Radboud University Nijmegen

Faculty of Science

Connectivity Estimation in Financial Time Series Using Deep Learning Models

Thesis MSc Computing Science, spec. Data Science

Supervisor:
dr. Yuliya Shapovalova,
Assistant Professor

Author:
Bruno Jerković
s1061740

Second reader:
dr. ir. Tom Claassen,
Assistant Professor

April 2023

Abstract

As financial data becomes increasingly complex, understanding the relationships between assets is crucial for informed decision-making. This thesis explores the potential of employing deep learning models to estimate connections between financial time series, paving the way for more robust investment strategies and risk management approaches. To this end, we applied both the neural relational inference (NRI) model and the neural Granger causality (NGC) model to estimate the connections. We also introduce a novel architecture, cNSVM, which combines the NGC model and the neural stochastic volatility (NSVM) model. To compare the aforementioned deep learning models with machine learning methods, we also analyzed the performance of the t-VAR model. We have evaluated these models on synthetically generated datasets and a real-world dataset of historical stock prices. On the real-world stock dataset, t-VAR and NRI emerged as the best-performing models. Our findings contribute to the existing literature in financial time series by showcasing the strengths and weaknesses of connectivity estimation deep learning models. Looking ahead, future research could focus on refining the formulation of the newly introduced cNSVM to further enhance its performance, ultimately leading to better financial decision-making and risk management.

Contents

1 Introduction

In an ever-evolving financial landscape, understanding the intricate connections between assets has become crucial for making informed investment decisions [1]. By analyzing how assets are connected, investors can identify patterns that help them make strategic choices and construct more effective portfolios that balance risk and return [2]. By leveraging the knowledge of asset connectivity, they can diversify their holdings across a range of assets [3]. Moreover, recognizing the relationships between different assets enables investors to mitigate potential risks and hedge their investments, ultimately protecting their portfolios from market volatility and potential losses [4]. But how can we reliably estimate these connections in a complex, interconnected world? This thesis explores the potential of employing deep learning models to estimate connectivity between financial time series, paving the way for more robust investment strategies and risk management approaches.

The primary goal of this thesis is to explore how deep learning can help estimate the connections between financial time series data. Traditional methods for connectivity estimation often **lack generalizability** [5, 6]. In a survey by *Broto et al.* [7], various models were used based on different properties of financial data. Deep learning could overcome this problem by generalizing multiple models at once [8]. Another issue with traditional methods is that they can be **computationally expensive in high dimensions** [9]. They often use sampling-based approaches, which are slow in high dimensions. On the other hand, deep learning methods are known to scale well [10]. Additionally, parameter estimation for some traditional methods can be difficult and may **require specific prior knowledge** about parameter distributions (as the likelihood function is hard to estimate and uses Bayesian approaches) [11]. On the contrary, deep learning methods can learn well without needing such detailed prior knowledge [12]. In summary, current state-of-the-art methods can lack generalizability and scalability or require extensive prior knowledge. All of these limitations could be tackled with deep learning approaches.

In order to properly analyze the generalizability of these methods, we applied them to both the synthetically generated data and a real-world stock market dataset. Using synthetically generated datasets allowed us to create test-scenarios with different connection strengths and numbers of variables (to test for scalability). Synthetic data is generated from the vector autoregressive model (VAR, section 3.1) and the stochastic volatility model (SV-AR, section 2.2.2). VAR is a statistical model often used in financial time series analysis that captures the linear interdependencies between multiple time series variables, while SV-AR is a financial model that accounts for the time-varying and random nature of asset price volatility [13]. Stochastic volatility is often influenced by volatility spillovers, which are the transmissions of volatility shocks from one financial asset to another, which is what makes its modeling a difficult task [14, 15]. However, both models have their limitations. VAR assumes that the dependencies are linear, which does not have to be the case for real-world data. On the other hand, the parameters of SV-AR are hard to estimate (due to the intractable estimation of the likelihood function), particularly in high dimensions, which can make it challenging to apply it to real-world scenarios [16, 17].

The models used for estimating connectivity are deep learning-based models, such as a neural relational inference (NRI, section 3.2) model and the neural Granger causality (NGC, section 3.3) model [18, 19]. While it has not yet been used for financial data, the NRI model offers a way to estimate connections between assets in the form of a graph, with nodes being assets and edges being the connections between them. The NGC model offers a framework for directly estimating the connections between time series. Even though its original formulation is limited to using multilayer perceptron (MLP) and long short-term memory (LSTM) networks, we extend it by combining it with an architecture that is better suited for stochastic volatility modeling, a neural stochastic volatility model (section 3.4) introduced by *Luo et al.* [20]. Furthermore, to compare deep learning against the machine learning approaches, we used t-VAR, introduced by *Barbaglia et al.* [21], that fits the data to the model similar to VAR.

Other than analyzing the performance of deep learning models, we also wanted to analyze some additional useful properties. To that note, we also analyzed the scalability of connectivity estimation. The financial data is often high-dimensional and constantly evolving, requiring models to be able to handle large datasets [22, 23]. Furthermore, we are also interested in analyzing how well the deep learning models estimate the covariance matrix of the noise generation process. The presence of a correlation between noise distributions can negatively impact the performance of connectivity estimation models if they mistake the connection between assets with the covariance of the noise signal [24].

Additionally, estimating the noise distribution of financial models allows for modeling the unobserved factors that influence the time series and are not present in the observed data [25]. This motivated us to investigate the performance of these models to different noise magnitudes in the data. Finally, we also inspected if these models have a bias toward estimating connections rather than non-connections. Understanding such biases is crucial for accurately interpreting the results and making well-informed investment decisions.

To make an extensive analysis of the applicability of these deep learning models for estimating connections between financial time series, we aim to answer the following research questions:

1. How do deep learning models for connectivity estimation perform on data with different strengths of connections, considering both the VAR and SV-AR generated data?
2. How robust are connectivity estimation models to noise? Additionally, can they effectively distinguish between contemporaneous correlation in the correlation terms of the covariance matrix of the noise generation process and the connections encoded in the coefficient matrices of both VAR and SV-AR models?
3. How does data dimensionality affect the performance and training time of connectivity estimation models, particularly in high-dimensional scenarios?
4. Are the connectivity estimation models biased towards estimating connections or non-connections?
5. How well do connectivity estimation models perform on real-world historical stock price datasets compared to existing models in the literature?

In summary, this study is organized into different chapters. The first chapter starts by defining the concept of connectivity. It further provides a background on the definition and importance of financial models used to create synthetic data and the analysis of the real-world dataset used in the study. The second chapter introduces the models used for estimating connectivity. It provides a solid background on how they work, their motivation for using them, and how they compare. The third chapter displays an analysis of the results of the experiments. The fourth chapter summarizes the main thoughts of the experiments and provides answers to the above research questions. Finally, the last chapter puts the critical findings into perspective and concludes this study.

2 Background

In this section, we explained some of the crucial concepts that were used during this study. Granger causality plays a vital role in connectivity estimation, allowing us to investigate relationships between different time series. In this section, we provide the necessary concepts for understanding Granger causality (section 2.1), as well as two models used for generating synthetic data: vector autoregressive (VAR, section 2.2.1) and stochastic volatility (SV-AR, section 2.2.2) models. An understanding of the Granger causality concept is necessary as we have imposed the connectivity between the variables of the synthetic time series by imposing the Granger causality on the variables of data generation models. Moreover, the applicability of these models extends beyond synthetic data; they have also been tested on a real-world dataset, which is described later in this chapter (section 2.3).

2.1 Modelling Connectivity as Granger Causality

The concept of causality is crucial for inferring connectivity between time series. Two main properties for determining cause-effect relationships are temporal precedence and physical influence [26]. Temporal precedence states that a time series that causes another precedes it in time, and physical influence states that manipulations of causing time series change the affected time series. Both of these statements are true for financial data. For example, if the value of one stock changes, it should have an effect on the value of another stock (physical influence) but at a later point in time (temporal precedence). This suggests that quantifying causality is valuable in assessing connectivity. However, there are multiple types of causality, such as structural causality, intervention causality, and Granger causality [26] [27]. In this study, we have used Granger causality as it provides a straightforward way of imposing connectivity between synthetically created time series (more in sections 2.2.1, 2.2.2).

The idea of Granger causality was introduced by C.W.J. Granger in 1969 [26]. In time series analysis, inferring the cause-effect relationship is commonly based on Granger causality. Specifically, let X and Y be two stationary stochastic processes (or time series). If disregarding X from predicting values of Y makes the larger residuals of predictions, it indicates that X contains some information regarding Y, and thus X Granger causes Y. This formulation allows for the modeling of both linear and non-linear dependencies. Additionally, it should be noted that Granger causality can lead to wrongly inferred causal relations in the presence of latent variables. If, for example, there are three-time series: X, Y, and Z, where X_{t-2} causes the change in Y_{t-1} and Z_t. If only Y and Z are observable, Granger causality infers that Y Granger causes Z, even though that is not the case.

2.2 Data Generation Models

To test the performances of the neural models used to determine the connectivity in financial data (more in section 3), they were initially tested on synthetically created stochastic processes. These stochastic processes were generated from two well-known time series models frequently used to model the dynamics of financial data. Variables in these stochastic processes represent asset returns, which can be: currency prices over time, values of index returns over time, etc.

2.2.1 Vector Autoregressive Model

The first data-generation model used in this study is the vector autoregressive (VAR) model, introduced by C. Sims in 1980 [28]. It is a simple and interpretable model that models multiple time series variables and allows for modeling connections between them. It has been extensively used in various applications with financial data [29], [30], [31].

In this study, we use VAR to model the asset returns. Let $\boldsymbol{y_t}$ be asset returns, then the probabilistic formulation of the model is defined in equations (1) and (2), and analytical formulation of the model is defined in equation (3).

$$\boldsymbol{y_t}|\boldsymbol{y_{t-1}} \sim N(\boldsymbol{\mu} + \Phi(\boldsymbol{y_{t-1}} - \boldsymbol{\mu}), \boldsymbol{\Sigma_\eta}), \tag{1}$$

$$\boldsymbol{y_0} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma_\eta}/(1 - \boldsymbol{\Phi}^2)), \tag{2}$$

$$\boldsymbol{y_t} = \boldsymbol{\mu} + \boldsymbol{\Phi}(\boldsymbol{y_{t-1}} - \boldsymbol{\mu}) + \boldsymbol{\eta_t}, \tag{3}$$

where $\boldsymbol{y_t} = (y_t^i)_{i=1,\dots,N}$ is a vector of N asset returns at time t, $\boldsymbol{\mu}$ is the vector of N means of asset returns, $\boldsymbol{\Phi} = (\phi_{ij})_{i,j=1,\dots,N}$ is a matrix of size $N \times N$ that contains autoregression coefficients, $\boldsymbol{\eta_t}$ is a Gaussian noise, $\boldsymbol{\eta_t} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma_\eta})$, with $\boldsymbol{\Sigma_\eta} = (\sigma_{\eta,ij})_{i,j=1,\dots,N}$.

From equation (1), it is visible that the returns only depend on the previous timestep, $t-1$. Elements on the diagonal of $\boldsymbol{\Phi}$ define the linear dependence of the returns of the same asset, which indicates its self-dependence. Elements on the off-diagonal of $\boldsymbol{\Phi}$ define the linear dependence of one asset's return at the current timestep on the returns of the rest of the assets at the previous timestep. In this scenario, Granger causality is defined as the linear dependence between asset returns. If $\boldsymbol{\phi_{i,j}}$ is non-zero, we say that the asset j Granger causes asset i, and there is a connection from j to i. Otherwise, if $\boldsymbol{\phi_{i,j}}$ is zero, there is no connection between assets i and j. Granger causality could be indicative of risk estimation, as the non-existence of Granger causality indicates a market that is harder to predict, therefore, a riskier market.

In this study, the VAR model was used only to generate stationary data because non-stationary data would lead to exploding unpredictable values. For example, if a model is fit to non-stationary data exhibiting an upward trend, the model may predict that the trend continues indefinitely, which is unrealistic. Therefore, ensuring that financial data is stationary before modeling it is crucial. In order for the VAR model to generate stationary data, the roots of the equation $|I - \lambda\boldsymbol{\Phi}| = 0$ (where I is the identity matrix of size $N \times N$) should lie outside of the unit circle [32].

We further provide an example of the data generated from the VAR model. For variables $y^1_{1:t}$ and $y^2_{1:t}$, figure 1 illustrates an example of data generated from the VAR model. Model parameters used to generate this data are:

$$\boldsymbol{\mu} = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \quad \boldsymbol{\Phi} = \begin{pmatrix} 0.8 & 0. \\ 0.6 & 0.9 \end{pmatrix} \quad \boldsymbol{\Sigma}_\eta = \begin{pmatrix} 0.1 & 0. \\ 0. & 0.01 \end{pmatrix} \tag{4}$$

As already mentioned, $\boldsymbol{\mu}$ defines the mean of the data, which is the value around which $y^1_{1:t}$ and $y^2_{1:t}$ are centered. $\boldsymbol{\Sigma}_\eta$ is the variance-covariance of the noise, $\boldsymbol{\eta}$ that models the stochasticity of the data. Since $\sigma_{\eta,1,1} > \sigma_{\eta,2,2}$ we can see that $\boldsymbol{y_1}$ exhibits greater differences in its values than $\boldsymbol{y_2}$. $\boldsymbol{\Phi}$ is the autoregression coefficient matrix. Because $\Phi_{1,2} = 0$, $\boldsymbol{y_1}$ is independent of $\boldsymbol{y_2}$ and only depends on its previous value, in the Granger causal sense. However, $\boldsymbol{y_2}$ depends on its previous value and the previous value of $\boldsymbol{y_1}$, which indicates Granger causality.

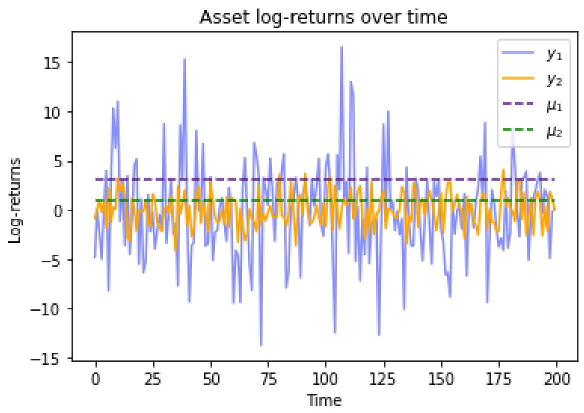


Figure 1: Data generated from VAR model with parameters specified in equation (4)

2.2.2 Stochastic Volatility Model

The VAR discussed in the previous section does not incorporate many properties of real financial data. Numerous empirical studies have been done on asset return data throughout the years to extract its qualitative properties known as stylized facts [33]. One of these, known as volatility clustering, states that volatility changes through time, and high-volatility events tend to cluster together [34]. Numerous models have been developed to generate or fit the data with changing volatility [35]. In this study, we also use one of such models - the discrete multivariate form of stochastic volatility autoregressive model, SV-AR model, proposed in [36]. It autoregressively models the volatility dynamic with the same definition that was used in VAR to model the dynamics of asset returns.

In addition to modeling volatility clustering, SV-AR also incorporates other stylized facts, such as volatility persistence and co-movements [33]. Volatility persistence states that there is a time dependence in volatility, meaning that future volatility is influenced by past volatility. Co-movements state that the returns of different assets tend to change in the same direction. Furthermore, while we do not consider it here, SV-AR can also incorporate one more stylized fact - the leverage effect.

Let $\boldsymbol{y_t}$ be asset returns and $\boldsymbol{h_t}$ volatility; then, the model is defined with the following equations. Equations (5) - (7) show the probabilistic definition of the model, and equations (8), (9) show the analytical definition of the model.

$$\boldsymbol{y_t}|\boldsymbol{h_t} \sim N(0, \boldsymbol{\Sigma_\epsilon} diag(\exp(\boldsymbol{h_t}))\boldsymbol{\Sigma_\epsilon}), \tag{5}$$

$$\boldsymbol{h_t}|\boldsymbol{h_{t-1}} \sim N(\boldsymbol{\mu} + \Phi(\boldsymbol{h_{t-1}} - \boldsymbol{\mu}), \boldsymbol{\Sigma_\eta}), \tag{6}$$

$$\boldsymbol{h_0} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma_\eta}/(1 - \boldsymbol{\Phi}^2)), \tag{7}$$

$$\boldsymbol{y_t} = diag(\exp(\boldsymbol{h_t}/2))\boldsymbol{\epsilon_t}, \tag{8}$$

$$\boldsymbol{h_t} = \boldsymbol{\mu} + \boldsymbol{\Phi}(\boldsymbol{h_{t-1}} - \boldsymbol{\mu}) + \boldsymbol{\eta_t}, \tag{9}$$

where $\boldsymbol{y_t} = (y_{i,t})_{i=1,\dots,N}$ is a vector of N observed asset returns at time t, $\boldsymbol{h_t} = (h_{i,t})_{i=1,\dots,N}$ is a vector of N unobserved volatilities of asset returns at time t. $\boldsymbol{h_t}$ parametrizes the volatility of asset returns since the variance of $\boldsymbol{y_t}$ in equation (8) is parameterized by $\boldsymbol{h_t}$. From equation (6), it is visible that the volatility only depends on one timestep, or one lag, in the past. Furthermore, the noise $\boldsymbol{\mu}$ is the vector of N means of asset volatilities. The coefficient matrix $\boldsymbol{\Phi} = (\phi_{ij})_{i,j=1,\dots,N}$ is a matrix of size $N \times N$ that contains autoregressive coefficients of the volatility process. Its diagonal elements, ϕ_{ii} determine how much effect the asset volatility at a preceding lag, $h_{i,t-1}$, has on its volatility at a current lag, $h_{i,t}$. Its off-diagonal elements, ϕ_{ij}, determine how much effect the volatility of one asset at a preceding lag, $h_{j,t-1}$, has on the volatility of another asset at a current lag, $h_{i,t}$. $\boldsymbol{\Sigma_\eta} = (\sigma_{\eta,ij})_{i,j=1,\dots,N}$ is a covariance matrix of size $N \times N$ for the conditional distribution of volatility, $\boldsymbol{h_t}$. $\boldsymbol{\epsilon_t}$ is a random noise vector of size N sampled from a Gaussian distribution, $\boldsymbol{\epsilon_t} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma_\epsilon})$, $\boldsymbol{\Sigma_\epsilon} = (\sigma_{\epsilon,ij})_{i,j=1,\dots,N}$. $\boldsymbol{\eta_t}$ is a random noise vector of size N sampled from a Gaussian distribution, $\boldsymbol{\eta_t} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma_\eta})$, $\boldsymbol{\Sigma_\eta} = (\sigma_{\eta,ij})_{i,j=1,\dots,N}$.

In SV-AR, asset returns are dependent on the unobservable process of volatilities. This makes the movement of asset returns more complex as they are defined by underlying, unobserved dynamics. Thus, the likelihood function is intractable which makes the model hard to estimate in higher dimensions [16, 17]. However, with its formulation, SV-AR can model the concept known as volatility persistence. Volatility persistence states that there is a time dependence in volatility, meaning that future volatility is influenced by past volatility. This is because volatility $\boldsymbol{h_t}$ is autoregressively modeled with $\boldsymbol{h_{t-1}}$, equations (6), (9). By modeling the volatility persistence, it also models the dependence of an asset i between its lags $t-1$ and t. With volatility persistence, volatility clustering is also modeled because it makes small and large volatilities cluster together in time. Furthermore, by having the off-diagonal values of the coefficient matrix $\boldsymbol{\Phi}$ to be non-zero, SV-AR incorporates co-movements stylized fact. Co-movements state that the returns of different assets tend to change in the same direction. While we do not consider it here, it is also possible to incorporate one more stylized fact - the leverage effect.

Furthermore, we are only interested in generating stationary data since the values of the non-stationary data would explode and would not be predictable. In SV-AR, the condition of stationarity is achieved (in the same manner as with the VAR model) by applying the constraint on the coefficient matrix of the autoregressive process: the roots of $|I - \lambda\mathbf{\Phi}|$ must lie outside of the unit circle. Thus, when generating the data in experiments, we take that into account.

