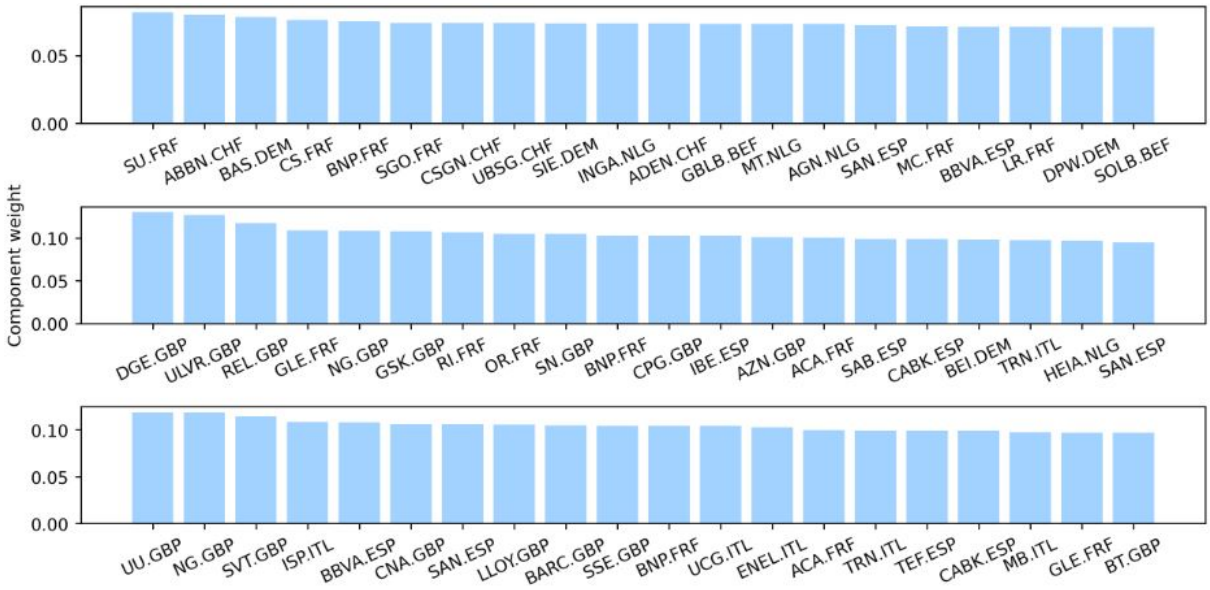
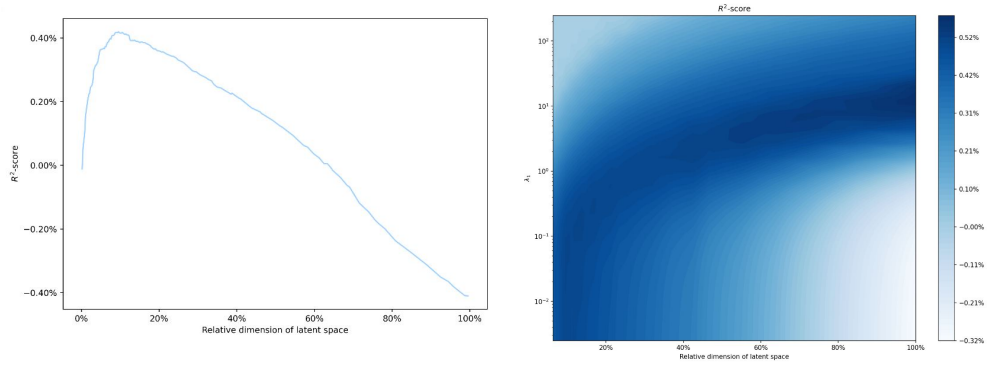


Figure 1.5: Top ten most weighted stocks for the three first principal components (from top to bottom)

is forgetting all these useless dimensions, giving a more compact representation of the data. In the particular case of the PCA, the manifold is restricted to N^*-dimensional plans. Hence the PCA is only able to project data in a linear subspace of the universe.

Working with the representation given by the PCA is very useful to solve ($\star$), working in a lower dimension reduce the risk of over-fitting the data and gives more predictor that generalise well to new data points. Moreover, the PCA has the great property to forget unimportant information, information that is useless to explain the current state of the market.



(a) Reduced dimension regularization only

(b) Reduced dimension and L^1 regularization

Figure 1.6: R^2-score of a linear model with dimension reduction

However, in practice, a good balance must be found between reducing to much the dimension of the space and keeping useless components. In 1.6, the results of incorporating the PCA in the predicator are shown.