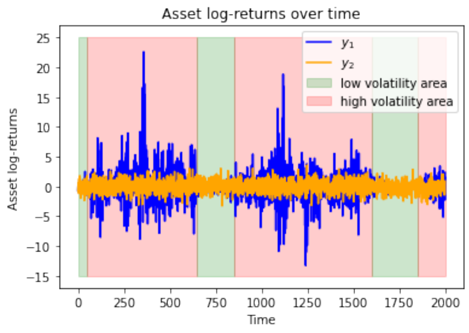
We further illustrate one example of data generated from this model. Figure 2 shows an example of the dynamic of asset returns and volatilities generated by SV-AR. The figures were generated with the following parameters:

$$\boldsymbol{\mu} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \mathbf{\Phi} = \begin{pmatrix} 0.99 & 0.3 \\ 0 & 0.2 \end{pmatrix} \quad \mathbf{\Sigma}_\eta = \begin{pmatrix} 0.05 & 0.0 \\ 0.0 & 0.01 \end{pmatrix} \quad \mathbf{\Sigma}_\epsilon = \begin{pmatrix} 1 & 0.6 \\ 0.6 & 1 \end{pmatrix} \tag{10}$$

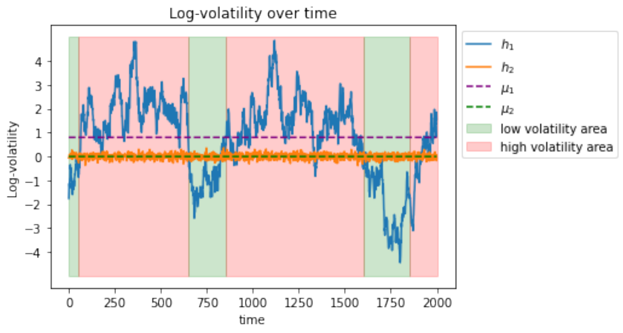
As shown in figure 2b, μ_1 and μ_2 define the unconditional means of volatilities of asset returns. Since $\mu_1 > \mu_2$, the returns of the first asset, y_1, are more volatile than the returns of the second asset, y_2.
In figure 2b, we notice that the second asset's volatilities, h_2, look very similar to the noise process. This is because the elements of the coefficient matrix relevant for that asset are very small (relative to other values of $\mathbf{\Phi}$). Since $\phi_{2,1} = 0$, the volatility more clusters on the volatility of the second asset. Elements on the diagonal of $\mathbf{\Phi}$ model volatility persistence and their larger value would mean that there are more clusters of the prices.
In figure 2a, it is possible to see two clusters of low volatility asset returns (in green areas) and three clusters of high volatility asset returns (in red areas). This is because, in figure 2b, the volatility in red areas has larger values than the volatility in green areas.
$\boldsymbol{\Sigma_\eta}$ defines the covariance matrix of the noise process of the dynamics of volatilities. Because the value of the variance of the noise of the first volatility, $\sigma_{\eta,1,1}$ is larger than the value of the variance of the noise of the second volatility, $\sigma_{\eta 2,2}$, volatility of the second asset, h_2, is more stochastic than the volatility of the first asset h_1.

$\boldsymbol{\Sigma_\epsilon}$ is the covariance matrix of the normal distribution from which $\boldsymbol{\epsilon}_t$ is sampled. With larger values in $\boldsymbol{\Sigma_\epsilon}$, the volatility less reflects the true volatility that the asset returns are experiencing.

Furthermore, it is worth indicating the differences between the asset returns generated by the VAR model (figure 1) and the asset returns generated by the SV-AR model (figure 2a). While the former one exhibits no clusters of variance, the latter one shows areas of high or low variance.



(a) Example of asset returns for two random variables, $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$.



(b) Example of volatilities for two random variables, $\boldsymbol{h_1}$ and $\boldsymbol{h_2}$ (with their respective mean values).

Figure 2: Asset returns and volatility generated from SV-AR with parameters specified in equation (10)

2.3 Dataset of Historic Stock Prices

Other than using the aforementioned data-generation models to create the dataset, we also used a historical dataset of stock prices. Since we compared the results of our neural models with the results from *Hecq et al.*, we used to same datasets as the authors [37]. This subsection shows some of the statistics measured from the real-world dataset.

To get more insight into the datasets of historic stock prices, the following summary statistics were calculated: mean, median, mode, range, standard deviation, variance, skewness, kurtosis, and Pearson's correlation. The mean and median provide information about the central tendency. The mean is the average of the stock prices, and the median is the middle value when the stock prices are ordered (and, as such, is more robust to outliers). Furthermore, the range, standard deviation, and variance provide insight into the spread of the data. The range is the difference between the highest and lowest stock prices, and as such, it provides insight into how much variation exists in the data and how widely spread the values are. The standard deviation is a measure of the spread of the stock prices around the mean, while the variance is the square of the standard deviation. They give an indication of how tightly or loosely the values cluster around the mean. Moreover, kurtosis and skewness provide additional information about the shape of the distribution of stock prices. Kurtosis quantifies the degree to which the data deviates from a normal distribution, with a positive kurtosis indicating a distribution with more extreme values than a normal distribution and negative kurtosis indicating a flatter distribution. Skewness measures the asymmetry of the distribution, with a positive skewness indicating a distribution that is skewed to the right (with a long tail on the positive side of the mean) and negative skewness indicating a distribution that is skewed to the left (with a long tail on the negative side of the mean). Finally, Pearson's correlation is a statistical measure of the linear association between two variables, where the values range between -1 and +1, with +1 indicating a strong positive linear relationship, -1 indicating a strong negative linear relationship, and 0 indicating no linear relationship. Measuring Pearson's correlation between stocks before using more complex methods for estimating connectivity between stocks can provide a quick and easy insight into the level of the linear relationship between the stocks and help to identify trends and patterns in the data.

Stock Name	**Mean**	**Median**	**Range**	**Stand. Dev.**	**Variance**	**Kurtosis**	**Skewness**
AAPL	64.11	61.32	126.09	34.69	1203.64	-1.22	0.16
AXP	54.35	53.49	83.49	20.73	429.69	-0.98	-0.04
BA	84.47	66.16	149.67	37.32	1392.67	-1.32	0.34
CAT	69.83	74.49	84.18	19.45	378.23	-0.12	-0.78
CSCO	20.52	19.98	22.61	4.88	23.85	-0.70	0.42
CVX	83.39	86.22	79.57	19.87	394.66	-1.16	-0.26
DD	44.61	42.38	66.61	15.73	247.28	-1.01	0.06
DIS	56.77	46.06	105.83	30.55	933.27	-1.29	0.46
GE	19.67	19.78	27.89	6.34	40.14	-1.01	0.11
GS	145.68	150.47	209.22	34.96	1222.00	0.27	-0.06
HD	61.73	50.28	129.11	39.23	1539.34	-1.09	0.57
IBM	140.02	148.35	134.98	32.31	1043.83	-0.83	-0.55
INTC	22.08	20.37	28.97	7.16	51.31	-0.90	0.45
JNJ	72.14	60.43	87.88	23.94	573.36	-1.22	0.49
JPM	44.07	38.39	78.51	14.46	209.05	-0.09	0.66
KO	31.27	32.95	31.30	8.25	68.01	-1.23	-0.21
MCD	77.54	81.50	93.89	23.65	559.52	-0.82	0.16
MMM	99.95	81.78	150.26	40.75	1660.24	-1.17	0.46
MRK	38.88	37.22	50.79	13.28	176.26	-1.35	0.24
MSFT	31.47	25.85	53.38	12.60	158.69	-0.41	0.84
NKE	30.30	24.52	58.29	16.19	262.20	-0.98	0.61
PFE	21.41	20.41	27.71	7.85	61.69	-1.50	0.16
PG	62.31	57.88	57.23	13.85	191.87	-1.26	0.16
TRV	68.03	57.89	99.62	27.76	770.60	-1.28	0.37
UNH	64.43	51.39	151.25	39.24	1539.87	-0.52	0.81
UTX	78.03	73.76	87.44	22.53	507.40	-1.21	-0.04
V	39.40	30.98	77.82	23.33	544.51	-1.22	0.52
VZ	34.41	35.77	39.69	10.65	113.34	-1.43	0.03
WMT	58.08	59.04	47.90	12.45	154.96	-1.50	0.07
XOM	72.24	73.69	50.70	12.19	148.72	-1.06	-0.22

Table 1: Statistics of the historical stock prices dataset measured on the full interval

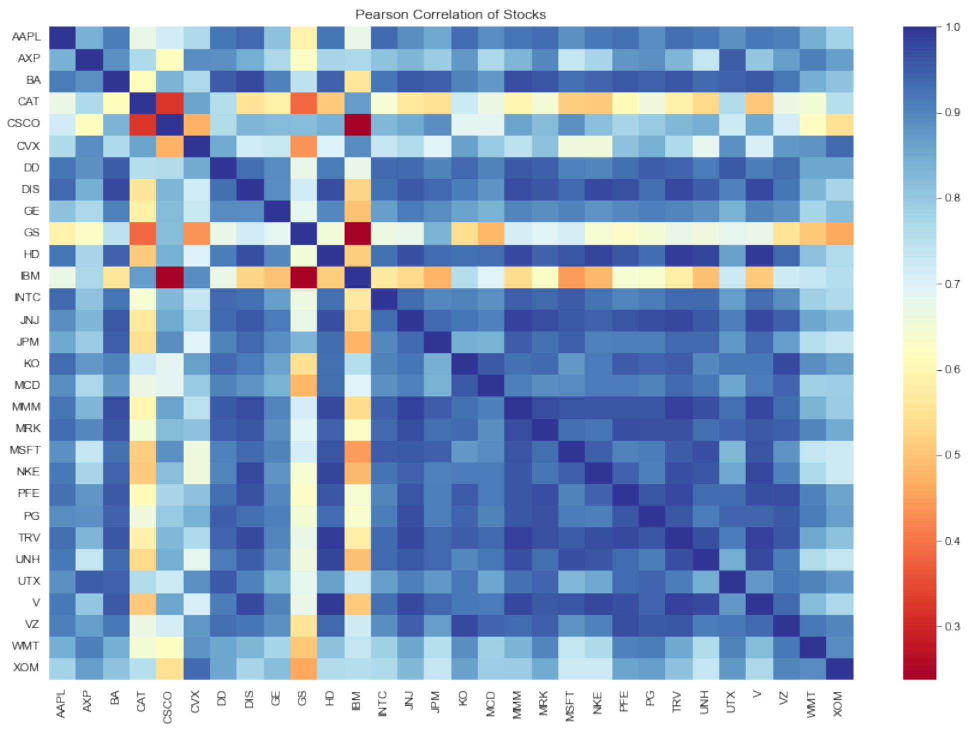


Figure 3: Heatmap of Pearson's correlation between stocks taken from the full dataset interval

The first dataset consists of 30 stock prices on 1-minute intervals from March 2008 until February 2017 (2236 trading days). The table 1 shows the mean, median, range, standard deviation, variance, kurtosis, and skewness for each stock. Based on the data, we can see that GS (Goldman Sachs Group Inc) has the highest mean of 145.68, and GE (General Electric) has the lowest mean of 19.67. It seems that most stocks' mean and median values are close to each other, which means that the data is likely to be symmetrically distributed. However, 20 out of 30 stocks have a slightly higher mean than the median, indicating a slightly positive skew in the 67% of the data and a slightly negative skew in the 33% of the data. In terms of kurtosis, the data shows that the kurtosis is generally negative for most of the stocks, which suggests that the distributions of prices for most stocks are less spread out than a normal distribution and have fatter tails. For example, American Express Co. (AXP) has a kurtosis value of -0.98. However, Goldman Sachs (GS) is the only stock among all with a positive kurtosis value of 0.27. The range of returns is relatively large for some stocks, such as Apple (AAPL) at 126.09 and Boeing (BA) at 149.67, indicating a higher degree of volatility. This suggests that the returns for these stocks have been more spread out over the observed period. Cisco Systems (CSCO) has the lowest range value among all the stocks, which is 22.61. The standard deviation of returns is generally higher for stocks with larger ranges and higher means, indicating that the returns for these stocks are more spread out. For example, The standard deviation of returns for Apple (AAPL) is 34.69, which is relatively high when compared to Cisco Systems (CSCO) of 4.88. 3M (MMM) has the highest standard deviation value among all the stocks, at 40.75.

Furthermore, the results of Pearson's correlation for each of the stock pairs are shown in figure 3 in the form of a heatmap. The correlation values generally range from 0.5 to 1.0, with some values slightly below 0.5, indicating that the relationship between the variables is relatively strong. There are no negative correlation values, indicating that none of the variables have a negative relationship.

Stock Name	**Mean**	**Median**	**Range**	**Stand. Dev.**	**Variance**	**Kurtosis**	**Skewness**
AAPL	105.45	105.42	48.01	10.30	106.12	-0.03	0.60
AXP	64.60	63.73	30.75	6.82	46.49	-0.35	0.37
BA	133.51	129.54	74.37	14.39	207.02	-0.08	0.81
CAT	78.81	78.90	45.01	11.15	124.36	-0.87	-0.06
CSCO	28.47	29.40	12.27	2.54	6.47	0.00	-0.84
CVX	98.70	99.00	46.38	10.62	112.71	-0.38	-0.25
DD	66.20	66.64	29.02	5.85	34.17	0.16	-0.35
DIS	97.81	96.86	26.74	5.38	28.89	-0.08	0.81
GE	29.89	29.92	6.21	1.21	1.47	-0.33	-0.32
GS	175.82	161.86	114.92	33.13	1097.65	-0.21	1.17
HD	129.36	129.93	35.66	6.11	37.33	0.28	-0.63
IBM	149.61	150.56	69.81	14.71	216.40	-0.06	-0.39
INTC	33.03	33.93	11.33	2.90	8.41	-1.32	-0.16
JNJ	112.22	114.18	32.57	7.21	51.98	-0.02	-0.80
JPM	66.95	63.78	39.82	10.28	105.75	-0.48	0.91
KO	42.59	42.25	6.51	1.45	2.11	-0.99	0.35
MCD	118.72	119.01	19.54	4.52	20.39	-1.08	0.08
MMM	166.46	168.37	53.08	11.25	126.59	1.03	-1.23
MRK	57.13	58.08	19.39	4.97	24.73	-1.13	-0.38
MSFT	55.19	55.30	18.73	4.82	23.25	-1.11	0.31
NKE	55.66	55.44	15.90	3.42	11.68	-0.93	0.17
PFE	31.95	32.15	9.11	2.08	4.32	-0.80	-0.22
PG	82.47	82.70	19.61	3.75	14.04	-0.27	-0.34
TRV	112.64	113.46	23.98	5.15	26.54	-0.53	-0.36
UNH	137.03	137.26	59.20	15.31	234.35	-0.67	0.04
UTX	100.54	100.77	31.55	7.45	55.43	0.12	-0.78
V	78.08	78.48	21.97	4.09	16.77	-0.30	-0.42
VZ	50.04	50.02	13.14	2.60	6.76	0.61	-0.58
WMT	68.42	68.83	15.78	3.20	10.21	-0.14	-0.65
XOM	83.99	85.11	24.50	4.80	23.05	0.49	-0.84

Table 2: Statistics of the historical stock prices dataset measured on 2016-2017 interval

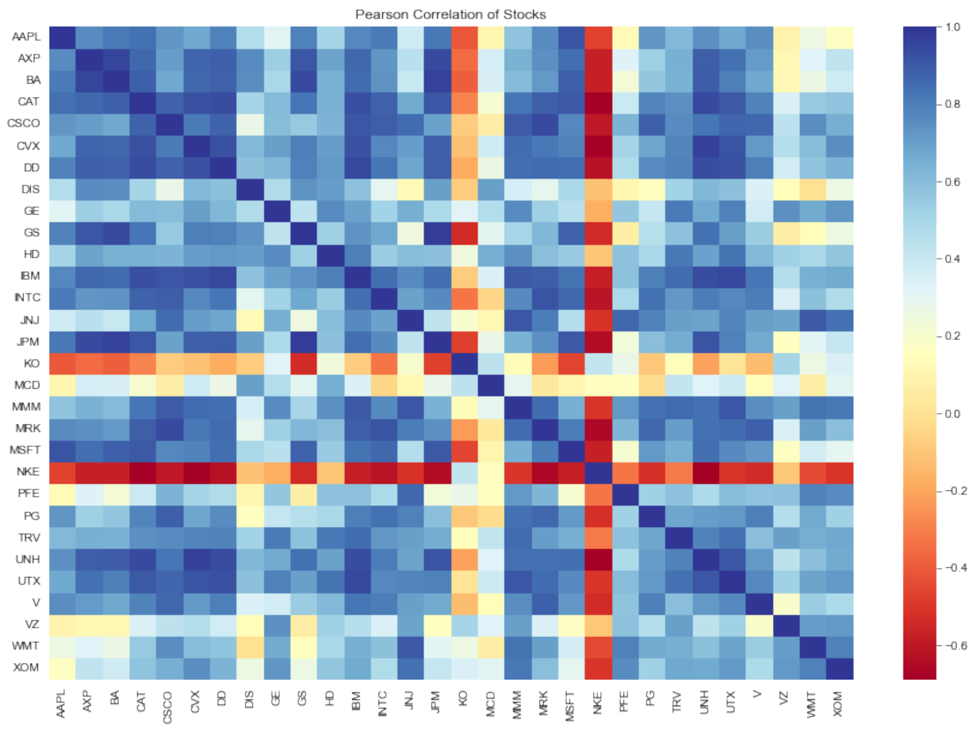


Figure 4: Heatmap of Pearson's correlation between stocks measured on 2016-2017 interval

To make the dataset distribution more closely approximate a normal distribution (and to remove the outliers), *Heq et al.* also used another dataset of a shorter period [37]. The authors hypothesize that the relationships are more stable over a shorter time frame. The chosen time frame specifically skips two significant past occurrences that significantly increased market instability: the 2008 global financial crisis and the 2011 U.S. debt ceiling crisis. Therefore, the second dataset was taken from January 2016 until February at 1-minute intervals. The table 2 shows the mean, median, range, standard deviation, variance, kurtosis, and skewness for each stock for this new period. The range of returns for most stocks is relatively smaller in the new observation period than in the full period. This indicates lower volatility in the stock prices during the new period. Additionally, the standard deviation of returns for most stocks was found to be generally higher in the first period as well, further emphasizing the higher level of volatility in the stock prices during this period. Furthermore, the kurtosis values are closer to zero in the new period, which implies that the returns of the second period have thinner tails than the distributions of the first period. The skewness values are generally also closer to zero in the new period, suggesting that the return distributions for most stocks were more symmetric than in the full period. The two latter statements imply that the distributions of returns for most stocks are closer to an observation period than the new period of observations.

Additionally, the results of Pearson's correlation for each stock pair in the new period are shown in figure 4 as a heatmap. The correlation coefficients in the new period tend to be lower than those in the full period. This suggests that they are less correlated in this shorter period. However, we can still see from the heatmap that, for many stocks, the correlation only decreased slightly.

This information can provide insight into which stocks may be influenced by similar factors and how changes in one stock's price may affect the prices of other stocks. It can also give an idea of the market's overall state and identify any related trends in the stock prices. However, it is essential to note that correlation does not imply causation, and other factors, such as market trends, economic conditions, and company-specific events, also affect stock prices.

3 Methods

In this study, we employed four approaches for connectivity estimation between time series data. Firstly, we fitted the data to a model similar to the VAR model (section 3.1). Secondly, we used a deep neural model that uses graphs to represent the market state (nodes are variables, and edges are the connections between them) (section 3.2). The third approach utilized another neural method where the weights of the initial layer of the deep network represent Granger connections (section 3.3). And finally, we introduce the combination of the third approach with another neural network baseline that is more suited for data with stochastic volatility (section 3.4).

3.1 t-VAR Model

The first model we used for connectivity estimation is called the t-VAR model, an approach from *Barbaglia et al.* [21]. It is a modification of the VAR model (introduced in section 2.2.1). Using a model for connectivity estimation that is closely related to the VAR model (used for data generation) may result in a better fit for the synthetic data generated from the VAR model.

Instead of modeling the noise, $\boldsymbol{\eta}_t$, as a Gaussian, they assume that the noise follows the Student-t distribution. Student t-distribution has wider tails, which allows for better modeling of outliers, such as stock market crashes [38]. This modification could potentially improve the model's performance on real-world data. In t-VAR, the model returns, $\boldsymbol{y_t}$, are defined as:

$$\boldsymbol{y_t} = \boldsymbol{\mu} + \boldsymbol{\Phi}(\boldsymbol{y_{t-1}} - \boldsymbol{\mu}) + \boldsymbol{e_t} \qquad \boldsymbol{e_t} \sim t(v, 0, \boldsymbol{\Psi}) \tag{11}$$

where $\boldsymbol{y_t} = (y_t^i)_{i=1,\dots,N}$ is a vector of N asset returns at time t, $\boldsymbol{\mu}$ is the vector of N means of asset returns, $\boldsymbol{\Phi} = (\phi_{ij})_{i,j=1,\dots,n}$ is a matrix of size $N \times N$ that contains autoregression coefficients, $\boldsymbol{e_t}$ is a noise that follows a Student-t distribution, $\boldsymbol{e_t} \sim t(v, \mathbf{0}, \boldsymbol{\Psi})$ where v is the degrees of freedom, $\mathbf{0}$ is the location vector and $\boldsymbol{\Psi} = (\psi_i)_{i,j=1,\dots,J}$ is the $J \times J$ scale matrix, where J is the number of variables.

In order to estimate the connections between the returns, one must estimate the values of the coefficient matrix, $\boldsymbol{\Phi}$. Since the noise, $\boldsymbol{e_t}$, follows a Student-t distribution, the estimation of $\boldsymbol{\Phi}$ is now more complicated. To that note, the authors use the Expectation Maximization (EM) algorithm. To simplify the algorithm, the authors reformulate the distribution of noise, $\boldsymbol{e_t}$, as a scale multivariate normal and introduce an extra parameter $\boldsymbol{\tau}_t$, $\boldsymbol{e_t}|\boldsymbol{\tau}_t \sim N(\mathbf{0}, \frac{\Psi}{\boldsymbol{\tau}_t})$, where $\boldsymbol{\tau}_t$ is a random variable distributed as *Gamma*, $\boldsymbol{\tau}_t|\boldsymbol{e_t} \sim \Gamma(\frac{v+J}{2}, \frac{v+\boldsymbol{e_t^T\Sigma^{-1}e_t}}{2})$, with J being the number of variables of t-VAR model. Furthermore, the $N \times N$ matrix $\boldsymbol{\tau} = (\boldsymbol{\tau}_t)_{t=1,\dots,N}$. The formulation of $\boldsymbol{e_t}$ through $\boldsymbol{\tau}$ simplifies the minimization objective of the M-step.

The goal of the EM algorithm is to estimate both $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$. The EM algorithm iterates between the E-step and the M-step until the estimations converge. During the E-step, the expected value of $\boldsymbol{\tau}_t$ is calculated as:

$$\widehat{\boldsymbol{\tau}_t}^{(m)} = E[\boldsymbol{\tau}_t^m|\boldsymbol{e_t}] = \frac{v+J}{v + {e_t}^T \Psi^{-1} e_t} = \frac{v+J}{v + \boldsymbol{y_t} - \boldsymbol{y_{t-1}}\boldsymbol{\Phi^{(m-1)}}\Psi^{-1}\boldsymbol{y_t} - \boldsymbol{y_{t-1}}\boldsymbol{\Phi^{(m-1)}}} \tag{12}$$

where $\boldsymbol{\Phi}^{(m-1)}$ is the estimated value of $\boldsymbol{\Phi}$ at a previous iteration of the EM algorithm. Thus, we have the vector $\widehat{\boldsymbol{\tau}}^{(m)} = (\widehat{\boldsymbol{\tau}}_t)^{(m)}_{t=1,\dots,N}$.

During the M-step, the authors estimate $\boldsymbol{\Phi}$. To ensure that the elements of this matrix can reach zeros, the authors impose a Lasso regularization on the values of $\boldsymbol{\Phi}$ during its estimation. The authors use maximum likelihood and jointly estimate $\boldsymbol{\Phi}$ and $\boldsymbol{\Sigma}$ by minimizing the negative log-likelihood. Simultaneously modeling the coefficient matrix, $\boldsymbol{\Phi}$ and the variance of the noise distribution, $\boldsymbol{\Sigma}$ can potentially help the model distinguish between the connections of returns and the contemporaneous correlations of their noise vector distributions. The minimization problem is given by:

$$\begin{aligned}(\widehat{\boldsymbol{\Phi}}^{(m)}, \widehat{\boldsymbol{\Psi}}^{(m)}|\widehat{\boldsymbol{\tau}}^{(m)}) = arg \min_{\boldsymbol{\Phi},\boldsymbol{\Psi}} \frac{1}{2N} tr[\widehat{\boldsymbol{\tau}}^{(m)1/2}(\boldsymbol{Y} - \boldsymbol{Y\Phi})\boldsymbol{\Psi}^{-1}(\widehat{\boldsymbol{\tau}}^{(m)1/2}(\boldsymbol{y_j} - \boldsymbol{Y\Phi}))^T] + \lambda \sum_{i,j=1}^{J} |\phi_{ij}| \\ + \gamma \sum_{i \neq j}^{J} |\psi_{ij}| \\ - \frac{1}{2} log|\boldsymbol{\Psi}|^{-1}\end{aligned} \tag{13}$$

where the first row of the equation comes from the Lasso estimation of $\mathbf{\Phi}$, where $\boldsymbol{Y} = (\boldsymbol{y_t})_{t=2,\dots,N}$ and $\boldsymbol{X} = (\boldsymbol{x_{t-1}})_{t=1,\dots,N-1}$, N is the total number of timesteps, $tr(\cdot)$ is the trace function of the matrix which is defined as the sum of its diagonal elements, and $\lambda > 0$ is the regularization parameter for Lasso. The second row of the equation ensures that the estimation of $\mathbf{\Psi}$ is feasible even if the number of parameters exceeds the time series length, and it also encourages some elements of $\widehat{\mathbf{\Psi}}$ to zero, with $\gamma > 0$ being the regularization parameter. The third row accounts for the contemporaneous correlation of noise generation. To estimate $(\widehat{\mathbf{\Phi}}^{(m)}, \widehat{\mathbf{\Psi}}^{(m)})$, in the M-step of the mth iteration the authors use Gaussian Lasso algorithm, after which the EM algorithm iteratively proceeds to the next iteration. The iterations repeat until the convergence of $(\widehat{\mathbf{\Phi}}, \widehat{\mathbf{\Psi}})$.

In conclusion, since this study evaluates the performance of connectivity estimation models on VAR-generated data (where the noise is sampled from a Gaussian distribution), the concern is whether this model can still perform well (as it makes an assumption that the noise is generated from a Student-t distribution). Since the t-distribution generalizes the Gaussian distribution, the t-VAR model should fit VAR-generated data. Furthermore, this study investigates the performance of connectivity estimation models on real-world data as well, where the noise distribution is not known. The Student-t noise assumption of the t-VAR model could make it fit the real-world data better than its counterparts (other connectivity estimation models used in this study). Additionally, the t-VAR model estimates both the scale and the coefficient matrix, allowing for simultaneous estimation of the scale matrix of the noise generation and the coefficient matrix. Finally, the t-VAR model stands out as the only non-deep learning model considered in this study, making a comparison of its performance with that of its deep learning counterparts both interesting and valuable.

3.2 Neural Relational Inference

3.2.1 Preliminary Concept: Graph Neural Network

A graph is a data structure made of vertices (data samples) that are connected by edges (relationships between samples). Edges can be directed (one-sided connection) or undirected (mutual connection). Graphs do not exist in Euclidean space, making them difficult to analyze. Examples include social networks, physical systems, molecules, knowledge graphs, etc. This raises the question of whether it is possible to model time series variables as graph nodes and the connections between them as graph edges.

Graph neural networks (GNNs) are neural networks that are specialized for handling graph data [39]. They can be applied to perform node-level, edge-level, or graph-level predictions [40, 41, 42]. Since our goal is to estimate connections (edges), we focus on edge-level predictions. GNNs use two key operations: node-to-edge message passing and edge-to-node message passing.

Node-to-edge message passing, as shown in equation 14, calculates the edge embedding, $\boldsymbol{h^l_{(i,j)}}$:

$$\text{v}\to\text{e:}\ \boldsymbol{h}^l_{(i,j)} = f^l_e([\boldsymbol{h}^l_i, \boldsymbol{h}^l_j, \boldsymbol{x}_{(i,j)}]) \tag{14}$$

where $\boldsymbol{h}^l_{(i,j)}$ is the edge embedding vector between the nodes i and j at the layer l, f^l_e is the lth neural network edge layer, $\boldsymbol{h}^l_i$ and $\boldsymbol{h}^l_j$ are the node embeddings at layer l, and $\boldsymbol{x}_{(i,j)}$ is vector of initial the initial edge features.

Edge-to-node message passing, as shown in equation 15, calculates the node embedding, $\boldsymbol{h}^{l+1}_j$:

$$\text{e}\to\text{v:}\ \boldsymbol{h}^{l+1}_j = f^l_v([\sum_{i\in N_j} \boldsymbol{h}^l_{(i,j)}, \boldsymbol{x}_j]) \tag{15}$$

where $\boldsymbol{h}^{l+1}_j$ is the embedding vector of node j at layer $l+1$, f^l_v is the lth neural network node layer, $\sum_{i\in N_j} \boldsymbol{h}^l_{(i,j)}$ is the sum of all N_j edge embeddings connecting to node j, and $\boldsymbol{x}_j$ is the vector of initial node features.

In the case of GNN, the edge and node functions, f_e and f_v, can be either fully connected layers or any other layers. These functions are the building blocks for the neural relational inference model (NRI), which uses node-to-edge and edge-to-node message-passing layers.

3.2.2 Preliminary Concept: Variational Autoencoder

Autoencoders (AEs) are neural networks with two sub-networks: an encoder and a decoder [43]. The encoder compresses the data samples into a lower-dimensional latent representation, while the decoder reconstructs the data from its lower-dimensional representation back to its original dimension. The goal is that the input to the encoder should be as close as possible to the output of the decoder, ensuring that the latent space captures enough information to reconstruct the original data samples. However, a drawback of vanilla AE is the lack of structure in the latent space, which can result in similar data points being far apart, regardless of their semantic meaning.

Variational autoencoders (VAE) address this issue by enforcing a prior distribution on the latent space, consequentially turning the latent representation into a latent distribution [44]. The loss function used to achieve this is called evidence lower bound (ELBO) loss, and it is defined as the sum of the negative log-likelihood and the Kullback-Leibler (KL) divergence:

$$L = -\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] + KL[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})] \tag{16}$$

where q_ϕ is the encoder network with parameters ϕ, p_θ is the decoder network with parameters θ, $\boldsymbol{x}$ is the input vector to the encoder, $\boldsymbol{z}$ is a lower-dimensional latent representation vector. The negative log-likelihood term, $-\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]$, denotes the expectation (or average) taken over the distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. This can have a form of a mean-square error (MSE) loss or some variation of MSE loss. The KL term, $KL[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})]$, measures the similarity between the latent distribution and its predefined prior.

The neural relational inference (NRI) model uses a VAE-based architecture where the latent space is a distribution of graphs. This model is discussed in the next section.

3.2.3 Explanation of NRI

The Neural Relational Inference (NRI) model, introduced by *Kipf et al.*, is a deep learning model that learns interactions of variables by forecasting observational data [18]. It combines a variational autoencoder (VAE) and a graph neural network (GNN), allowing it to infer interactions between time series variables. In the context of financial data analysis, the NRI model can potentially capture the interactions between variables and uncover more complex relationships (maybe even the ones generated from the SV-AR model or found in real-world data).

The architecture itself is similar to the one from VAE, with some key differences. The input to the NRI is the vector of returns at a previous timestep, $\boldsymbol{y_{t-1}}$, and its output is the vector of returns at a current timestep $\boldsymbol{y_t}$. However, the latent space is a graph encoding the connections between returns. It should also be noted that this graph does not contain self-edges, which means that the NRI is not able to estimate self-connections (connection from the future timestep to the previous timestep of the same variable). Nevertheless, for the latent space to represent graphs, the encoder and the decoder alternate between node-to-edge and edge-to-node message-passing layers. The encoder consists of three layers, as shown in equations 17 - 20. The equations show the encoder message passing for one return, y_j. The same set of equations is repeated for all returns, $\boldsymbol{y}_j$. The first encoder layer is an embedding layer that transforms returns into their node embeddings:

$$y_j \rightarrow \text{v:}\ \boldsymbol{h}_j^1 = f_{emb}(y_{j,t-1}) \tag{17}$$

where $y_{j,t-1}$ is the return j at time $t-1$, f_{emb} is a fully connected layer that projects the input into the node embedding, h_j^1 is the first node embedding of the return y_j.

The second encoder layer is a node-to-edge message-passing layer:

$$\text{v}\rightarrow\text{e:}\ \boldsymbol{h}_{(i,j)}^1 = f_e^1([\boldsymbol{h}_i^1, \boldsymbol{h}_j^1]) \tag{18}$$

where $\boldsymbol{h}_i^1$ and $\boldsymbol{h}_j^1$ are node embeddings of returns i and j, f_e^1 is the first node-to-edge message-passing layer, and $\boldsymbol{h}_{(i,j)^1}$ is the first edge embedding of returns i and j.

The third encoder layer is the edge-to-node message-passing layer:

$$\text{e}\rightarrow\text{v:}\ \boldsymbol{h}_j^2 = f_v^1(\sum_{i\neq j} \boldsymbol{h}_{(i,j)}^1) \tag{19}$$

where $\sum_{i \neq j} \boldsymbol{h}^1_{(i,j)}$ is the sum of embeddings of all of the edges that connect to node j, f^1_v is the first edge-to-node message-passing layer, and $\boldsymbol{h}^2_j$ is the second node embedding of the return j.

The fourth and final encoder layer is another node-to-edge message-passing layer:

$$\text{v}\rightarrow\text{e: } \boldsymbol{h}^2_{(i,j)} = f^2_e([\boldsymbol{h}^2_i, \boldsymbol{h}^2_j]) \tag{20}$$

where $\boldsymbol{h}^2_i$ and $\boldsymbol{h}^2_j$ are node embeddings of returns i and j, f^2_e is the second node-to-edge message-passing layer, and $\boldsymbol{h}_{(i,j)^2}$ is the output edge embedding of returns i and j.

In a typical VAE, the encoder outputs the parameters of the distribution (for example, a mean and a covariance matrix in the case of a Gaussian distribution), but this is not the case for NRI. The latent variables, $\boldsymbol{h}^2_i$, are normalized probabilities for the discrete latent variables representing the interaction graph. In order to convert these logits into a continuous approximation of the discrete distribution over the latent variables was used. Continuous approximation is a method used to represent a discrete random variable as a continuous random variable. In NRI, the authors used the Gumbel-Softmax trick (type of continuous approximation), which is defined as:

$$q_\phi(\boldsymbol{z}_{ij}|x) = softmax(\frac{\boldsymbol{h}^2_{(i,j)} + \boldsymbol{g}}{\tau}) \tag{21}$$

where $\boldsymbol{h}^2_{(i,j)}$ are the logits outputted from the encoder, $\boldsymbol{\tau}$ is a smoothness factor, $\boldsymbol{g}$ is a random sample from Gumbel distribution, $\boldsymbol{g} \sim Gumbel(0, 1)$, and $\boldsymbol{z}_{ij}$ are edges of a latent graph with the introduced stochasticity that are used for the input to the decoder.

The task of the decoder is to predict the return of a current timestep, y_t, while given the return at a past timestep, y_{t-1}, and the graph of interactions outputted from the encoder, $\boldsymbol{Z} = (z_{ij})_{i,j=1,\ldots,N}$. Firstly, the returns are transformed to their edge embeddings with node-to-edge message passing layer:

$$\text{v}\rightarrow\text{e} : \tilde{\boldsymbol{h}}^3_{(i,j)} = \sum_k z_{ij} \tilde{f}^k_e([\boldsymbol{y}_i, \boldsymbol{y}_j]) \tag{22}$$

where z_{ij} is an edge representation from node i to node j, $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are returns at timestep $t-1$, $\tilde{f}^k_e$ is the decoder node-to-edge message passing layer, and $\tilde{\boldsymbol{h}}^3_{(i,j)}$ is an edge representation. Finally, since the node representations are needed for the final predictions, the final edge-to-node message-passing layer is defined as:

$$\text{e}\rightarrow\text{v} : \widehat{y}_{j,t} = \tilde{f}_v(\sum_{i \neq j} \tilde{h}^t_{(i,j)}) \tag{23}$$

where $\sum_{i \neq j} \tilde{h}^t_{(i,j)}$ is the sum of all embeddings of all of the edges that connect to node j, $\tilde{f}_v$ is the decoder edge-to-node message-passing layer, and $\widehat{y}_{j,t}$ is the return of asset j at a current timestep.

In order to train NRI (so it can obtain correct estimations of y_t and, ultimately, the correct estimations of the connections), the ELBO loss function was used. ELBO loss function is defined as the sum of the negative log-likelihood and the KL term. The negative log-likelihood term is defined as:

$$-\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] = -\sum_j^J \sum_{t=2}^T ||\widehat{y}_t - y_t||^2 \tag{24}$$

where $\widehat{y}_t$ is the estimated return at time t, y_t is the true return, T is the number of timesteps, J is the number of variables, and the result is an MSE loss.

The other part of the ELBO loss function, the KL term, is defined as:

$$KL[q_\phi(z|x)||p_\theta(z)] = \sum_{i \neq j} H(q_\phi(z_{ij}|x)) \tag{25}$$

where $q_\phi(z_{ij}|x)$ is the output of encoder.

After calculating the ELBO loss, gradients are computed, and the weights of the model are updated using the Adam optimizer [45]. When the ELBO loss converges, the latent graph represents the estimated connections.

To recapitulate, the NRI model uses the combination of VAE and GNN to estimate connections between returns in the form of a graph represented in the latent space. It uses deep layers that could fit the complex data well (such as real-world stock data). On the other hand, it is more likely to overfit on the simpler dynamics, such as the synthetic data generated from the VAR model. Unlike t-VAR, NRI cannot estimate the noise distribution.

3.3 Neural Granger Causality Models

The Neural Granger Causality (NGC) model, introduced by *Tank et al.* [19], is a deep learning-based approach for identifying Granger causality between time series. It is particularly effective at detecting Granger causality in nonlinear and complex systems, where traditional methods, like the t-VAR model (section 3.1), may fall short due to their linear assumptions. However, the NGC can use a variety of deep neural networks to fit the data and estimate connectivity, and therefore, its performance heavily depends on the choice of deep learning architecture.

The authors of NGC propose a component-based deep learning setting where there is one neural network, g_i, for every variable of the time series. Each network takes as input the returns at a current timestep and predicts the value of the returns at a future timestep of one variable, as follows:

$$y_{t+1,i} = g_i(\boldsymbol{y_t}) + \eta_{t,i} \tag{26}$$

where $y_{t+1,i}$ is the return at time $t+1$ of variable i, g_i is a neural network for that variable, $\boldsymbol{y_t} = (y_t)_{t=1,\dots,N}$ represents the vector of N returns, and $\eta_{t,i}$ is the additive noise at time t for variable i.

In this study, we experimented with three existing NGC model architectures: component-wise multilayer perceptron (cMLP) with one layer, component-wise multilayer perceptron (cMLP) with two layers, and component-wise long short-term memory network (cLSTM). MLPs are particularly well suited for modeling recursive time series data where the value of the future timestep explicitly depends on the value of a previous time step, such as the synthetic data generated from the VAR model. In order to test the effect of having multiple layers, we have experimented with both one-layer and two-layer cMLPs. The one-layer and two-layer cMLPs are defined as:

$$\text{1 layer cMLP: } y_{t+1,i} = \boldsymbol{W}_i^1 \cdot \boldsymbol{y_t} + b_i^1 + \eta_{t,i} \tag{27}$$

$$\text{2 layer cMLP: } y_{t+1,i} = \boldsymbol{W}_i^2 \cdot \text{ReLU}(\boldsymbol{W}_i^1 \cdot \boldsymbol{y_t} + b_i^1) + b_i^2 + \eta_{t,i} \tag{28}$$

where $\boldsymbol{W_i}^1, \boldsymbol{W_i}^2$ are matrices of the first and second layer of variable i respectively, b_i^1, b_i^2 are biases of the first and second layer of variable i respectively, and ReLU is a rectified linear unit activation function [46].