1.4 Limitations

As seen previously, a simple linear regression cannot learn on its own, the relationship between $\widetilde{\mathcal{M}}$ and $\widetilde{\mathcal{M}}_\delta$ because of the high dimensionality of the data. Regularization techniques and dimensionality reduction with PCA improve the performances but they assume a big prior on the structure of the data and the model:

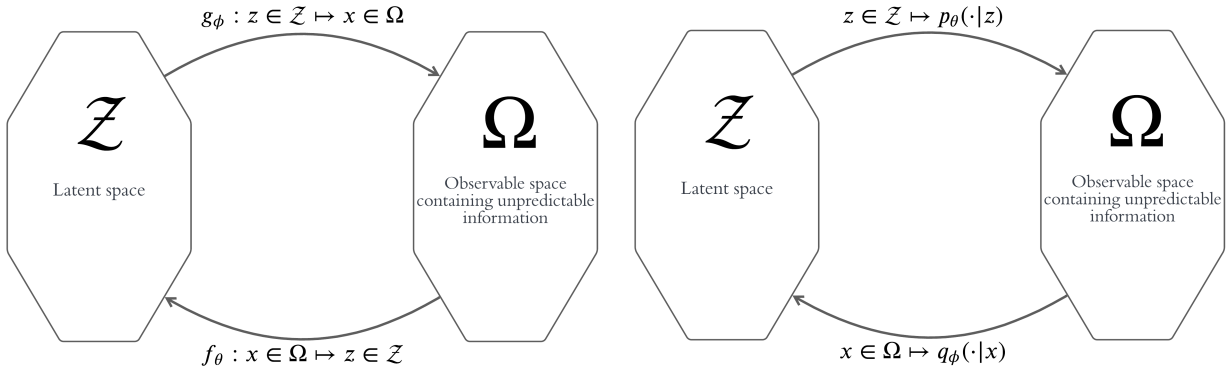
1. L^1 and L^2 have no prior meaning, they just artificially limit the size of the weights to have a smoother function
2. the PCA assumes that $\mathcal{M}$ lies in a linear subspace of Ω.

In the next chapters we will see how we can implement other ways to control the model and prevent it from over-fitting while also capturing non-linear information. Deep learning techniques developed for Computer Vision can be applied to the stock market to learn more expressive models that generalise well to new data points and enable better prediction of stock returns.

Chapter 2

Bayesian inference and structured latent space

Linear models augmented with the methods presented in *Chapter 1* are not able to use the non-linear relationships that exist between the different stocks. To solve this problem, neural networks are well adapted but they tend to over-fit more easily than a linear regression. With only L^1 and L^2 regularization techniques the network is not able to perform well but using the ideas introduced by the PCA it is possible to find meaningful non-linear information in the data. By considering that the observable event of Ω is just the manifestation of a more tractable event in another space, it is possible to create a representation of each state of the market that only contains the necessary information without noise. From this latent representation, predicting the next state of the market is easier and neural networks can be used. This chapter describes how to create this latent space and how to learn good representations. To be able to solve this problem, we will study two types of neural networks described in 2.1 that are massively used in Computer Vision and can be adapted to the stock market.



(a) Auto-encoder

(b) Variational auto-encoder

Figure 2.1: Relationships between observable space and latent space

2.1 Auto-encoders: a non-linear generalization of the PCA

Auto-encoders (Fig 2.1a) are a special type of neural networks that learn a mapping between the observable space Ω and the latent space $\mathcal{Z}$ by creating an information bottleneck in the network and then minimizing the reconstruction cost. This forces the network to choose what information to keep and what information to forget. This problem can be written as :

$$\underset{\theta,\phi}{\operatorname{argmin}} \quad \mathbb{E}(||\widetilde{M} - g_\phi \circ f_\theta(\widetilde{M})||^2) \tag{2.1}$$

In this equation, f_θ is called the encoder and g_ϕ is the decoder. The information bottleneck in this case can be build in several ways such as:

1. Forcing the dimension of $\mathcal{Z}$ to be smaller than the dimension of Ω, this will force the encoder to compress the data.

2. Forcing a criterion of sparsity on $\mathcal{Z}$ either by using the L^1 norm or the Kullback-Leibler divergence between the outputs of the encoder and a Bernoulli of parameter $p > 0$.