Another NGC deep learning architecture that we used is cLSTM. It is effective for modeling time series data where the future value implicitly depends on its previous value, such as in the case where the data is synthetically generated from the SV-AR model. To model this implicit dependency, NGC uses LSTMs hidden states $(\boldsymbol{c_t, h_t})$ [47]. LSTM is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem encountered when training deep neural networks [48]. This is a well-known problem faced by deep neural networks where the gradient becomes too small during backpropagation, hindering the training process. By using a series of gates, LSTMs can effectively capture long-term dependencies in the data. The LSTM cell is defined by the following set of equations:

$$\text{Input gate: } i_t = \sigma(\boldsymbol{W_i}[\boldsymbol{y_t}, \boldsymbol{h_{t-1}}] + \boldsymbol{b_i}) \tag{29}$$

$$\text{Forget gate: } f_t = \sigma(\boldsymbol{W_f}[\boldsymbol{y_t}, \boldsymbol{h_{t-1}}] + \boldsymbol{b_f}) \tag{30}$$

$$\text{Cell state update: } g_t = \tanh(\boldsymbol{W_c}[\boldsymbol{y_t}, \boldsymbol{h_{t-1}}] + \boldsymbol{b_c}) \tag{31}$$

$$\text{Cell state: } \boldsymbol{C_t} = f_t \cdot \boldsymbol{C_{t-1}} + i_t \cdot g_t \tag{32}$$

$$\text{Output gate: } y_{t+1} = \sigma(\boldsymbol{W_o}[\boldsymbol{y_t}, \boldsymbol{h_{t-1}}] + \boldsymbol{b_o}) \tag{33}$$

$$\text{Hidden state: } \boldsymbol{h_t} = y_{t+1} \cdot \tanh(\boldsymbol{C_t}) \tag{34}$$

where $\boldsymbol{y_t}$ is the input vector of size N at time step t, $\boldsymbol{h_{t-1}}$ is the hidden state at the previous time step, and σ is the sigmoid activation function. The input gate i_t determines how much new information should be stored in the cell state, while the forget gate f_t determines how much information should be discarded from the cell state. The cell state updates g_t is the new candidate information that could be stored in the cell state, and the output gate o_t determines how much of the cell state should be used to compute the hidden state $\boldsymbol{h_t}$.

After fitting the data to either cMLP or cLSTM, the connectivity from a variable j to i is estimated to be true if there still exists a non-zero input weight on input j in the neural network, g_j. In order to ensure that the input weights are able to reach zeros, the Lasso penalty is defined for each network as the following optimization process:

$$\min_{\boldsymbol{W}} \sum_{t=1}^{T} (y_{i,t} - g_i(\boldsymbol{y_{t-1}}))^2 + \lambda \sum_{j=1}^{J} |w_{i,j}^1| \tag{35}$$

where $\lambda > 0$ is the regularization term, and J is the number of variables.

To further ensure that exact zeros are reached, the authors have used proximal gradient descent [49]. Proximal gradient descent first performs a gradient update on every layer of the neural network and then a proximal update on the input layer. Proximal gradient descent update of $\boldsymbol{W}$ is defined as:

$$\boldsymbol{W}^{(m+1),1} = prox_{\gamma\lambda\omega}(\boldsymbol{W}^{(m),1}) = ReLU(1 - \frac{\lambda\gamma}{|\boldsymbol{W}^{(m),1}|}) \cdot \boldsymbol{W}^{(m),1} \tag{36}$$

where $\boldsymbol{W}$ is the $J \times H$ matrix of input weights where J is the number of variables, and H is the number of hidden layers (or 1 in case of the last layer), $\gamma > 0$ is the learning rate, $\lambda > 0$ is the Lasso regularization term and $|\boldsymbol{W}^1| = \sqrt{\sum_i^J \sum_j^H |w_{ij}|^2}$ is the Forbenius norm of the matrix of the input layer. The proximal step further ensures that the weights reach zeros because, for $\gamma\lambda > |\boldsymbol{W}^{(m),1}|$, the weight matrix is updated to zero.

In summary, the NGC model incorporates the expressiveness of deep networks to predict the connections between the returns. In this study, we conducted experiments using one-layer cMLP, two-layer cMLP, and cLSTM models to determine whether multiple layers are necessary for cMLP models and to explore the importance of implicit dependence in cLSTM models. However, unlike the t-VAR model, the NGC model does not estimate the noise distribution parameters. In terms of scalability, the NGC model can be easily scaled when using small deep network architectures. However, for many variables, it may need a large number of simultaneously employed networks, increasing computational demands. Nonetheless, the NGC model demonstrates the most efficient scaling when compared to the other connectivity estimation models considered in this study. Finally, concerning possible model biases, the combination of group Lasso and proximal gradient descent could make this model overestimate sparse coefficient matrices.

3.4 Component-wise Neural Stochastic Volatility Model

Finally, as the last connectivity estimation model used in this research, we introduce a novel model, the component-wise neural stochastic volatility model (cNSVM). It combines the approaches from the neural stochastic volatility model (introduced by *Luo et al.* [20]) and the neural Granger causality model (section 3.3, [19]). The former research was utilized as the authors introduce a novel deep learning architecture more suited for the data with stochastic volatility, such as the stochastic volatility model from section 2.2.2. The latter research was utilized to estimate connections between two time series (section 3.3).

The deep learning architecture used from NSVM is compromised out of 2 deep learning networks: the inference network and the generative network. The goal of the inference network is to model the latent volatility process, $q_\psi(Z|X)$ (where Z is the volatility process, and X is the input series), as the factor of the Gaussian distribution with the following equations:

$$q_\Psi(Z|X) = \prod_{t=1}^{T} q_\Psi(z_t|\boldsymbol{z}_{<t}, \boldsymbol{y}_{1:T}) \tag{37}$$

$$q_\Psi(Z|X) = \prod_{t=1}^{T} N(z_t; \tilde{\mu}_\Psi^z(\boldsymbol{z}_{<t}, x_{1:T}), \tilde{\Sigma}_\Psi^z(\boldsymbol{z}_{<t}, \boldsymbol{x}_{1:T})) \tag{38}$$

where q_Ψ is the volatility distribution of z_t dependant on the previous volatility values, $\boldsymbol{z}_{<t}$, and the returns, $\boldsymbol{y}_{1:T}$. $\tilde{\mu}_\psi^z$ is the mean and $\tilde{\Sigma}_\psi^z$ is the covariance matrix of the Gaussian distribution. Both of these values are determined from both previous volatilities, $z_{<t}$ and datapoints, $x_{1:T}$.

These parameters are the outputs of the inference network. The inference network is implemented as a cascade of a bidirectional RNN [50], an autoregressive RNN, and one multilayer perception (MLP_I^Z), and outputs the mean and the covariance matrix of the latent distribution, z_t:

$$\tilde{h}_t^{\rightarrow} = RNN_I^{\rightarrow}(\tilde{h}_{t-1}^{\rightarrow}, x_{t-1}; \Psi) \tag{39}$$

$$\tilde{h}_t^{\leftarrow} = RNN_I^{\leftarrow}(\tilde{h}_{t+1}^{\leftarrow}, x_{t+1}; \Psi) \tag{40}$$

$$\tilde{h}_t^z = RNN_I^z(\tilde{h}_{t-1}^z, [z_{t-1}, \tilde{h}_t^{\rightarrow}, \tilde{h}_t^{\leftarrow}]; \Psi) \tag{41}$$

$$\tilde{\mu}_t^z, \tilde{\Sigma}_t^z = MLP_I^z(\tilde{h}_t^z; \Psi) \tag{42}$$

$$z_t \sim N(\tilde{\mu}_t^z, \tilde{\Sigma}_t^z; \Psi) \tag{43}$$

where $RNN_I^{\rightarrow}$ is the first RNN that takes a datapoint at a past timestep, x_{t-1}, as input and outputs its hidden state, $\tilde{h}_t^{\rightarrow}$. $RNN_I^{\leftarrow}$ takes as input a datapoint from a future timestep, x_{t+1}, and outputs its hidden state, $\tilde{h}_t^{\leftarrow}$. Both of these hidden states are concatenated together with the value of the volatility at a previous timestep, z_{t-1}, and used as an input to the third RNN, RNN_I^z. Finally, the hidden state of the third RNN, $\tilde{h}_t^z$ is used as an input to the MLP_I^z that outputs the mean, $\tilde{\mu}_t^z$, and the covariance matrix $\tilde{\Sigma}_t^z$ of volatility at the current time step, z_t. The sample of this distribution was used as an input to the generative network.

The generative network models the distribution of returns as a Gaussian distribution dependent on the latent volatility process:

$$p_\Phi(X|Z) = \prod_{t=1}^{T} p_\Phi(x_t|x_{t-1}, z_t) \tag{44}$$

$$p_\Phi(X|Z) = \prod_{t=1}^{T} N(x_t; \mu_\Psi^x(y_{t-1}, z_t), \Sigma_\Psi^x(x_{t-1}, z_t)) \tag{45}$$

where p_Φ is a Gaussian distribution of datapoints, y_t, with parameters, μ_Ψ^x and Σ_Ψ^x that are dependant on the value of a previous datapoint, x_{t-1} and the current sample of the volatility process, z_t. The generative network is implemented as a cascade of an RNN and an MLP:

$$h_t^x = RNN_G^x(h_{t-1}^x, [x_{t-1}, z_t]; \Psi) \tag{46}$$

$$\tilde{\mu}_t^x, \tilde{\Sigma}_t^x = MLP_G^x(h_t^x; \Psi) \tag{47}$$

$$x_t \sim N(\tilde{\mu}_t^x, \tilde{\Sigma}_t^x) \tag{48}$$

where RNN_G^x is an RNN that takes as input the datapoint at a previous timestep, x_{t-1} concatenated with the volatility at a current timestep, z_t. The hidden state of this RNN, h_t^x, is used as an input to an MLP, MLP_G^x that outputs the mean, $\tilde{\mu}_t^x$ and the covariance matrix, $\tilde{\Sigma}_t^x$ of the normally distributed current datapoints, x_t.

In summary, the architecture of NSVM consists of two stacked deep networks. The first one, the inference network, estimates the distribution of the latent volatility process, $q_\Psi(Z|X)$, and the second one, the generative network, estimates the distribution of the datapoints, $p_\Phi(X|Z)$.

Also, regarding the implementation details of this architecture, we have used the same approaches as in the original paper. More specifically, we have also used dropout on each RNN module as a regularization technique to prevent overfitting [51]. Furthermore, the latent space, Z, is 4 dimensional, and the hidden states of an RNN, h, are 10 dimensional. Also, as for the specific architecture of RNN cells, we have used GRU cells as they have fewer parameters than LSTMs but still show state-of-the-art results [52]. Finally, as opposed to estimating the Gaussian mean (which is straightforward), estimating the Gaussian covariance matrix requires postprocessing of the outputs to ensure that the resulting output is positive and semi-definite, which is a constraint that any covariance matrix needs to satisfy. To construct a positive semi-definite matrix from the aforementioned MLP outputs, we use an MLP to output the values on the upper triangular matrix. Then, this upper triangular matrix is summed with its own transpose to acquire a symmetric matrix, and the resulting symmetric matrix is multiplied with its own transpose to acquire a positive semi-definite matrix, which we use as an estimated covariance matrix. As for the loss function, we have used an ELBO loss, which is computed as a sum of the MSE of the predicted datapoints and a KL divergence loss imposed on the latent space.

The NSVM model provides a good theoretical ground for fitting the data with a latent stochastic volatility process. Therefore, we combine the approach used in section 3.3 and introduce cNSVM. cNSVM model consists of p NSVM architectures, where p is the number of variables of the multivariate dataset. Every NSVM architecture takes as input all of the datapoints at time t, $\boldsymbol{x}_{t-1}$, but each of them outputs one variable. Also, each NSVM architecture has a fully-connected initial layer. This is because this fully-connected layer needs to perform variable selection.

Furthermore, we use L2 regularization on every layer other than the first one. For the first one, we use group Lasso regularization. We use Lasso since we want the values to be able to reach close to 0, and more specifically, a group adaptation of the Lasso regularization since the input to the NSVM model is x_{t-1} and x_{t+1} and we want to group the weights depending on the variable.

In conclusion, we propose the cNSVM model, which combines the advantages of the neural stochastic volatility model and neural Granger causality model (section 3.3), enabling to better fit the data with stochastic volatility. Similar to other NGC models, the cNSVM model does not estimate any parameters of the noise distribution. Finally, regarding scalability, it is expected to scale efficiently in a manner similar to other NGC models, although it may require additional computational resources due to its deeper layers.

4 Results

In this section, we present the results of the experiments where we estimated the connectivity of financial data with different connectivity estimation models. The experiments have been performed on the data generated from the vector autoregressive model (VAR, section 4.1), the stochastic volatility model (SV-AR, section 4.2), and the historical stock price dataset (section 4.3). We estimated connectivity with six different connectivity estimation models (presented in section 3): t-VAR, NRI, NGC with 1-layer cMLP, NGC with 2-layer cMLP, NGC with cLSTM, and NGC with cNSVM. For clarity, we refer to these models using the following abbreviations: t-VAR, NRI, cMLP-1, cMLP-2, cLSTM, and cNSVM.

Each model outputs a binary adjacency matrix representing connections between variables. In the experiments with synthetically created data using VAR and SV-AR models, we measured accuracy as the percentage of correctly estimated elements in the adjacency matrix. Since the historical stock price dataset does not have ground-truth labels for the estimations, we compared the outputs of our models with the results from three different models presented in *Hecq et al.* [37]. We calculated the similarity between their models and ours as the percentage of connections estimated by our model that were also estimated by models in *Hecq et al* [37].

Some models may produce varying results based on different seed values for the random number generator, resulting in inconsistent performance. To ensure a more reliable comparison of performances between different models, we trained and evaluated each model $N_{runs} = 5$ times with different seed values. We then calculated the mean and standard deviation of testing accuracies to compare the performance of different model evaluations. The standard deviation provides insight into the stability of the model's outputs. Therefore, this section aims to provide the results of the experiments to determine the advantages and disadvantages of these models.

To that note, the results are presented in bar plots, with the mean accuracy (or mean similarity in the case of the historical stock price dataset) and the standard deviation on the y-axis, and different sub-experiments on the x-axis. Each bar plot is accompanied by a table displaying the details of different sub-experiments.

Before stating the results of the experiments, it is important to note the values of the hyperparameters used during this research. For noise distribution in t-VAR, we used 15 degrees of freedom. For NRI, we used 256 nodes in the encoder and decoder layers and a learning rate of 0.005. For the NGC-based models, we used 100 nodes for its hidden layers, a learning rate of 0.005, a Lasso coefficient of 0.002, a Ridge coefficient of 0.01, and 10 nodes for the recurrent layers.

4.1 Data Generated from VAR Model

In this section, we examined the performance of connectivity estimation models on the synthetic datasets generated from the VAR model. We estimated connectivity with four models: t-VAR, NRI, cMLP-1, and cMLP-2.

4.1.1 Preliminary: Experiment 1

Goal: To ensure a fair comparison in future experiments with synthetic data generated by the VAR model (experiments 4.1.2 - 4.1.5), all models must be trained under the same conditions (this experiment does not yield any insights related to the research questions posed in the Introduction, section 1). Therefore, it is necessary to determine the dataset size needed to train these models. Other than only determining the dataset size, they should also be tested under the same conditions. To that note, it is also important to determine the percentage of the dataset used for testing these models. The goal of Experiment 1 is to do just that; determine the dataset size (the number of timesteps created by the VAR model) and the train/test ratio of that dataset that produces the most stable results. These choices should yield a small standard deviation over different random seed initializations. A small standard deviation ensures the reliability of model performance.

Setup: In this experiment, we first trained the connectivity estimation models on five different dataset sizes: 500, 1000, 2000, 3000, and 4000. To ensure the model is also unbiased towards the train/test split, we train the models with four different train/test ratios. Then we repeated this process five times with different seed values for the random number generator. Thus, each model was trained 100 times (5 dataset sizes, 4 train/test ratios, and 5 different seed values). They were trained on the VAR data generated from the coefficient matrix of $\mathbf{\Phi} = \begin{bmatrix} 0.8 & 0.8 \\ 0 & 0.8 \end{bmatrix}$. Figure 5 shows the models' performances based on different dataset sizes, and figure 6 shows the models' performances based on different train/test ratios.

Results: In figure 5, we see different models' performances based on various dataset sizes. t-VAR, NRI, and cMLP-1 demonstrate consistent performance for all five dataset sizes, as they all have a standard deviation of 0. However, the standard deviation of cMLP-2 is 0 only when the dataset is generated with 3000 or 4000 timesteps. Since having a smaller dataset is computationally more efficient, we have used the dataset size of 3000 for conducting future experiments on the data generated by the VAR model.
Furthermore, figure 6 shows the performances of different models with different train/test ratios: 0.6, 0.7, 0.8, and 0.9. We can see that, again, t-VAR, NRI, and c-MLP have the same performance throughout different ratios and that cMLP-2 is the only model whose standard deviation varies. The mean of its accuracy remains the same, but the standard deviation is the smallest for the ratios of 0.8 and 0.9. We chose the train/test ratio of 0.8 because having a smaller train set implies fewer computations of trainable parameters. It also implies having a larger test set, which produces more reliable results (as the models were tested on more timesteps). Therefore, we proceeded to the following experiments with a dataset of 3000 timesteps and a train/test ratio of 0.8 in the next sections, where we experimented with the VAR model.

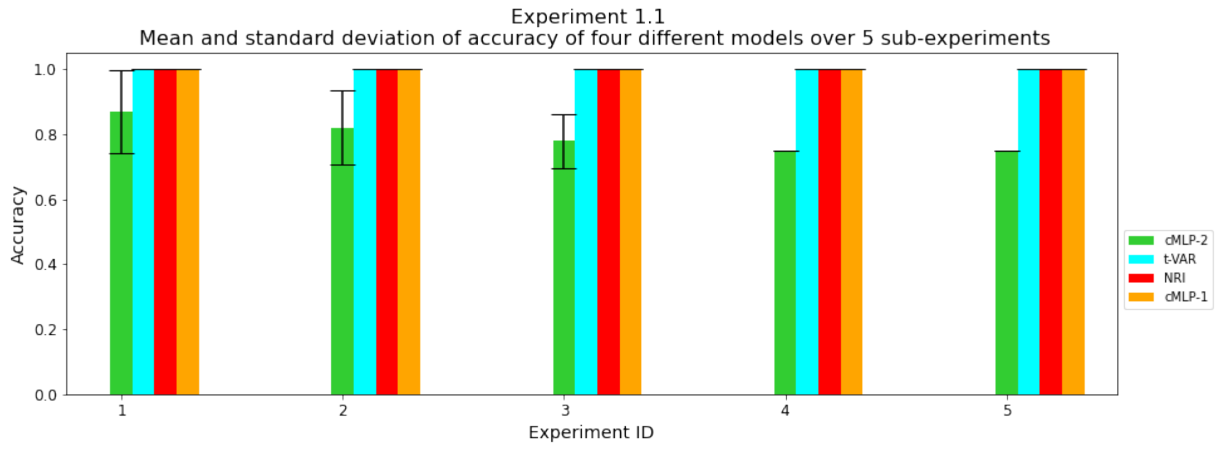


Figure 5: Accuracy of different models with VAR-generated data and different training subset sizes in table 3

Experiment ID	Dataset size
1.1.1	500
1.1.2	1000
1.1.3	2000
1.1.4	3000
1.1.5	4000

Table 3: Dataset sizes for experiments from figure 5

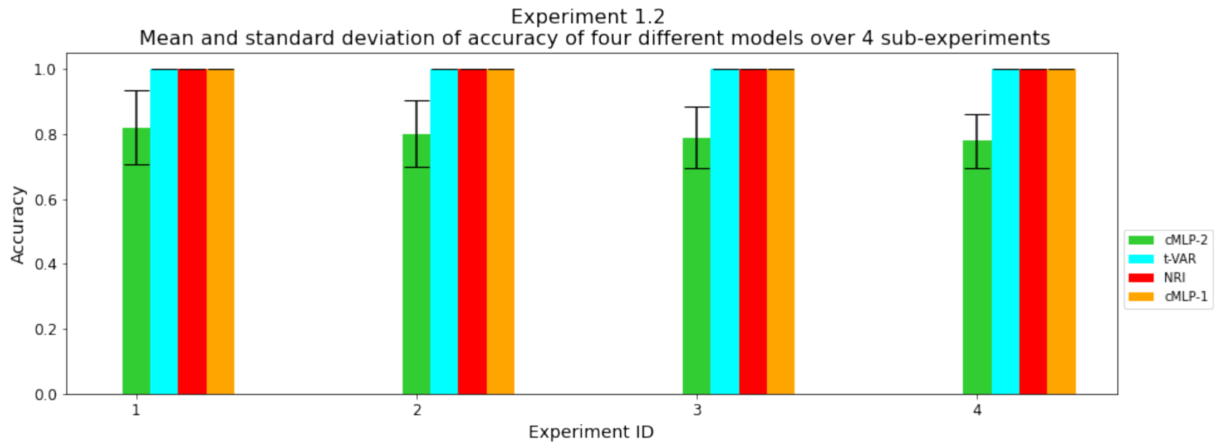


Figure 6: Performance of different models with VAR-generated data and different train/test ratios in table 4

Experiment ID	Train/test ratio
1.2.1	0.6
1.2.2	0.7
1.2.3	0.8
1.2.4	0.9

Table 4: Train/test ratios for experiments from figure 6

4.1.2 Experiment 2

Goal: In the second experiment, we tested the models' performance on the data with two variables and different strengths of connections. Specifically, we examined the performance of models depending on different coefficient matrices, $\mathbf{\Phi}$, of the VAR model.

Setup: There are four possible connectivity patterns that the coefficient matrix $\mathbf{\Phi}$ can represent:

1. No connection between variables; $\mathbf{\Phi} = \begin{bmatrix} \neq 0 & = 0 \\ = 0 & \neq 0 \end{bmatrix}$

2. Connection from variable 1 to variable 2; $\mathbf{\Phi} = \begin{bmatrix} \neq 0 & = 0 \\ \neq 0 & \neq 0 \end{bmatrix}$

3. Connection from variable 2 to variable 1; $\mathbf{\Phi} = \begin{bmatrix} \neq 0 & \neq 0 \\ = 0 & \neq 0 \end{bmatrix}$

4. Mutual connection between variables: $\mathbf{\Phi} = \begin{bmatrix} \neq 0 & \neq 0 \\ \neq 0 & \neq 0 \end{bmatrix}$

The values of the non-zero elements in the $\mathbf{\Phi}$ matrix influence the strength of the connections. Larger diagonal elements indicate stronger autocorrelation, meaning that the value of a variable at time t is more dependent on its value at time $t-1$. Greater off-diagonal elements represent stronger connections between variables. Thus, this experiment evaluates the models' performance in all four scenarios with varying diagonal element values in the $\mathbf{\Phi}$ matrix.

Results: Figures 7 - 10 present the results of the second experiment. Figure 7 displays the models' performance for different on-diagonal values when variables are not connected. In subexperiment 1.1.1, we see that none of the models can perform well when there is no self-connection (connection from the future timestep to the previous timestep of the same variable). In this case, the generated data is purely Gaussian noise. NRI appears to perform best, but since it cannot estimate self-connections, its 80% mean accuracy should be interpreted as correctly identifying the absence of connections between variables 80% of the time. In contrast, the other models, which do estimate self-connections, achieve

better overall performance. For small diagonal values of $\mathbf{\Phi}$, these models struggle to estimate self-connections, but their performance improves as the on-diagonal values increase.
Figures 8 and 9 show the models' performance when variable 1 is connected to variable 2 and vice versa. The results exhibit similar behavior to the previous subexperiment, with all models performing well when the connection strength is sufficiently large. NRI can achieve 100% accuracy when the connection strength is 0.4 or higher, outperforming cMLP-2 for smaller connection strengths. However, cMLP-2 still performs better than NRI for weaker connections, detecting them more accurately. t-VAR and cMLP-1 achieve the best performance, largely because their formulations are very similar to the VAR model used to generate the data. The difference between these models' formulations is that t-VAR uses a Student-t distribution instead of a Gaussian distribution for modeling noise, and cMLP-1 does not capture the noise component of the VAR model.
Figure 10 presents the performance of the models when both variables are connected. Similar to the other subexperiments in this section, the results indicate that the models can recognize connections if the connection strength is large enough. NRI exhibits the poorest performance, while t-VAR and cMLP-1 demonstrate the best performance among the four models. This outcome is consistent with the findings of the other subexperiments, highlighting the importance of connection strength in determining model performance.

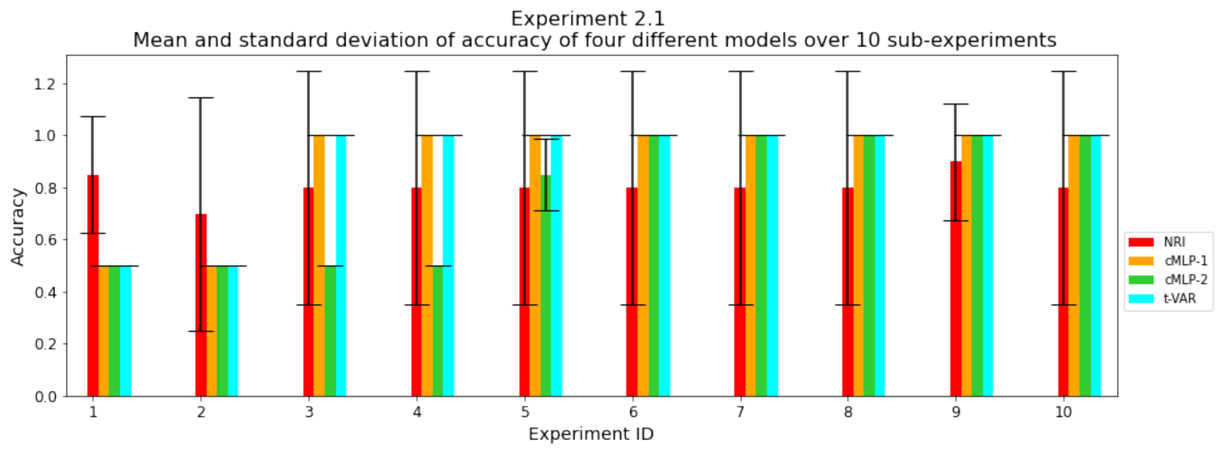


Figure 7: Performance of different models with VAR-generated data of two variables and coefficient matrices with no connections between variables from table 5

Experiment ID	$\mathbf{\Phi}$
2.1.1	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
2.1.2	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$
2.1.3	$\begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$
2.1.4	$\begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix}$
2.1.5	$\begin{bmatrix} 0.4 & 0 \\ 0 & 0.4 \end{bmatrix}$
2.1.6	$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$
2.1.7	$\begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$
2.1.8	$\begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}$
2.1.9	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$
2.1.10	$\begin{bmatrix} 0.9 & 0 \\ 0 & 0.9 \end{bmatrix}$

Table 5: Values of coefficient matrix $\mathbf{\Phi}$ for experiments from figure 7

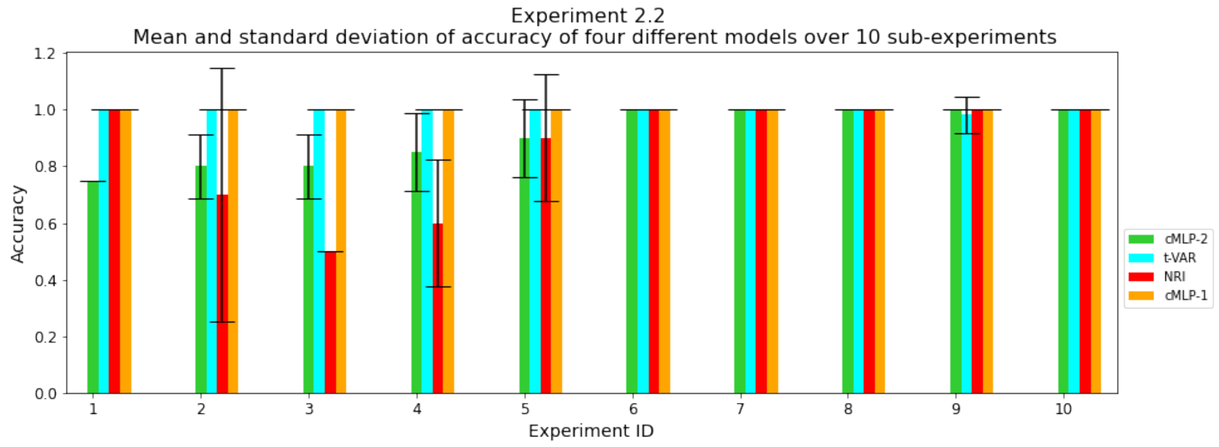


Figure 8: Performance of different models with VAR-generated data of two variables and coefficient matrices with a connection from variable 1 to variable 2 from table 6

Experiment ID	$\mathbf{\Phi}$
2.2.1	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$
2.2.2	$\begin{bmatrix} 0.8 & 0.1 \\ 0 & 0.8 \end{bmatrix}$
2.2.3	$\begin{bmatrix} 0.8 & 0.2 \\ 0 & 0.8 \end{bmatrix}$
2.2.4	$\begin{bmatrix} 0.8 & 0.3 \\ 0 & 0.8 \end{bmatrix}$
2.2.5	$\begin{bmatrix} 0.8 & 0.4 \\ 0 & 0.8 \end{bmatrix}$
2.2.6	$\begin{bmatrix} 0.8 & 0.5 \\ 0 & 0.8 \end{bmatrix}$
2.2.7	$\begin{bmatrix} 0.8 & 0.6 \\ 0 & 0.8 \end{bmatrix}$
2.2.8	$\begin{bmatrix} 0.8 & 0.7 \\ 0 & 0.8 \end{bmatrix}$
2.2.9	$\begin{bmatrix} 0.8 & 0.8 \\ 0 & 0.8 \end{bmatrix}$
2.2.10	$\begin{bmatrix} 0.8 & 0.9 \\ 0 & 0.8 \end{bmatrix}$

Table 6: Values of coefficient matrix $\mathbf{\Phi}$ for experiments from figure 8

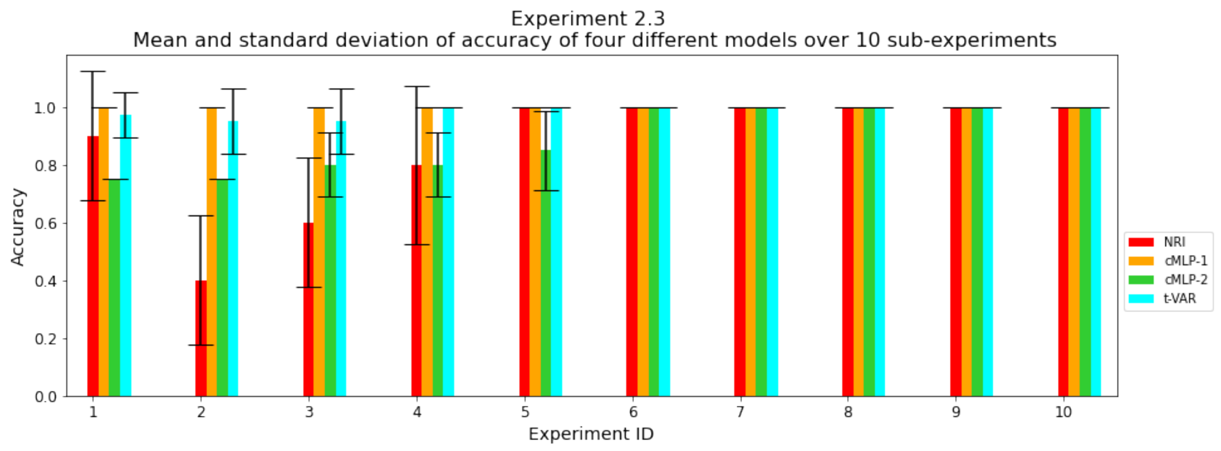


Figure 9: Performance of different models with VAR-generated data of two variables and coefficient matrices with a connection from variable 2 to variable 1 from table 7

Experiment ID	$\mathbf{\Phi}$
2.3.1	$\begin{bmatrix}0.8 & 0\\0 & 0.8\end{bmatrix}$
2.3.2	$\begin{bmatrix}0.8 & 0\\0.1 & 0.8\end{bmatrix}$
2.3.3	$\begin{bmatrix}0.8 & 0\\0.2 & 0.8\end{bmatrix}$
2.3.4	$\begin{bmatrix}0.8 & 0\\0.3 & 0.8\end{bmatrix}$
2.3.5	$\begin{bmatrix}0.8 & 0\\0.4 & 0.8\end{bmatrix}$
2.3.6	$\begin{bmatrix}0.8 & 0\\0.5 & 0.8\end{bmatrix}$
2.3.7	$\begin{bmatrix}0.8 & 0\\0.6 & 0.8\end{bmatrix}$
2.3.8	$\begin{bmatrix}0.8 & 0\\0.7 & 0.8\end{bmatrix}$
2.3.9	$\begin{bmatrix}0.8 & 0\\0.8 & 0.8\end{bmatrix}$
2.3.10	$\begin{bmatrix}0.8 & 0\\0.9 & 0.8\end{bmatrix}$

Table 7: Values of coefficient matrix $\mathbf{\Phi}$ for experiments from figure 9

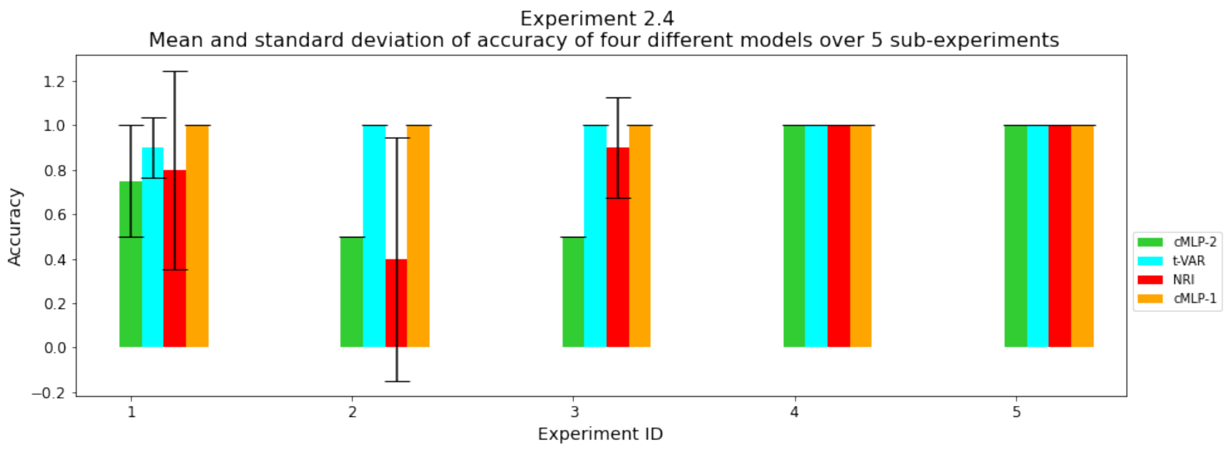


Figure 10: Performance of different models with VAR-generated data of two variables and coefficient matrices with a mutual connection between variables from table 8

Experiment ID	$\mathbf{\Phi}$
2.4.1	$\begin{bmatrix} 0.8 & 0.1 \\ 0.1 & 0.8 \end{bmatrix}$
2.4.2	$\begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$
2.4.3	$\begin{bmatrix} 0.8 & 0.3 \\ 0.3 & 0.8 \end{bmatrix}$
2.4.4	$\begin{bmatrix} 0.8 & 0.4 \\ 0.4 & 0.8 \end{bmatrix}$
2.4.5	$\begin{bmatrix} 0.8 & 0.5 \\ 0.5 & 0.8 \end{bmatrix}$

Table 8: Values of coefficient matrix $\mathbf{\Phi}$ for experiments from figure 10

4.1.3 Experiment 3

Goal: The objective of the third experiment was to evaluate the models' ability to differentiate between contemporaneous correlation in the correlation terms of the noise covariance matrix, $\mathbf{\Sigma}_\eta$, and the connections encoded in the coefficient matrix, $\mathbf{\Phi}$, of the VAR model.

Setup: The experiment involved introducing a correlation between the noise generation processes of the variables, then removing the correlation and repeating the experiments. The hypothesis was that if a model is robust to contemporaneous correlation, its accuracy would not decrease when noise correlation is introduced. We tested noise covariance values of 0.49, 0.09, and 0.04, with the covariance being as high as possible while adhering to the positive semidefinite constraint (required for any covariance matrix). Each setup was repeated five times with different seed values, resulting in 30 trains per model.

Results: Figure 11 displays the results of this experiment. From the figure, it is evident that the NRI model performs poorly for this coefficient matrix and should not be considered when drawing conclusions about this experiment. NRI's performance remains similar to the setting without noise correlation. In contrast, t-VAR demonstrates complete resistance to the added correlation in the noise distribution, likely because it models the noise distribution simultaneously while estimating the coefficient matrix. However, NGC-based models, cMLP-1 and cMLP-2, also perform well, although they are not entirely resilient to larger noise. Their performance is somewhat worse in examples with larger noise compared to those with smaller noise. Nonetheless, we can observe only a slight increase in accuracies from subexperiment 1 to subexperiment 2, as well as from 3 to 4 and 5 to 6, because subexperiments 2, 4, and 6 have no noise correlation. This indicates that these models are almost entirely resistant to the contemporaneous correlation in the noise generation matrix but not to the overall larger noise. Therefore, it can be concluded that the models can distinguish between contemporaneous correlation in the correlation terms of the covariance matrix of the noise generation process, $\mathbf{\Sigma}_\eta$, and the connectivity encoded in the coefficient matrix, $\mathbf{\Phi}$, of the VAR model. This is because none of them exhibit a significant decline in performance when the contemporaneous correlation in the noise distribution is introduced.

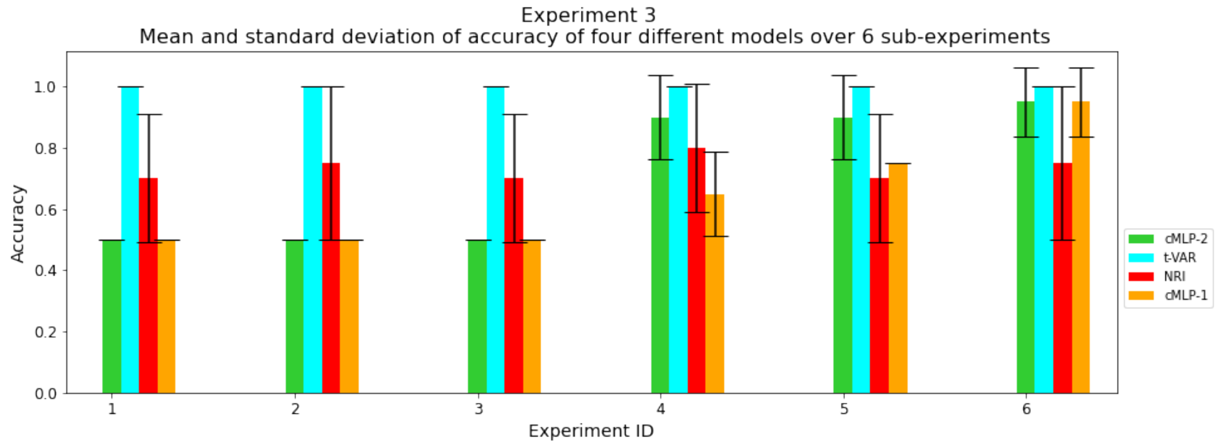


Figure 11: Performance of different models with VAR-generated data and different noise covariances from table 9

Experiment ID	$\mathbf{\Phi}$	$\mathbf{\Sigma}$
3.1	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 0.4 \\ 0.4 & 0.5 \end{bmatrix}$
3.2	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$
3.3	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0.09 \\ 0.09 & 0.1 \end{bmatrix}$
3.4	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$
3.5	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.05 & 0.04 \\ 0.04 & 0.05 \end{bmatrix}$
3.6	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}$

Table 9: Values of different noise co-variances for experiments from figure 11

4.1.4 Experiment 4

Goal: Experiment 4 aimed to evaluate the connectivity estimation models' performance for high-dimensional data in terms of both estimation accuracy and training time. This experiment compares the models based on their accuracy of estimation as well as their training time. In high-dimensional datasets, the performance of the models might stay the same, but if the model takes too long to train, then it cannot be used on real-world high-dimensional data (such as portfolios containing numerous stocks).

Setup: We tested these models on 2 to 8 variables and repeated the process 5 times with different seed values, resulting in 35 runs per model in total. As we increased the number of variables, we randomly initialized the positions of non-zero elements in matrix $\mathbf{\Phi}$. The sparsity of all coefficient matrices was 50%. This means that 50% of the variables are connected, and 50% of the variables are not connected. Furthermore, all of the non-zero values in $\mathbf{\Phi}$ are the same and as large as possible, subject to the stationarity constraint (necessary for all time series in this study).