An interesting fact is that the PCA algorithm is closely linked to auto-encoders, indeed, the spaces created by the first $N^* < N$ columns of the projection matrix is the same space that the latent space of a linear auto-encoder with a bottleneck of dimension N^*. The power of an auto-encoder compared to the PCA is that it can learns non-linear mappings from Ω to $\mathcal{Z}$, this means that if the intrinsic surface of $\widetilde{\mathcal{M}}$ is not a plane, the auto-encoder will be able to model it whereas the PCA will approximate it by a plane, losing valuable information and adding noise in the representation. However, by having the possibility to extract new relationships in the data, the risk of over-fitting is also increased and solving (2.1) is harder. To solve these problems, techniques that were invented for Computer Vision can be applied and will be discussed in the following sections.

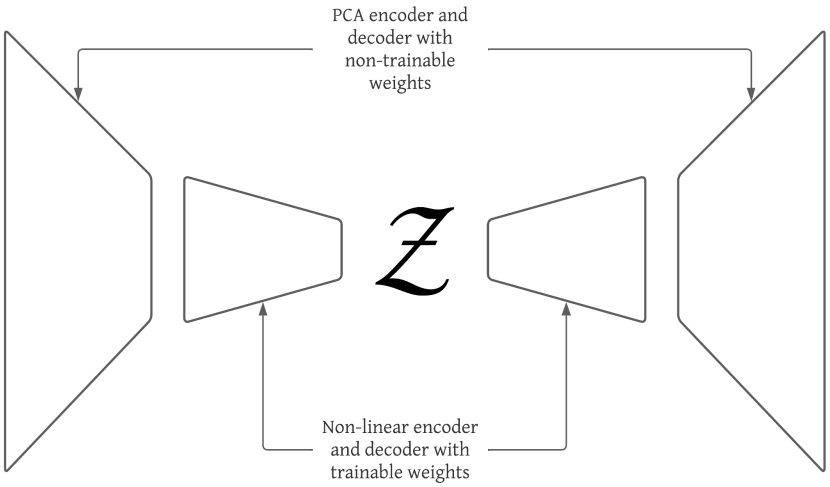


Figure 2.2: Auto-encoder architecture

2.2 Residual networks

As stated before, the problem of stock price prediction is particularly tough due to the high amount of noise, this means that it is very hard to extract meaningful non-linear information. Networks must be very deep to capture this information and this makes training such models harder. Not only having deeper networks increases the risk of over-fitting, it can also be very hard to find the optimal parameters (mode collapse). Having many layers in the network creates a well-known problem: vanishing and exploding gradients. This makes the training very unstable and even fitting training data can be very difficult. Having a lot of trainable parameters makes the loss surface very irregular and it is simple to be stuck in a local minima. The Residual network architecture solves this problem. It was introduced in Computer Vision to be able to have networks with tens of layers(the ResNet [6] network have 156 layers and won the Imagenet competition). The idea is to add skip connections to facilitate the propagation of the gradient through the different layers of the network as described in Fig 2.3. With such connections, the network can choose its own depth during training, trying to add more and more layers as the epochs pass by. This has the effect of training the network without collapsing due to gradient vanishing due to the depth of the layers.