Results: Figure 12 reveals that the models' performance does not change based on the number of variables. However, table 10 highlights a significant difference in models' training times. NRI takes the longest time to train. This is expected because, as discussed in section 3.2, for every new variable in the dataset, a new node needs to be added to the fully connected graph, which severely increases the complexity of the model. cMLP-1 and cMLP-2 tend to scale the best. t-VAR scales poorly, as its performance is almost 22 times slower when going from 2 to 8 variables. This is expected because the complexity of NGC-based models increases linearly when new variables are added. On the other hand, t-VAR is an EM-based algorithm, and they are known to scale poorly because of the complex calculations performed during the E-step and the M-step (section 3.1). Even though NRI has the longest runtime, the poor scalability of t-VAR's runtime suggests that adding many more variables would cause t-VAR's runtime duration to surpass the one of NRI, making it challenging to use in real-life scenarios with numerous variables.

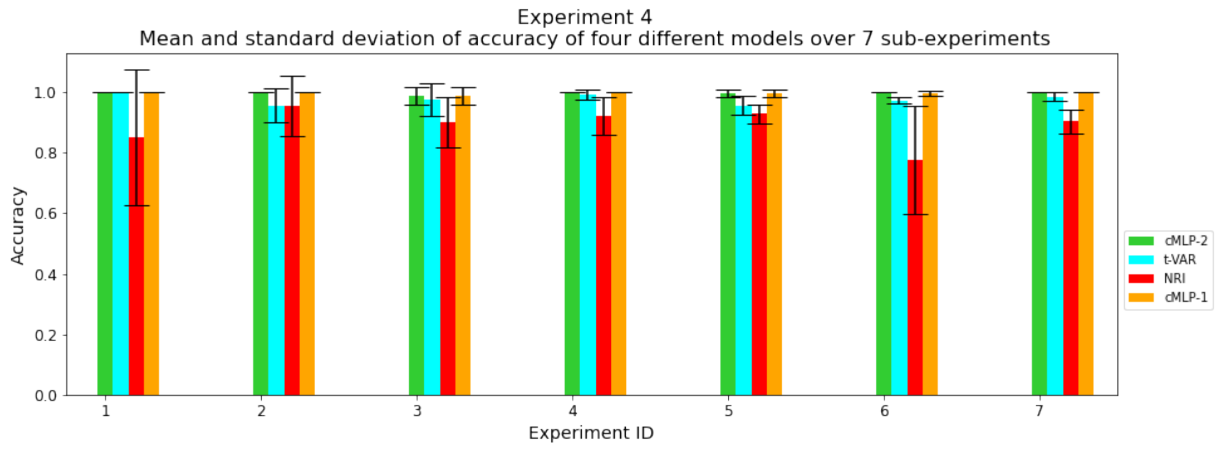


Figure 12: Performance of different models with VAR-generated data and different number of variables from table 10

Experiment ID	Number of variables	Average time per run per model (minutes)			
		NRI	**NGC-0**	**NGC**	**t-VAR**
4.1	2	3.617	2.15	3.32	0.48
4.2	3	5.24	2.95	5.75	0.93
4.3	4	6.81	3.29	7.13	1.62
4.4	5	8.165	4.95	9.51	3.11
4.5	6	11.53	5.8	11.19	3.88
4.6	7	14.96	7.17	12.15	6.97
4.7	8	18.18	8.39	13.42	10.48

Table 10: Values of different numbers of variables and average time duration for experiments from figure 12

4.1.5 Experiment 5

Goal: In the fifth experiment, we wanted to test if estimating connections is easier than estimating non-connections. That is, we wanted to check for the existence of a possible bias toward estimating connections (or non-connections). This is especially important to check for the NGC-based models, such as cMLP-1 and cMLP-2, as they initially start the training with all connections present and gradually remove connections as training progresses.

Setup: We generated a high-dimensional dataset with 8 variables and differently sparsed coefficient matrices, $\mathbf{\Phi}$, of the VAR model. We tested matrices with 1 to 8 connections per variable, maintaining the same number of connections per variable for each subexperiment. The number of connections per variable is the same per one subexperiment. If the sparsity is 0.25, it implies that every variable is connected to 2 other variables (including itself). This results in 8 different trains per model, with 5 repetitions for random seed, consequentially resulting in a total of 40 trains per model.

Results: Figure 13 shows the results of this experiment. NRI's performance decreases as the sparsity increases. With fewer connections between variables, NRI's estimations worsen. This further shows that the NRI model is biased towards estimating connections compared to estimating non-connections. However, other models do not exhibit the same behavior. cMLP-1, cMLP-2, and t-VAR achieve excellent performance (around 100%) no matter the sparsity of VAR's coefficient matrix, $\mathbf{\Phi}$. This suggests that NRI struggles with increased numbers of variables, predominantly estimating no connections between them. In contrast, the other models do not suffer from this issue. Their performance remains consistent as more connections are introduced, indicating no bias towards either estimating connections or non-connections.

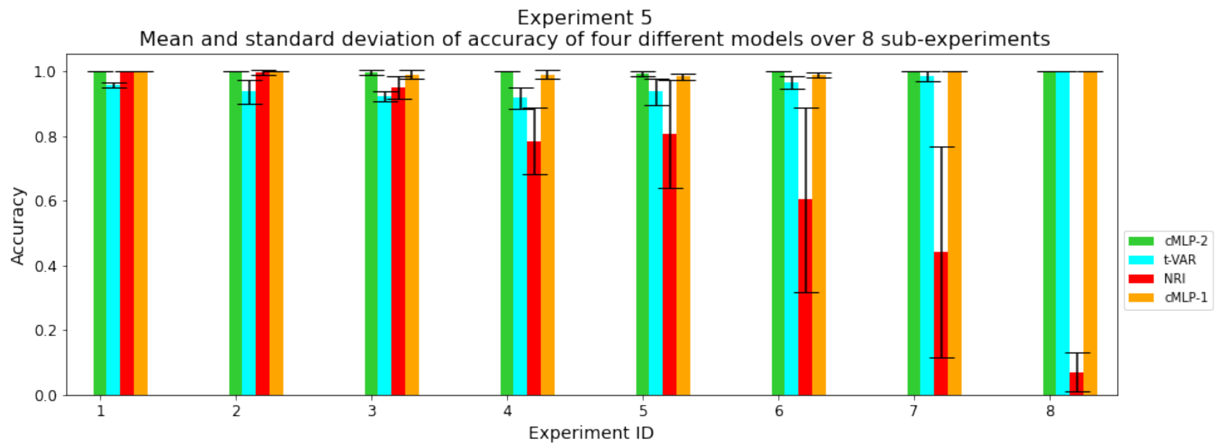


Figure 13: Performance of different models with VAR-generated data with 8 variables and differently sparsed coefficient matrices from table 11

Experiment ID	Percentage of connections between variables (%)
5.1	12.5
5.2	25
5.3	37.5
5.4	50
5.5	62.5
5.6	75
5.7	87.5
5.8	100

Table 11: Values of differently sparsed coefficient matrices with 8 variables for experiments from figure 13

4.2 Data Generated from Stochastic Volatility Model

After evaluating connectivity estimation models on the data generated from the VAR model, we proceeded to test them on the data generated from the SV-AR model. In this set of experiments, we only used two models: cLSTM and the novel model we have introduced, cNSVM.

Before running the experiments on the synthetic datasets generated by the SV-AR model, we first wanted to justify why we did not use the connectivity estimation models presented in the previous section. As shown in figure 14, these models perform poorly on the few samples of SV-AR-generated data. All models have an accuracy of 50% with a high standard deviation. An accuracy of 50% is the lowest possible and indicates that the models have failed to fit the data, leading them to randomly determine connectivities. Therefore, for the synthetic data generated by the SV-AR model, we only experimented with cLSTM and cNSVM connectivity estimation models as they provide a more robust theoretical foundation for fitting SV-AR-generated data. The experimental setup of this section corresponds to the experimental setup with the data generated by the VAR model in the previous section (section 4.1).

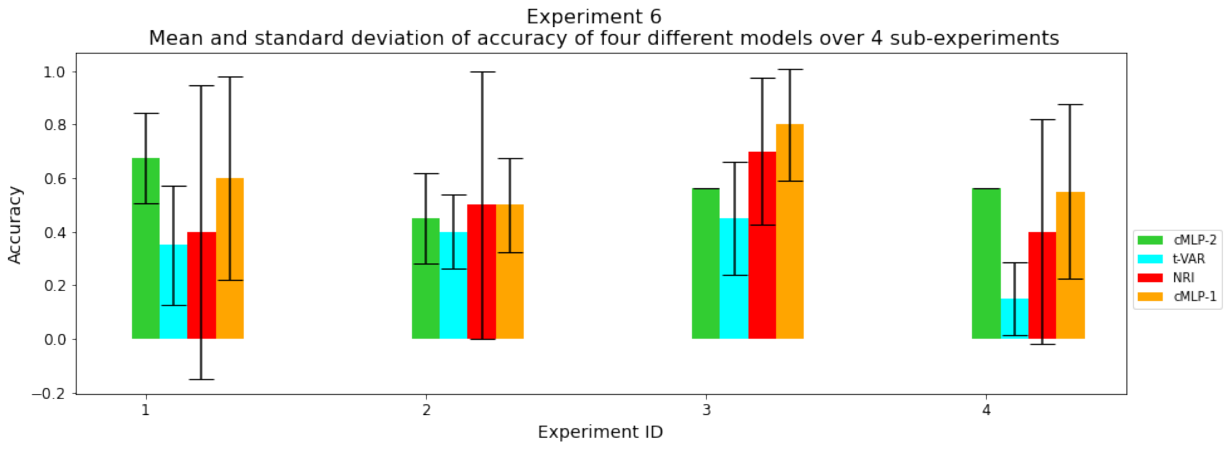


Figure 14: Performance of different models with SV-AR-generated data with 2 variables from coefficient matrices from table 12

Experiment ID	$\mathbf{\Phi}$
6.1	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$
6.2	$\begin{bmatrix} 0.8 & 0.4 \\ 0 & 0.8 \end{bmatrix}$
6.3	$\begin{bmatrix} 0.8 & 0 \\ 0.4 & 0.8 \end{bmatrix}$
6.4	$\begin{bmatrix} 0.8 & 0.1 \\ 0.1 & 0.8 \end{bmatrix}$

Table 12: Values of coefficient matrices with 2 variables for experiments from figure 14

4.2.1 Preliminary: Experiment 7

Goal: The goal of this experiment was to determine the dataset size and train/test ratio required to run these models (it does not yield any insights related to the research questions posed in the Introduction, section 1). To compare the models fairly, we trained them under the same conditions, with the same dataset size and the same train/test ratio. We wanted to use the smallest dataset size for which the models give stable results (the standard deviation is the smallest), as increasing the size of the dataset increases the computational time. Additionally, we wanted the smallest possible train/test ratio that yields stable results (the smallest standard deviation) because having a smaller train/test ratio means that the model needs fewer data points to fit the parameters (which results in less compute time), and more data for model evaluation.

Setup: To investigate this, we employed the same experimental setup as in experiment 1 (section 4.1.1). We trained both models using five different dataset sizes: 500, 1000, 2000, 3000, and 4000. When training with a specific dataset size, we repeated the experiment with four train/test ratios: 0.6, 0.7, 0.8, and 0.9. To account for the stability of the results, we repeated this process 5 times with five different seeds for the random number generators. In total, each model was trained 100 times. Finally, we averaged the results based on the dataset size and, then, based on the train/test ratio to get unbiased results. They were trained on the SV-AR data generated from the coefficient matrix $\mathbf{\Phi} = \begin{bmatrix} 0.8 & 0.8 \\ 0 & 0.8 \end{bmatrix}$ and the noise covariance matrix $\mathbf{\Sigma}_{\boldsymbol{\eta}} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$.

Results: In figure 15, we can see different models' performances based on different dataset sizes. cLSTM consistently achieves the same result regardless of the dataset, with an accuracy of 75%. On the other hand, cNSVM's performance generally declines as the size of the dataset increases, along with its standard deviation. This indicates a more stable training process. Furthermore, cNSVM shows the same performance for the datasets of 3000 and 4000 timesteps. Therefore, we have, again,

determined that the size of 3000 would be sufficient for training these models, and we proceeded with further experimentation with the aforementioned dataset size.
Figure 16 shows models' performance based on different train/test ratios. Again, cLSTM achieves a consistent accuracy of around 75% regardless of the train/test ratio. However, cNSVM attains the highest, most stable accuracy for the last two train test ratios. Therefore, we selected the train/test ratio of 0.8 for future experiments.

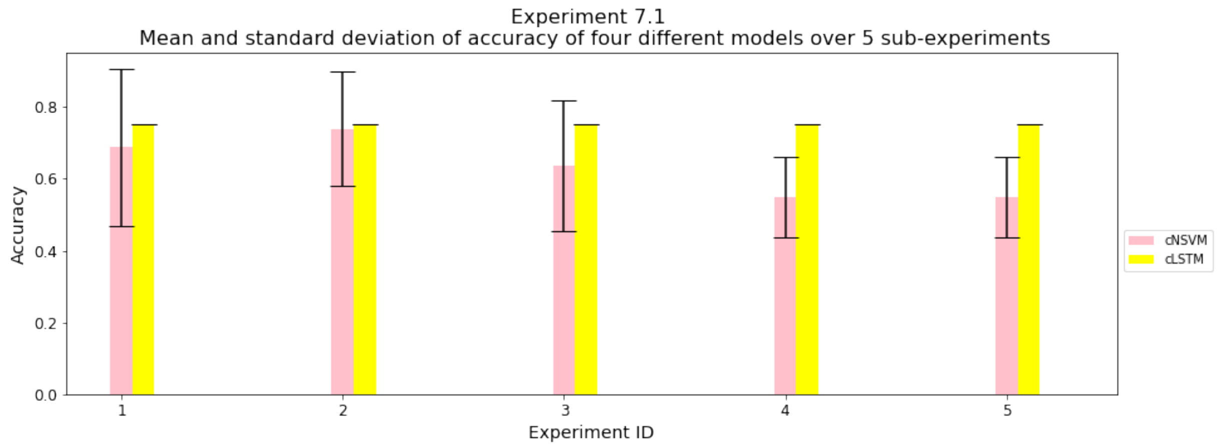


Figure 15: Accuracy of different models with SV-AR-generated data and different training subset sizes in table 13

Experiment ID	Dataset size
7.1.1	500
7.1.2	1000
7.1.3	2000
7.1.4	3000
7.1.5	4000

Table 13: Dataset sizes for experiments from figure 15

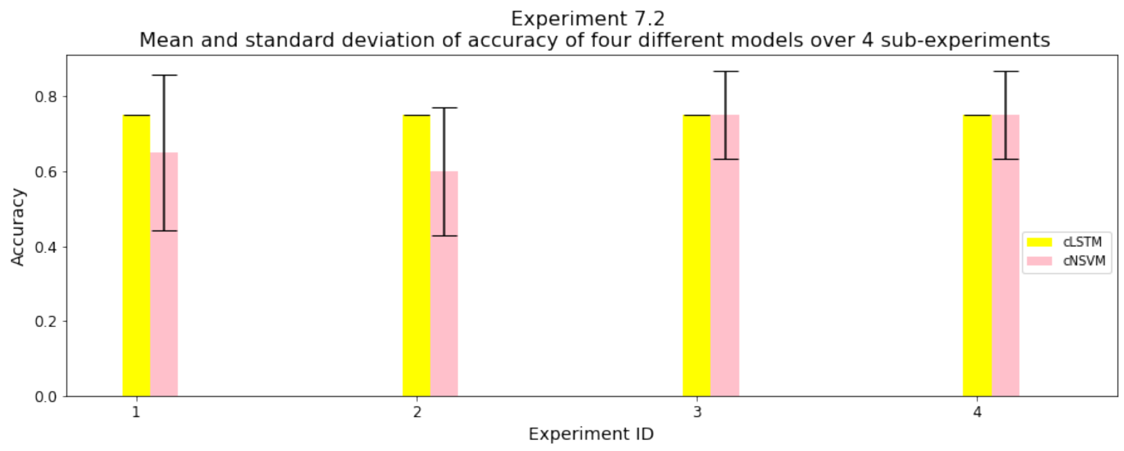


Figure 16: Performance of different models with SV-AR-generated data and different train/test ratios in table 14

Experiment ID	Train/test ratio
7.2.1	0.6
7.2.2	0.7
7.2.3	0.8
7.2.4	0.9

Table 14: Train/test ratios for experiments from figure 16

4.2.2 Experiment 8

Goal: In experiment 8, we assessed these models on the data with two variables and varying connection strengths between the volatilities of the SV-AR model. This was done to investigate how well do models perform based on different coefficient matrices of the SV-AR model.

Setup: The experimental setup of experiment 8 corresponds to the experimental setup of experiment 8 (section 4.1.2). We extensively tested the models for numerous combinations of the coefficient matrix, $\mathbf{\Phi}$ of the SV-AR model:

1. No connection between variables; $\mathbf{\Phi} = \begin{bmatrix} \neq 0 & = 0 \\ = 0 & \neq 0 \end{bmatrix}$
2. Connection from variable 1 to variable 2; $\mathbf{\Phi} = \begin{bmatrix} \neq 0 & = 0 \\ \neq 0 & \neq 0 \end{bmatrix}$
3. Connection from variable 2 to variable 1; $\mathbf{\Phi} = \begin{bmatrix} \neq 0 & \neq 0 \\ = 0 & \neq 0 \end{bmatrix}$
4. Mutual connection between variables: $\mathbf{\Phi} = \begin{bmatrix} \neq 0 & \neq 0 \\ \neq 0 & \neq 0 \end{bmatrix}$

Results: In figure 17, we see models' performance for different on-diagonal values when there is no connectivity between the variables. Both models perform poorly in the case when the data is generated by random noise (subexperiment 8.1.1). Instead of estimating no connectivity in the coefficient matrix, both models estimate full connectivity. This, again, shows how the models perform poorly on random data. Furthermore, we see that cLSTM gives an accuracy of 50% for all other variations. This indicates that cLSTM cannot perform well when there is no connectivity between variables (as it estimates full connectivity between variables). However, cNSVM performs with an accuracy of 50% throughout this entire subexperiment, 8.1. Its results have a high standard deviation. This indicates that cNSVM is also not able to estimate the data when there is no connection between variables. To conclude, neither cLSTM nor cNSVM can accurately recognize when there is no connection between the variables.
In figures 18 and 19, we see the models' performance when there is only one connection between

variables. Figure 18 shows the performance of the models when there is a connection from variable 1 to variable 2, and figure 19 shows the performance of the models when there is a connection from variable 2 to variable 1. It is clear that the models cannot achieve good performance (that is close to 100%). Both models result in an accuracy close to 75%. Given that there are only four connections, this is a poor score. When the off-diagonal value of the coefficient matrix is 0.2 (or higher), the models can estimate the third connection, but when the off-diagonal value of the coefficient matrix is 0.1, the models do not recognize it as a connection. Overall, as the standard deviation of cLSTM is always 0, it shows that cLSTM does not produce any results with 100%, unlike cNSVM.
Lastly, in figure 20, we see the models' performance when there is a mutual connection between the variables. In this scenario, cLSTM performs better than cNSVM. When the values on the off-diagonal element of the coefficient matrix are larger than 0.1, it performs with a consistent 100% accuracy. On the other hand, the mean accuracy of cNSVM is around 80% and is more unstable. For the case of low value on the off-diagonal element of the coefficient matrix, neither of the models performs well.
In summary, this experiment shows that the connectivity estimation models cannot perform well on the SV-AR-generated data. Although they show signs of learning, overall, they cannot perform as well as t-VAR, cMLP-1, cMLP-2, and NRI did on VAR-generated data. Furthermore, even though there have been numerous runs where cNSVM performed better than cLSTM, overall, it shows a slightly worse performance than cLSTM. On top of that, the overall performance of cLSTM throughout this experiment suggests that it might be biased toward estimating connections instead of non-connections. In experiment 8.1, where there are 2 out of 4 connections between variables, it has an overall accuracy of around 50%; in experiments 8.2 and 8.3, where there are 3 out of 4 connections between variables, it has an overall accuracy of around 75%, and in experiment 8.4 where there are 4 out of 4 connections it shows accuracy on two examples of 100%. However, it is also certain that cLSTM does not always estimate complete connectivity, as there are examples to contradict this, such as 8.1.1, 8.2.1, 8.2.2, 8.3.1, 8.3.2, 8.4.1, and 8.4.2.