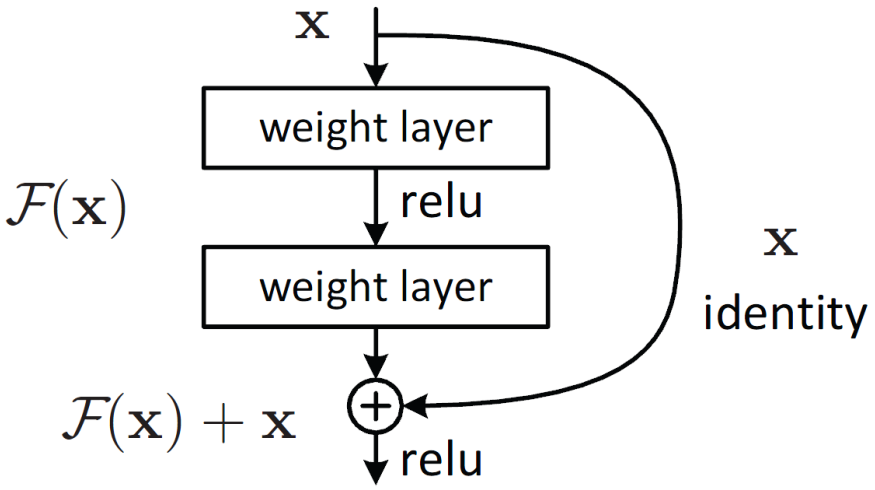
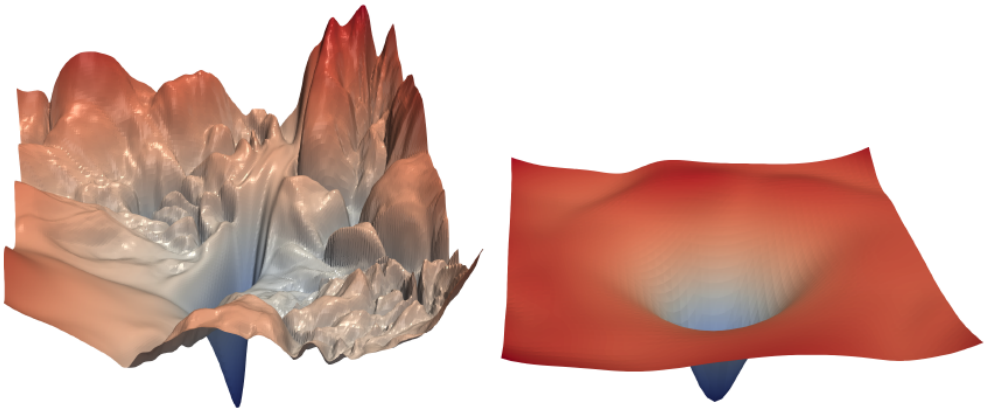


Figure 2.3: Skip connection in residual networks

Not only does it facilitate the propagation of the gradient in the network but the loss surface is also smoother (2.4). The risk of being stuck in a local minimum drastically decreases and the training of the network is a lot more stable.
By adding skip connections in the encoder and the decoder, it is possible to train very deep auto-encoders that are very powerful compared to PCA and can learn a better representation of the stock market.



(a) without skip connections (b) with skip connections

Figure 2.4: Loss surface

2.3 Variational auto-encoders

Auto-encoders are very powerful yet in their current formulation we have no control over the latent space structure which makes it very hard to use the representation for other tasks such as predicting the next state of the market. To solve this problem Variational Auto-encoders (Figure 2.1b) were introduced (Kingma et al. [9]). By using VAE's the encoder and decoder must learn a continuous latent space that is much more robust than a discontinuous one.

Variational Auto-encoders are auto-encoders that are placed in a probabilistic setting. Consider the probability space $(\mathcal{Z} \times \Omega, \mathcal{B}(\mathcal{Z} \times \Omega), \mathbb{P})$, in this space, the prior distribution is $p_{\mathcal{Z}} : z \mapsto p(z)$, the likelihood of $x \in \Omega$ given $z \in \mathcal{Z}$ is $p(x|z)$ and the posterior of $z \in \mathcal{Z}$ given $x \in \Omega$ is $p(z|x)$. Here $x \mapsto p(\cdot|x)$ is the probabilistic encoder mapping an observation to a distribution over the possible latent representations that can explain it and $z \mapsto p(\cdot|z)$ is the probabilistic encoder mapping a representation to a distribution over the possible observations. Let $x \mapsto q_\phi(\cdot|x)$ be a parametric model of the probabilistic encoder (in practice $q_\phi(\cdot|x) \sim N(\mu_\phi(x), \sigma_\phi(x))$.

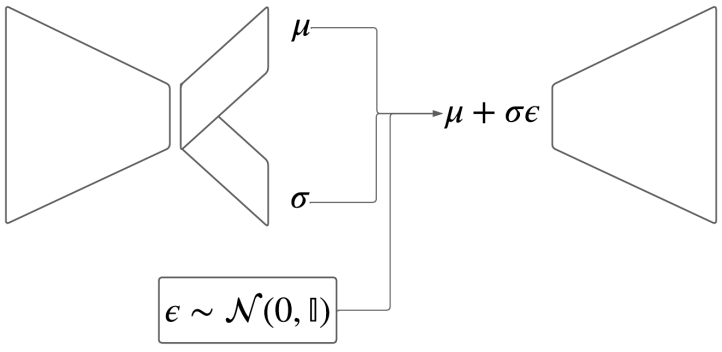


Figure 2.5: Arcitecture of a Variational Auto-encoder

Training a variational auto-encoder means finding the best probabilistic encoder, hence min-

imizing the Kullback-Leibler divergence between the parametric encoder and the true one:

$$\begin{aligned} KL\left[q_\phi(\cdot|x)||p(\cdot|x)\right] &= \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[log(q_\phi(z|x)) - log(\frac{p(x|z)p(z)}{p(x)})\right] \\ &= \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[log(q_\phi(z|x)) - log(p(x|z)) - log(p(z))\right] + Constant \\ &= -\mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[log(p(x|z))\right] + KL\left[q_\phi(\cdot|x)||p_{\mathcal{Z}}\right] + Constant \end{aligned} \tag{2.2}$$