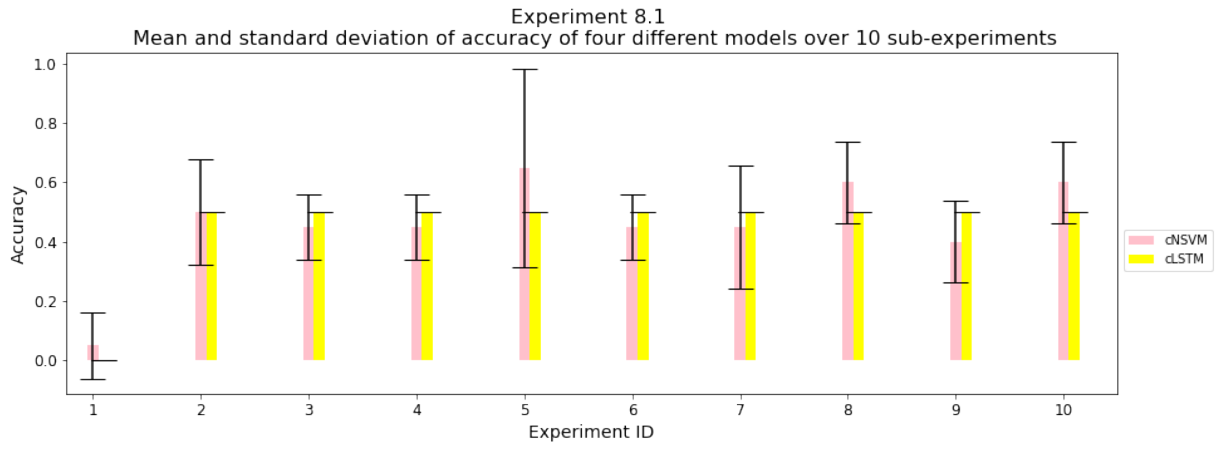


Figure 17: Performance of different models with SV-AR-generated data of two variables and coefficient matrices with no connections between variables from table 15

Experiment ID	$\mathbf{\Phi}$
8.1.1	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
8.1.2	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$
8.1.3	$\begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$
8.1.4	$\begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix}$
8.1.5	$\begin{bmatrix} 0.4 & 0 \\ 0 & 0.4 \end{bmatrix}$
8.1.6	$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$
8.1.7	$\begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$
8.1.8	$\begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}$
8.1.9	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$
8.1.10	$\begin{bmatrix} 0.9 & 0 \\ 0 & 0.9 \end{bmatrix}$

Table 15: Values of coefficient matrix $\mathbf{\Phi}$ for experiments from figure 17

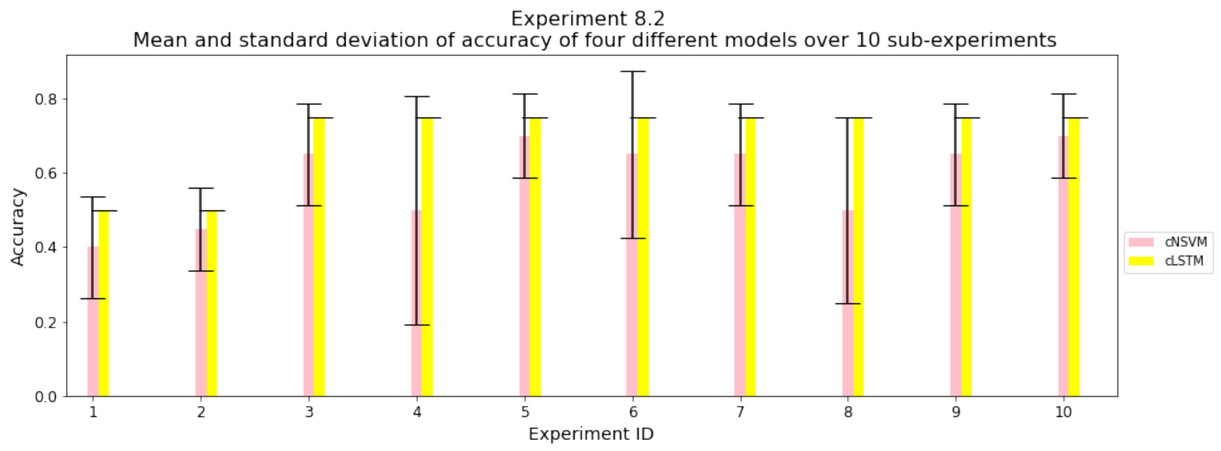


Figure 18: Performance of different models with SV-AR-generated data of two variables and coefficient matrices with a connection from variable 1 to variable 2 from table 16

Experiment ID	$\mathbf{\Phi}$
8.2.1	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$
8.2.2	$\begin{bmatrix} 0.8 & 0.1 \\ 0 & 0.8 \end{bmatrix}$
8.2.3	$\begin{bmatrix} 0.8 & 0.2 \\ 0 & 0.8 \end{bmatrix}$
8.2.4	$\begin{bmatrix} 0.8 & 0.3 \\ 0 & 0.8 \end{bmatrix}$
8.2.5	$\begin{bmatrix} 0.8 & 0.4 \\ 0 & 0.8 \end{bmatrix}$
8.2.6	$\begin{bmatrix} 0.8 & 0.5 \\ 0 & 0.8 \end{bmatrix}$
8.2.7	$\begin{bmatrix} 0.8 & 0.6 \\ 0 & 0.8 \end{bmatrix}$
8.2.8	$\begin{bmatrix} 0.8 & 0.7 \\ 0 & 0.8 \end{bmatrix}$
8.2.9	$\begin{bmatrix} 0.8 & 0.8 \\ 0 & 0.8 \end{bmatrix}$
8.2.10	$\begin{bmatrix} 0.8 & 0.9 \\ 0 & 0.8 \end{bmatrix}$

Table 16: Values of coefficient matrix $\mathbf{\Phi}$ for experiments from figure 18

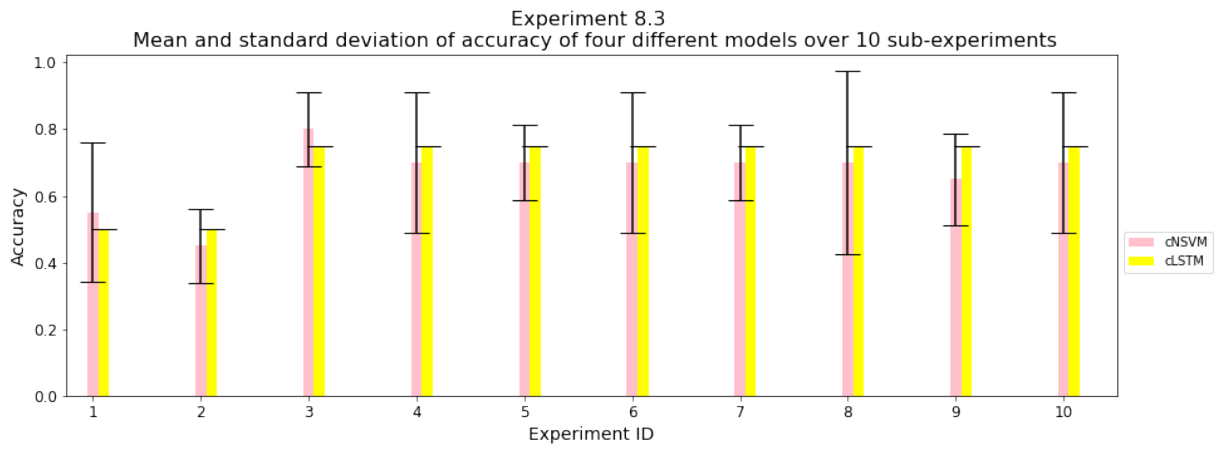


Figure 19: Performance of different models with SV-AR-generated data of two variables and coefficient matrices with a connection from variable 2 to variable 1 from table 17

Experiment ID	$\mathbf{\Phi}$
8.3.1	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$
8.3.2	$\begin{bmatrix} 0.8 & 0 \\ 0.1 & 0.8 \end{bmatrix}$
8.3.3	$\begin{bmatrix} 0.8 & 0 \\ 0.2 & 0.8 \end{bmatrix}$
8.3.4	$\begin{bmatrix} 0.8 & 0 \\ 0.3 & 0.8 \end{bmatrix}$
8.3.5	$\begin{bmatrix} 0.8 & 0 \\ 0.4 & 0.8 \end{bmatrix}$
8.3.6	$\begin{bmatrix} 0.8 & 0 \\ 0.5 & 0.8 \end{bmatrix}$
8.3.7	$\begin{bmatrix} 0.8 & 0 \\ 0.6 & 0.8 \end{bmatrix}$
8.3.8	$\begin{bmatrix} 0.8 & 0 \\ 0.7 & 0.8 \end{bmatrix}$
8.3.9	$\begin{bmatrix} 0.8 & 0 \\ 0.8 & 0.8 \end{bmatrix}$
8.3.10	$\begin{bmatrix} 0.8 & 0 \\ 0.9 & 0.8 \end{bmatrix}$

Table 17: Values of coefficient matrix $\mathbf{\Phi}$ for experiments from figure 19

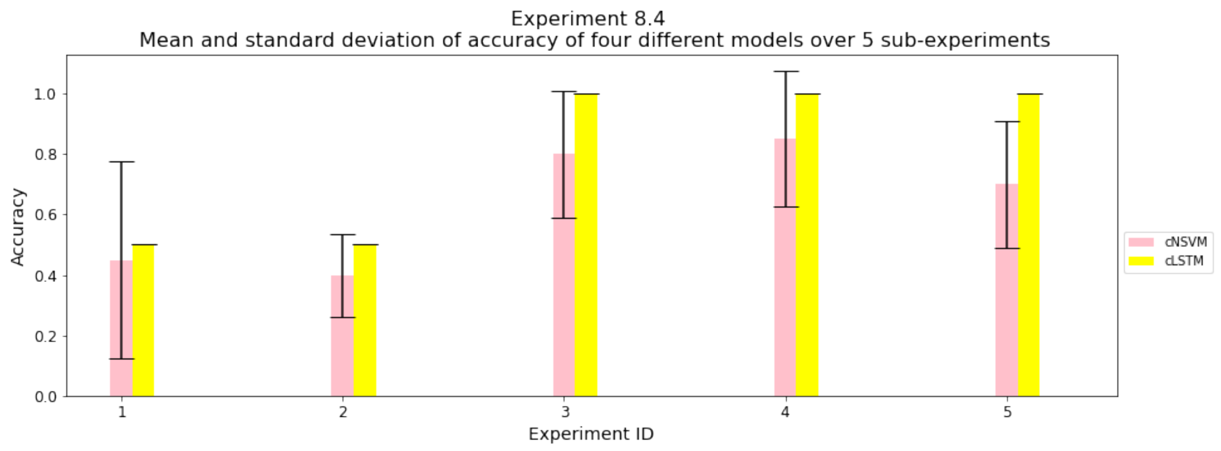


Figure 20: Performance of different models with SV-AR-generated data of two variables and coefficient matrices with a mutual connection between variables from table 18

Experiment ID	$\mathbf{\Phi}$
8.4.1	$\begin{bmatrix} 0.8 & 0.1 \\ 0.1 & 0.8 \end{bmatrix}$
8.4.2	$\begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$
8.4.3	$\begin{bmatrix} 0.8 & 0.3 \\ 0.3 & 0.8 \end{bmatrix}$
8.4.4	$\begin{bmatrix} 0.8 & 0.4 \\ 0.4 & 0.8 \end{bmatrix}$
8.4.5	$\begin{bmatrix} 0.8 & 0.5 \\ 0.5 & 0.8 \end{bmatrix}$

Table 18: Values of coefficient matrix $\mathbf{\Phi}$ for experiments from figure 20

4.2.3 Experiment 9

Goal: The goal of experiment 9 was to test if the models can distinguish between contemporaneous correlation in the correlation terms of the covariance matrix of the volatility's noise generation process $\mathbf{\Sigma}_\eta$, and the connection encoded in the coefficient matrix, $\mathbf{\Phi}$, of SV-AR model. Even though there are two noise sources, we only experimented with the additive noise of the volatility process, $\boldsymbol{\eta}$ (section 2.2.2).

Setup: The setup of the experiment is the same as in experiment 3 (section 4.1.3). There is no connection between the variables encoded on the coefficient matrix, $\mathbf{\Phi}$. The variables only have a strong self-connection (connection from the future timestep to the previous timestep of the same variable), which results in $\mathbf{\Phi}$ taking the following form $\mathbf{\Phi} = \begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$. We first ran the models three times where the variances on the covariance matrix, $\mathbf{\Sigma}_\eta$, have the following values: 0.5, 0.1, and 0.05, and with the covariances being 0.4, 0.09 and 0.04. We kept the covariances as high as possible while still satisfying the positive definite constraint of the covariance matrix of the noise generation process. This corresponds to the subexperiments: 9.1, 9.3, and 9.5, respectively. We then repeated these experiments, removing the covariance of the noise generation. This corresponds to subexperiments: 9.2, 9.4, and 9.6, respectively. We repeated this process 5 times with different seed values of the noise generation process to make a fair comparison. In total, this resulted in 30 runs for each of these two models.

Results: Figure 21 shows the results of this experiment. Initially, we can see that cLSTM does not perform well, with an overall performance of around 50%. The performance of cLSTM in this experiment should not be considered to draw any conclusions as cLSTM cannot recognize that there is no connection. It was also not able to do so even with a smaller noise in experiment 8 (section 4.2.2).

Furthermore, in the case of cNSVM, we can see a slight improvement when the contemporaneous correlation is removed. However, it still achieves a low overall score, and its accuracies have a high standard deviation. But, with smaller noise, cNSVM's performance increases. This shows that the cNSVM's performance is susceptible to the magnitude of the introduced noise of the volatility process.

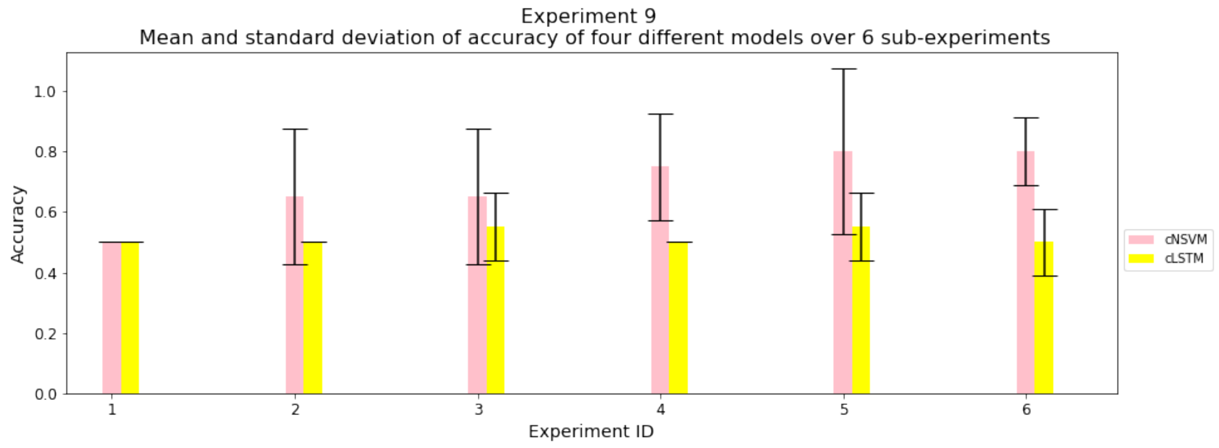


Figure 21: Performance of different models with SV-AR-generated data and different noise covariance matrices from table 19

Experiment ID	$\mathbf{\Phi}$	$\mathbf{\Sigma}$
9.1	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 0.4 \\ 0.4 & 0.5 \end{bmatrix}$
9.2	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$
9.3	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0.09 \\ 0.09 & 0.1 \end{bmatrix}$
9.4	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$
9.5	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.05 & 0.04 \\ 0.04 & 0.05 \end{bmatrix}$
9.6	$\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}$

Table 19: Values of different noise covariance matrices for experiments from figure 21

4.2.4 Experiment 10

Goal: In experiment 10, we wanted to test the performance of these models in the setting with high-dimensional data. Similar to experiment 4 (section 4.1.4), the goal is not only to compare the models' accuracies but also their training time. This is particularly important for the data generated by the SV-AR model, as there already exist methods that can fit the data of the stochastic volatility well, but they rely on sampling approaches (which are too slow).

Setup: The setup of this experiment corresponds to the setup of experiment 4 (section 4.1.4). We used the data of dimensions from 2 to 8 and repeated the experiment 5 times with different seed values of the random number generator. As a result, each of the two models was trained 35 times. Also, we randomly initialized the positions of non-zero elements in the matrix $\mathbf{\Phi}$ while maintaining the same sparsity of the coefficient matrix. This prevented the models from achieving higher accuracy simply because they could estimate connections more easily than non-connections. Regardless of their dimensions, the coefficient matrices had a sparsity of 50%. Additionally, all non-zero values in $\mathbf{\Phi}$ were identical and maximized within stationarity constraints and a 50% fill rate. Lastly, the coefficient matrix was filled with zeros at random positions in each run.

Results: Table 20 shows that, time-wise, cLSTM is more scalable than cNSVM, as it takes less time to run in higher dimensions. We can also observe that, generally, these models take considerably longer time than the models from experiment 4 (section 4.1.4). However, figure 22 shows that cNSVM performs better. Although, as in the previous experiments, cNSVM has a large standard deviation. cLSTM maintains an overall stable performance regardless of the number of dimensions in the data (around 50% mean accuracy), indicating that it is not able to estimate the connections.

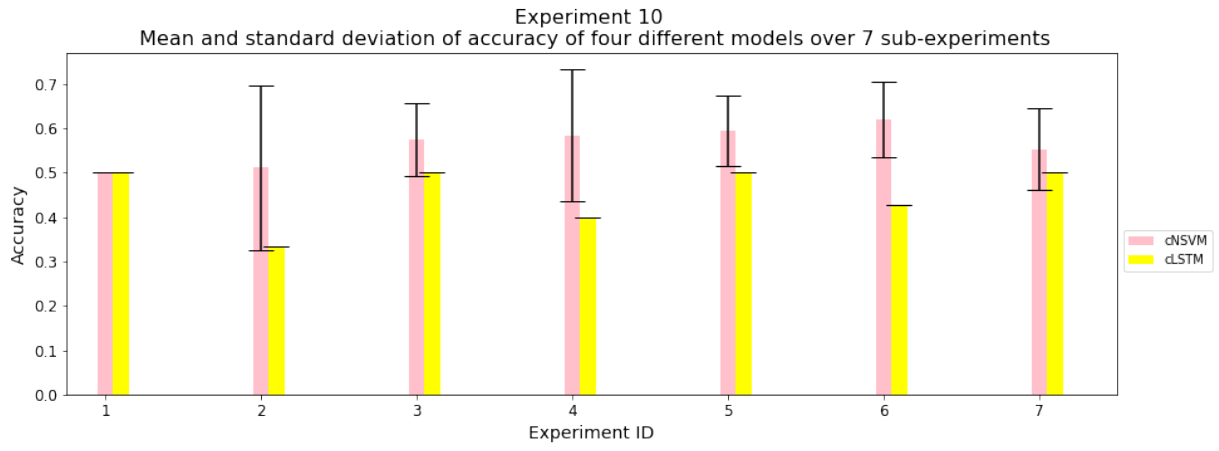


Figure 22: Performance of different models with SV-AR-generated data and different number of variables from table 20

Experiment ID	Number of variables	Average time per run per model (minutes)	
		cLSTM	**cNSVM**
10.1	2	15.909	21.92
10.2	3	21.43	32.06
10.3	4	31.41	42.14
10.4	5	38.3	52.78
10.5	6	44.69	63.53
10.6	7	51.04	74.14
10.7	8	59.45	85.16

Table 20: Values of different numbers of variables and average time duration for experiments from figure 22

4.2.5 Experiment 11

Goal: In the final experiment with SV-AR data, we wanted to test if models are biased towards estimating connections rather than non-connections or vice-versa. Since both cNSVM and cLSTM are NGC-based models, they begin the training process with all connections present, and during the training, they gradually remove unnecessary connections to better fit the data.

Setup: The experimental setup of this experiment follows the one from experiment 5 (section 4.1.5). To that attempt, a dataset with 8 variables and differently sparsed coefficient matrices of the volatility process, $\mathbf{\Phi}$, was generated. This means that the initial subexperiment, 11.1, has a dataset with 8 connections between 8 variables, and the final subexperiment, 11.8, has a dataset with 64 connections between 8 variables. Each subexperiment was re-run 5 times to account for different seed values for random number generators, resulting in a total of 35 runs per model.