In addition, by defining $z \mapsto p_\theta(\cdot|z)$ as the probabilist decoder, the VAE objective is:

$$\underset{\theta,\phi}{\text{argmin}} \quad -\mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[log(p_\theta(x|z))\right] + KL\left[q_\phi(\cdot|x)||p_{\mathcal{Z}}\right]$$

Finally, in practice for solving the problem $(\star)$, the following parametric models are used:

$$\begin{aligned} q_\phi(\cdot|x) &\sim N(\mu_\phi(x), \sigma_\phi(x)) \\ p_\theta(\cdot|z) &\sim N(\mu_\theta(z), \sigma_\theta(z)) \end{aligned}$$

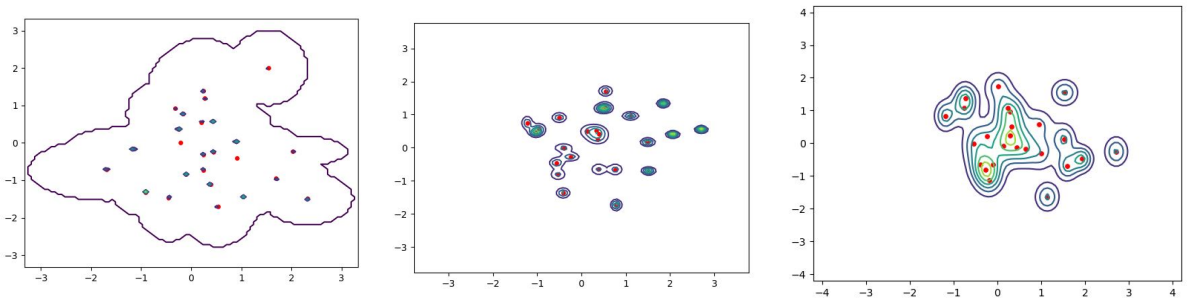
Hence, the loss is defined as:

$$\mathcal{L}(\theta, \phi; x) = ||x - \mu_\theta \circ \mu_\phi(x)||^2 + KL\left[\sigma_\phi(x)\mathcal{N}(0, \mathbb{I}) + \mu_\phi(x)||\mathcal{N}(0, \mathbb{I})\right]$$

This form of the loss is well-suited for gradient descent minimization.

To control the regularity of the latent space, we will use a slightly modified version of the VAE called the β-VAE (Higgins et al. [7]). A coefficient $\beta > 0$ is added to control the KL-divergence term. The final loss function applied during training is :

$$\mathcal{L}(\theta, \phi; x) = ||x - \mu_\theta \circ \mu_\phi(x)||^2 + \beta \times KL\left[\sigma_\phi(x)\mathcal{N}(0, \mathbb{I}) + \mu_\phi(x)||\mathcal{N}(0, \mathbb{I})\right]$$



(a) Discontinuous latent space (b) Well-structured latent space (c) Over-regularized latent space

Figure 2.6: Continuity of VAE's latent spaces

Yet to be able to use backpropagation in the network, it is necessary to use the reparametrization trick. It is impossible to backpropagate gradients through a random node so to solve this problem, as seen in 2.5, a random sample is generated from $\mathcal{N}(0, \mathbb{I})$ and then multiplying it to the standard deviation and adding the mean.

2.4 Invariant and equivariant properties of the latent space

It is very hard to train such a variational auto-encoder due to the high dimension of the problem, the low level of information in the signal and the limited dataset size. Reducing the size of the class of possible function can facilitate training the model.

Definition 2.1. Let $f : \Omega \mapsto \Omega$ and let G be a subgroup of $Aut(\Omega)$:

1. f is invariant under G if for all $g \in G$ and all $x \in \Omega$ then $f(g * x) = f(x)$
2. f is equivariant under G if for all $g \in G$ and all $x \in \Omega$ then $f(g * x) = g * f(x)$

Using geometric priors in the network architecture such as invariants and equivariants can facilitate the training process of networks and Bronstein and al[4] proved that many state-of-the art neural networks such as Convolutional Neural networks and Transformers can be explained from their invariant and equivariant properties.

In the case of stock market prediction, it is reasonable to assume that $\widetilde{\mathcal{M}} \mapsto \widetilde{\mathcal{M}}_\delta$ is equivaraiant under $G = \{\mathbb{I}d, -\mathbb{I}d\}$. Moreover, if we assume that the predictor verifies the property $\alpha \times \widetilde{\mathcal{M}} \mapsto \eta(\alpha) \times \widetilde{\mathcal{M}}_\delta$ for all $\alpha > 0$, we break the prediction problem in two parts:

1. Finding the optimal η such that $\eta(||\widetilde{\mathcal{M}}||) = ||\widetilde{\mathcal{M}}_\delta||$ and $\eta(-x) = -\eta(x)$
2. Finding a function $\widetilde{\mathcal{M}} \mapsto \dfrac{\widetilde{\mathcal{M}}_\delta}{||\widetilde{\mathcal{M}}_\delta||}$ that is invariant under $G = \{x \mapsto \alpha x \quad | \quad \alpha > 0\}$

In the case of the auto-encoder network, this symmetry can also be implemented by forcing a structure on the latent space. The space $\mathcal{Z} = \mathbb{R}^{N^*}$ can be structured as $\mathbb{R}^{+*} \times \mathbb{S}_{N^*-1}$. The norm is explicitly encoded and the network only learns a norm independent representation of the market state. Moreover, the first point will make sure that the latent space is point-wise symmetric with respect to the center of the space.

2.5 Results

With these methods, it is possible to learn robust non-linear relationships between different stocks. When using auto-encoders with the different modifications that are presented previously it is possible to beat by a large margin the reconstruction score of the Principal Component Analysis.Those results are summarised in Table 2.1.

With the PCA only linear relationships can be used and we see that as soon as we add non-linearity by working with an auto-encoder the reconstruction score rises quickly. However, it is difficult to scale-up the auto-encoder architecture to capture more information. Using the natural symmetries of the stock market, we are able to solve this problem and scale the auto-encoder to 5/8 layers.

	PCA	**Residual auto-encoder**	**Symmetrized auto-encoder**
1 layer	51.7%	58.9%	61.2%
3 layers	x	56.7%	64.6%
5 layers	x	52.2%	67.1%
8 layers	x	x	65%

Table 2.1: Comparison of the reconstructing power (R^2-score) of several auto-encoders.

As stated before, VAE's enable us to learn a better representation of the data (more continuous) without losing a lot of the reconstruction power. In the Table 2.2, those results are presented. By losing only 2% of the reconstruction score, we can obtain a very regular latent space.

	$\beta = 10^{-7}$	$\beta = 10^{-6}$	$\beta = 10^{-5}$	$\beta = 10^{-4}$	$\beta = 10^{-3}$
$\mathbf{R^2 - score}$	65%	64.9%	64.5%	64.4%	63.2%
KL Divergence	2140	720	240	170	90

Table 2.2: Results of the variational auto-encoder

Chapter 3

Prediction on the latent space

Building a high quality latent space highly facilitates the learning process of neural networks and makes enables the predictor to understand non-linear and complex relationships that are present on the market. In the previous chapter, we were able to create with a Variational Auto-encoder and by leveraging symmetries this regularized latent space. In this chapter, we will show how we can use these representations of the market to improve our prediction score and we will discuss the different ways to generalise this architecture to new tasks (prediction at other horizons, predicting the volatility, etc..)

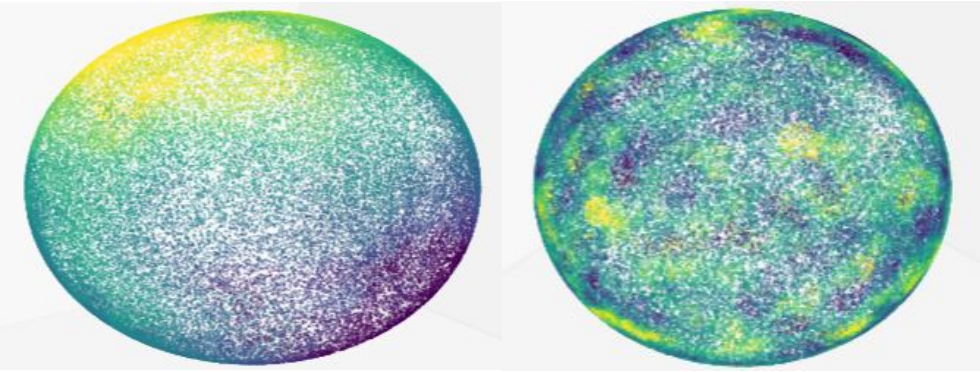
3.1 Predictive structure of the latent space

Even though the latent space was created by learning only instantaneous relationships in the stock market, it is very useful to find temporal dependencies in this space rather than working directly with the raw data from the stock market. Moreover, the continuity of the latent space that can be achieved with a VAE assures us that two points that are near each over in the latent space describe similar market states, meaning that they will likely share the same evolution in the future.

For example, by analysing the activation of a stock on the latent space (*i.e* where does the state of market lie in the latent space when the returns are positive) at time t and $t + \delta t$, what kind of market states will likely yield high returns. In Fig 3.1, we present the case of the *GlaxoSmithKline*. We see that the future price of *GlaxoSmithKline* depends on the its representation.

Using this representation in a K-nearest neighbors algorithm is able to predict pretty well the evolution of the stock market, with this method we reach a score of 0.3%. The main problem that stops us from having better scores is that in low dimension to much information is lost while in high dimension the K-nearest neighbors algorithm is not suited.

Working on the latent space let us use very rich neural networks compared to the raw data. The representations are in a lower dimension and moreover, the continuity of the latent space prevents the neural network from over-fitting on the data. This neural network can be seen as a transformation on the latent space.



(a) At time t (b) At time $t + \delta t$

Figure 3.1: Activations of the GlaxoSmithKline stock on a 2-dimension latent space (embedded in $\mathbb{R}^3$)

3.2 Transfer learning

Learning a simple problem is often a good idea to initialize the model, rather than having to learn all the important information from a difficult task, a simpler task is used to learn some important relationships in the data, then we use this network and the trained weights as the starting point for the training on the main task. This simple idea is one of the most powerful idea in Deep Learning. Many tasks are closely related either by the information they need to predict an output or by the similarity of the task. This means that we can *share* information between several task. This method is called transfer learning (Tan et al. [12]) and is used in all of the state-of-the-art deep learning model because enables the use of self-supervised learning. The reconstruction task is a self learning task, hence is particularly suited for pretraining the model.

3.3 Multi-task learning

Multi-task learning (Zhang et al. [13]) is a special training paradigm; as stated by Zhang: Multi-Task Learning (MTL) is a learning paradigm in machine learning and its aim is to leverage useful information contained in multiple related tasks to help improve the generalization performance of all the tasks. This can help us reach better results. For example, by learning to reconstruct the current market state and predict the next one, it is easier to learn interesting relationships because they should be useful for all tasks.

Learning the reconstruction and prediction task in parallel forces the latent space to contain information that are helpful to predict the future, meaning that with a smaller latent dimension it is possible to keep most of the prediction information.

With this learning paradigm, it is possible to generalise the tasks that our network can solve. By implementing the prediction task for several time horizons, we will improve the results on all tasks.

3.4 Architecture and training

To implement this method we used *Pytorch* (Paszke et al. [11]) and leverage a GTX 1000 GPU for faster training. Using the Multi-task learning paradigm, we structured the neural network as describe in Fig 3.2. The encoder and the decoder each contains 5 layers with skip connections while the predictor is a feed-forward neural network with 3 layers and skip connections. The encoder learns to work with a norm independent representation of the market state, but the predictor and the decoder uses the explicit norm of $\widetilde{\mathcal{M}}$ to reconstruct and predict $\widetilde{\mathcal{M}}_\delta$. The $\pm$ symmetry (invariance under $G = \{\mathbb{I}d, -\mathbb{I}d\}$) is implemented by data augmentation: we duplicate the dataset by applying $x \mapsto -x$ on the samples of $\widetilde{\mathcal{M}}$ and $\widetilde{\mathcal{M}}_\delta$.

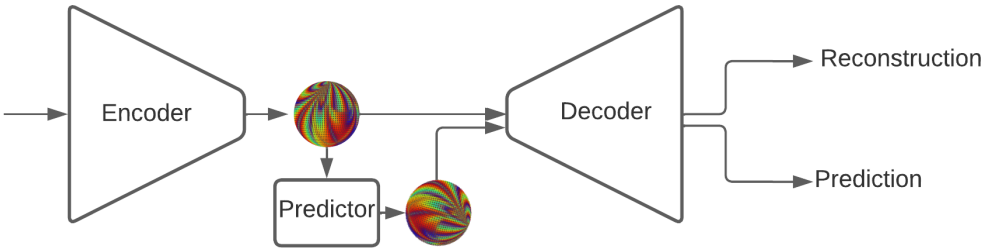


Figure 3.2: Architecture of a network that learns to reconstruct the instantaneous market state and predict the next one.

The training procedure leverage two training paradigms, firstly we train for 1000 epochs the variational auto-encoder only to learn a suitable representation of the market, then we jointly train the variational auto-encoder with predictor for 8000 epochs and finally we only train the predictor for 1000 epochs. During the second part of the training, we enable the predictor loss in the variational auto-encoder loss but we also scale it with a small factor, this have the effect to modify the latent space so that information that is useful for the prediction are forced to be present in the representation.

3.5 Results

Using the architecture and the training principles described in this chapter, we were able to significantly improve the prediction score of our model while also facilitating the training of the neural network. We showed also that learning a regularised space directly affects the results.

	Prediction R^2-Score	**Improvement**
Elastic-Net (Baseline)	0.56%	-
AE + Linear Predictor + no MTL	0.57%	+ 2%
VAE + Linear Predictor + no MTL	0.62%	+ 10.7%
Final Model	**0.668%**	**+ 19.3%**

Conclusion

The results obtained during my internship prove that non-linear relationships on the European stock market are not negligible. Deep neural network are able to leverage these relationships to improve the results, but working with a simple Multi-Layer Perceptron will over-fit.

Computer Vision techniques such as Variational Auto-Encoders and Residual Networks are in practice necessary to learn generalizable relationships. Moreover, we showed how helpful can symmetries be for training, particularly on the stock market due to the high amount of noise.

Implementing these methods enabled us to increase by a big margin the quality of the latent space that the model learns (in quantity of information and in regularity/continuity). Then using this representation facilitated the training and helped increase the prediction power of the model.

Finally, the framework developed during my internship is highly generalizable to new tasks and to new signals (*i.e* new features to do the prediction). Indeed, firstly it is possible to add new features (new stocks/indices, fundamental data, etc..) without increasing the dimension of the problem (we choose a reasonable size for the latent space), and secondly, with the multi-task training paradigm, it is possible to solve a many task concurrently.

Bibliography

[1] Hervé Abdi and Lynne J Williams. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[2] Richard Bellman. "Dynamic programming". In: *Science* 153.3731 (1966), pp. 34–37.

[3] Y. Bengio and Yann Lecun. "Convolutional Networks for Images, Speech, and Time-Series". In: (Nov. 1997).

[4] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. DOI: 10.48550/ARXIV.2104.13478. URL: https://arxiv.org/abs/2104.13478.

[5] Mahsa Ghorbani and Edwin K. P. Chong. *Stock Price Prediction using Principle Components*. DOI: 10.48550/ARXIV.1803.05075. URL: https://arxiv.org/abs/1803.05075.

[6] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[7] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=Sy2fzU9gl.

[8] Gyeeun Jeong and Ha Young Kim. "Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning". In: *Expert Systems with Applications* 117 (2019), pp. 125–138.

[9] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. DOI: 10.48550/ARXIV.1312.6114. URL: https://arxiv.org/abs/1312.6114.

[10] Prapanna Mondal, Labani Shit, and Saptarsi Goswami. "Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices". In: *International Journal of Computer Science, Engineering and Applications* 4.2 (2014), p. 13.

[11] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[12] Chuanqi Tan et al. *A Survey on Deep Transfer Learning*. 2018. DOI: 10.48550/ARXIV.1808.01974. URL: https://arxiv.org/abs/1808.01974.

[13] Yu Zhang and Qiang Yang. *A Survey on Multi-Task Learning*. 2017. DOI: 10.48550/ARXIV.1707.08114. URL: https://arxiv.org/abs/1707.08114.

[14] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the royal statistical society: series B (statistical methodology)* 67.2 (2005), pp. 301–320.

[15] Zhichao Zou and Zihao Qu. "Using LSTM in Stock prediction and Quantitative Trading". In: *CS230: Deep Learning, Winter* (2020), pp. 1–6.