Results: Figure 23 shows the results of the experiments. From the figure, we can see that cNSVM performs very poorly and achieves a score of around 50% every time. This indicates that the model's estimation is mostly random. However, the performance of cLSTM is highly dependent on the sparsity of the coefficient matrix. cLSTM estimates poorly sparser connectivities between variables. This further proves the statement made in experiment 8 (section 4.2.2): cLSTM is biased towards estimating connections rather than estimating non-connections.

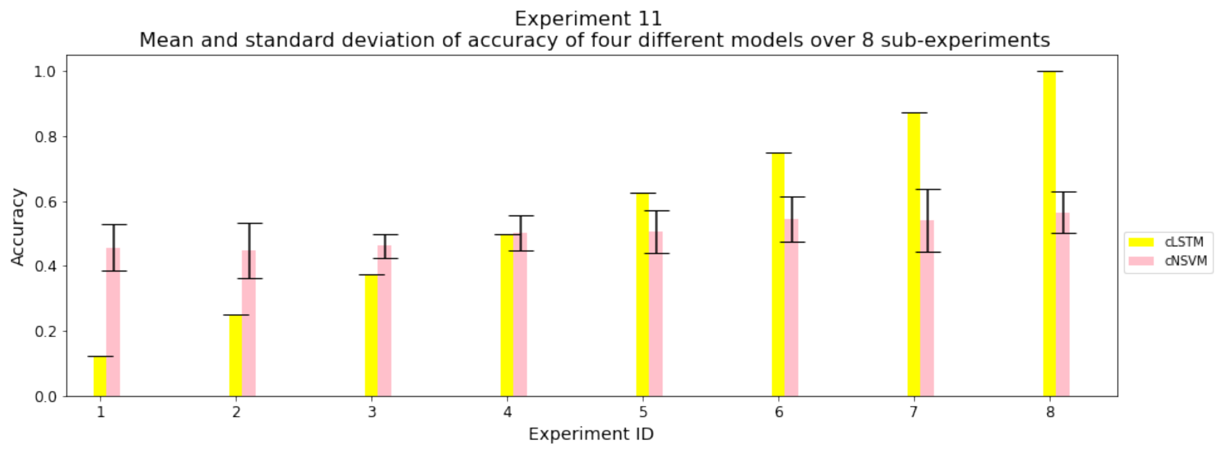


Figure 23: Performance of different models with SV-AR-generated data with 8 variables and differently sparsed coefficient matrices from table 21

Experiment ID	Percentage of connections between variables (%)
11.1	12.5
11.2	25
11.3	37.5
11.4	50
11.5	62.5
11.6	75
11.7	87.5
11.8	100

Table 21: Values of differently sparsed coefficient matrices with 8 variables for experiments from figure 23

4.3 Dataset of Historic Stock Prices

After testing these models on synthetically generated data, all six connectivity estimation models were tested on a real-world dataset of 30 stocks. The data was preprocessed in the same manner as in *Hecq et al.* [37]. These models' results were compared with those reported by the authors of the paper. The paper used three different models, and we compared our results with all three of them. Specifically, we evaluated the performance of our models against the novel PDS-LM HVAR method introduced by the authors (which employs stock selection before performing Granger causality tests). Additionally, since the authors of the paper compared the performance of their model with the other two models, Bi-HVAR and Full-HVAR, we have also compared the performance of these models with the ones we have used in our study. Bi-HVAR tests for causality between each pair of variables separately, making the results not as accurate as it only takes into account two stocks at a time. Full-HVAR estimates high-dimensional VAR models with all stocks simultaneously, which makes it computationally demanding and susceptible to overfitting.

We re-ran the training 5 times with different seed values of the random number generators. To compare the results, we calculated the similarity, defined as the percentage of the same estimations of connections between the models of *Hecq et al.* [37] and the models we used in our study. As in their research, we performed the experiments on both full datasets of prices and on the smaller dataset of prices in 2016-2017.

4.3.1 Experiment 12

Goal and setup: In experiment 12, we compared the performance of our models with the models form *Hecq et al.* [37] on the full dataset of the historical stock prices. This dataset contains 2236 trading days between March 2008 and February 2017.

Results: From figure 24, we can see that (for subexperiment 12.1) t-VAR and NRI are the least similar to the results of the BiHVAR model. The NGC-based models achieve the same performance as the BiHVAR model that the authors used. However, the authors have used the BiHVAR model to show that it does not perform well in a high dimensional setting as it estimates that all stocks are fully connected, which is not correct.
For the PDS-LM HVAR model (subexperiment 12.2) and the FullHVAR model (subexperiment 12.3), t-VAR and NRI achieve the most similar results to the ones from the authors of the paper. Both t-VAR and NRI are similar in 70% of the connections that were also estimated by PDS-LM HVAR and FullHVAR models. The performance of NRI is more unstable than the performance of t-VAR. Suppose the models from *Hecq et al.* [37] reflect the true connections between the stocks. In that case, this could indicate that Student's t-distribution, used in the t-VAR model, can model the unobserved processes of the real-world data.

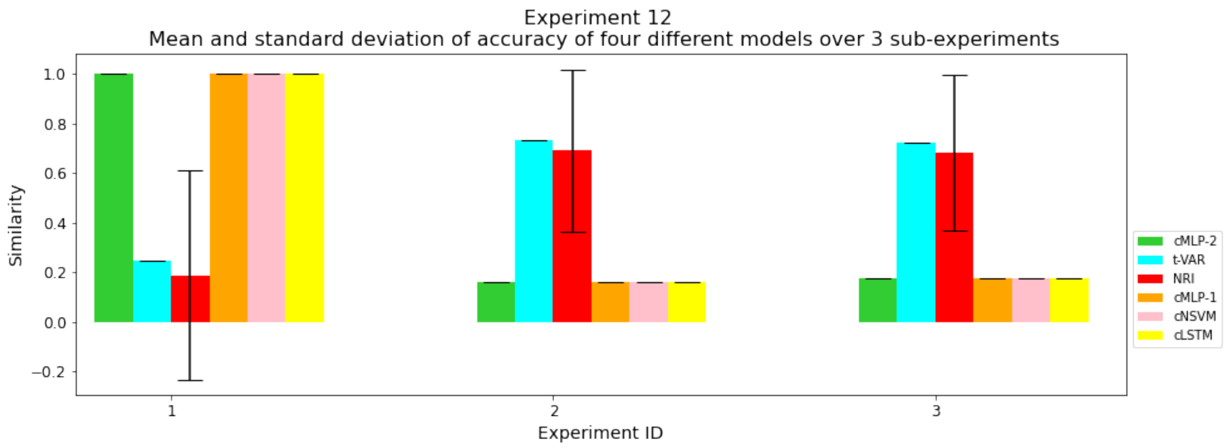


Figure 24: Comparison of models' results with models from *Hecq et al.* [37] (table 22)

Experiment ID	Model
12.1	BiHVAR
12.2	PDS-LM HVAR
12.3	FullHVAR

Table 22: Models from *Hecq et al.* [37] with comparison of results from figure 24

4.3.2 Experiment 13

Goal and setup: In experiment 13, we tested our models on a shorter market period of years 2016-2017 and compared the results with those of *Hecq et al.* [37]. The authors trained their models on a shorter period as the relations are more likely to stay stable over time since this period contains fewer outliers. It avoids two major events: the financial crisis of 2008 and the U.S. debt-ceiling crisis of 2011. This dataset contains 284 trading days.

Results: From figure 25, we can see that, once again, t-VAR and NRI achieve the most similar performance to the models used in *Hecq et al* [37]. They have a similarity score of around 80%, which is measured as the percentage of the connections that are estimated both by our models and the models from the paper. This score is higher than in the previous period in experiment 12 (section 4.3.1). Also, the standard deviation of NRI is now much smaller than in the full-time period in experiment 12 (section 4.3.1). NRI and t-VAR estimate connections the most similar to PDS-LM HVAR, which is the novel model introduced by the authors of the paper. As for the rest of the models, they perform poorly when compared with any of the three methods from *Hecq et al* [37].

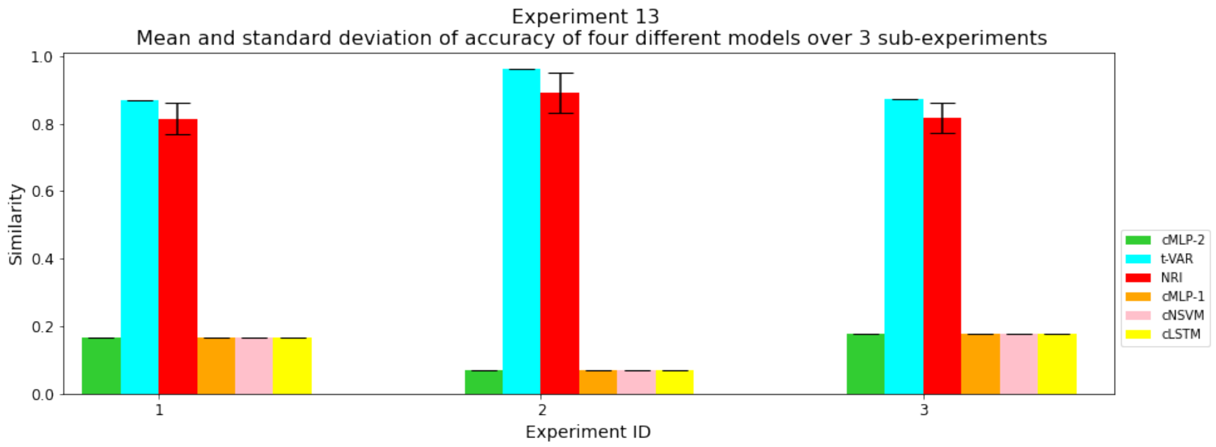


Figure 25: Comparison of models' results with models from *Hecq et al.* [37] (table 23)

Experiment ID	Model
13.1	BiHVAR
13.2	PDS-LM HVAR
13.3	FullHVAR

Table 23: Models from *Hecq et al.* [37] with comparison of results from figure 25

5 Discussion

Our study investigated the performance of various deep learning models for estimating connections on both synthetic and real-world data of historic stock prices (section 2.3). The synthetic data was generated by VAR (section 2.2.1) and SV-AR (section 2.2.2). Connectivity estimation models used in this study were: t-VAR (section 3.1), NRI (section 3.2), cMLP-1 (section 3.3), cMLP-2 (section 3.3), cLSTM (section 3.3), and cNSVM (section 3.4). In this section we have answered the research questions stated at the beginning of this study (section 1).

Research Question 1: How do deep learning models for connectivity estimation perform on data with different strengths of connections, considering both the VAR and SV-AR generated data?
Answer: We found that when the data is generated from standard models with linear connections (such as VAR), simpler models (e.g., t-VAR and cMLP-1) perform marginally better than more complex neural models (e.g., cMLP-2 and NRI). This may be because their formulation matches the formulation of the VAR model used to generate the data. Despite this, all models performed well on the VAR-generated synthetic data. Since cMLP-2 performs better than cMLP-1, we can conclude that adding more layers to the network makes it perform worse on the simple VAR-generated data. For more complex SV-AR-generated data, none of these models (cMLP-1, cMLP-2, t-VAR, NRI) were successful in estimating connections of SV-AR-generated data. To address this, we tested a cLSTM model and introduced a new architecture called cNSVM, which combined the NGC model and the Neural stochastic volatility model. Although cNSVM's performance was not ideal, it showed more potential than cLSTM, with an accuracy of over 50%.

Research Question 2: How robust are connectivity estimation models to noise? Additionally, can they effectively distinguish between contemporaneous correlation in the correlation terms of the covariance matrix of the noise generation process and the connections encoded in the coefficient matrices of both VAR and SV-AR models?
Answer: For data produced by VAR model, NRI and t-VAR showed robustness to noise as they were not affected by increased noise magnitudes (unlike cMLP-1 and cMLP-2). When the data contained contemporaneous correlation, the performances of cMLP-1, cMLP-2, and NRI were slightly affected, while t-VAR remained unaffected. This is possibly due to t-VAR simultaneously estimating both the coefficient matrix of the VAR model and the noise distribution of the data. Regarding the SV-AR-generated data, cNSVM was influenced by the noise magnitude in SV-AR-generated data, and it performed better when the noise levels were lower. It was also affected by the contemporaneous correlation in the noise generation process. cLSTM was not able to perform well on larger noise magnitudes as it achieves an accuracy of 50%.

Research Question 3: How does data dimensionality affect the performance and training time of connectivity estimation models?
Answer: For the case of VAR-generated data, t-VAR, cMLP-1, cMLP-2, and NRI performed well when applied to higher-dimensional VAR-generated data. However, t-VAR and NRI were found to be the least scalable in terms of computational time, as it was heavily impacted by increased data dimensionality. In the case of t-VAR, this is because it uses the EM algorithm, which does not scale well, and in the case of NRI, it is because every time a new variable is added, the number of computations quadratically increases. With respect to the SV-AR-produced data, cNSVM maintained its performance with high-dimensional data, but its computation time was affected. The performance of cLSTM did not change with high-dimensional data. Additionally, it proved to be more computationally efficient than cNSVM.

Research Question 4: Are the connectivity estimation models biased towards estimating connections or non-connections?
Answer: When the data is generated by the VAR model, NRI showed a bias toward estimating connections (rather than non-connections) and performed better when the data had more connections. Other models (t-VAR, cMLP-1, cMLP-2) did not show a bias toward estimating connections or non-connections. As for the data created using the SV-AR model, cLSTM exhibited a strong bias toward estimating the existence of connections, often predicting that all of the variables were interconnected. On the other hand, cNSVM did not show any bias toward estimating connections or non-connections.

Research Question 5: How well do connectivity estimation models perform on real-world historical stock price datasets compared to existing models in the literature?

Answer: t-VAR and NRI emerged as the best-performing models on the real-world data (as they estimated connections that were the most similar to the PDS-LM HVAR model introduced in *Hecq et al* [37]). All other models (cMLP-1, cMLP-2, cLSTM, cNSVM) overestimated the number of connections between stocks. This result indicates that using deep learning does not guarantee better connection estimation, as seen from the strong performance of t-VAR. Furthermore, the success of NRI suggests that using graphs to represent the financial market could be a promising approach.

In summary, our findings highlight the strengths and weaknesses of each connectivity estimation model when applied to both synthetic and real-world financial data, providing valuable insights for future research on deep learning applications in financial time series.

6 Conclusion

In this study, we explored the application of various deep learning models for connectivity estimation in financial time series, focusing on real-world stock market data and synthetic data generated from vector autoregressive (VAR, section 2.2.1) and stochastic volatility (SV-AR, section 2.2.2) models. By employing the t-vector autoregressive model (t-VAR, section 3.1), neural relational inference model (NRI, section 3.2), and the variations of neural Granger causality models (cMLP-1, cMLP-2, cLSTM, section 3.3), as well as introducing a novel model based on the existing NGC model formulation (cNSVM, section 3.4), we aimed to address the challenges of estimating connectivity in financial time series.

The results of our study contribute to the existing body of literature on connectivity estimation in financial time series by showcasing the potential of deep learning models. By comparing the performance of these models in various settings, our research highlights the advantages and disadvantages of using deep learning approaches. Furthermore, our findings on the scalability, contemporaneous correlation in the noise generation, and the biases of predicting connections provide valuable insights for future research and the development of new connectivity estimation models.

Building on our findings, there are several potential directions for future research and improvements in connectivity estimation. For instance, refining the formulation of cNSVM, such as using multiple layers and group Lasso to combine the input variables before passing them to NSVM architecture [20]. Additionally, the performance of connectivity estimation could be examined on alternative data generation models (such as Heston [53], GARCH [54], or SABR model [55]) as well as alternative real-world datasets (such as currency exchange or options data) may help uncover new insights into the effectiveness of these deep learning models in various financial contexts. Finally, investigating novel deep learning architectures and techniques that simultaneously model the connections and the noise generation process could lead to more accurate and reliable connectivity estimation models in cases with larger noise levels and contemporaneous correlation in the noise distribution.

In conclusion, this study contributes to the research on connectivity estimation in financial time series by showcasing the potential of deep learning models for this application. Our findings have important implications for understanding relationships in financial markets and the practical applications of connectivity estimation. With the results from these experiments, we hope to advance the field of financial time series analysis and support investors, traders, and portfolio managers in making more informed decisions based on accurate estimation of connections between financial assets.

References

[1] Rosario N Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B - Condensed Matter and Complex Systems*, 11(1):193–197, 1999.

[2] Mark Rubinstein. Markowitz's" portfolio selection": A fifty-year retrospective. *The Journal of Finance*, 57(3):1041–1045, 2002.

[3] Monica Billio, Massimiliano Caporin, Roberto Panzica, and Loriana Pelizzon. The impact of network connectivity on factor exposures, asset pricing and portfolio diversification. 2016.

[4] Zukarnain Zakaria and Sofian Shamsuddin. Empirical evidence on the relationship between stock market volatility and macroeconomics volatility in Malaysia. *Journal of Business Studies Quarterly*, 4(2):61, 2012.

[5] Alireza Heidarzadeh Hanzaee. Test of the generalizability of Altman's bankruptcy predication model. In *2010 International Conference on Financial Theory and Engineering*, pages 215–250. IEEE, 2010.

[6] Ibrahim Onur Oz and Tezer Yelkenci. The generalizability of financial distress prediction models: Evidence from Turkey. *Accounting and Management Information Systems*, 14(4):685, 2015.

[7] Carmen Broto and Esther Ruiz. Estimation methods for stochastic volatility models: a survey. *Journal of Economic surveys*, 18(5):613–649, 2004.

[8] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[9] Christophe Andrieu, Nando De Freitas, and Arnaud Doucet. An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, 2003.

[10] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[11] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC Press, 2013.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[13] Stephen J Taylor. *Modelling financial time series*. world scientific, 2008.

[14] G Geoffrey Booth, Teppo Martikainen, and Yiuman Tse. Price and volatility spillovers in Scandinavian stock markets. *Journal of Banking & Finance*, 21(6):811–823, 1997.

[15] Vasiliki D Skintzi and Apostolos N Refenes. Volatility spillovers and dynamic correlation in European bond markets. *Journal of International Financial Markets, Institutions and Money*, 16(1):23–40, 2006.

[16] Yuliya Shapovalova. "exact" and approximate methods for Bayesian inference: Stochastic volatility case study. *Entropy*, 23(4):466, 2021.

[17] Yuliya Shapovalova and Michael Eichler. Measuring and quantifying uncertainty in volatility spillovers: A bayesian approach. *Data Science in Science*, 2(1):2176379, 2023.

[18] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International conference on machine learning*, pages 2688–2697. PMLR, 2018.

[19] Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily B Fox. Neural Granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4267–4279, 2021.

[20] Rui Luo, Weinan Zhang, Xiaojun Xu, and Jun Wang. A neural stochastic volatility model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[21] Julien Chevallier and Florian Ielpo. Volatility spillovers in commodity markets. *Applied Economics Letters*, 20(13):1211–1227, 2013.

[22] Jianqing Fan and Jinchi Lv. High dimensional covariance matrix estimation using a factor model. *Journal of Econometrics*, 147(1):186–197, 2006.

[23] Noureddine El Karoui. High-dimensionality effects in Markowitz problem and other quadratic programs with linear constraints: Risk underestimation. 2010.

[24] Federico M Bandi and Jeffrey R Russell. Microstructure noise, realized variance, and optimal sampling. *The Review of Economic Studies*, 75(2):339–369, 2008.

[25] Adrian R Pagan and G William Schwert. Alternative models for conditional stock volatility. *Journal of econometrics*, 45(1-2):267–290, 1990.

[26] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

[27] Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.

[28] Christopher A Sims. Comparison of interwar and postwar business cycles: Monetarism reconsidered, 1980.

[29] Selcuk Bayraci, Yakup Ari, and Yavuz Yildirim. A vector auto-regressive (VAR) Model for the Turkish financial markets. 2011.

[30] Ana Cuvak and Zilvinas Kalinauskas. Application of vector autoregression model for Lithuanian inflation. *Economics and Management*, (14):145–150, 2009.

[31] Graham DI Barr and BS Kantor. The application of a vector autoregressive model to money, income and price links in the South African economy. *Studies in Economics and Econometrics*, 14(1):39–49, 1990.

[32] Eric Zivot and Jiahui Wang. *The Stationary Vector Autoregression Model*, page 386–390. Springer, 2006.

[33] Rama Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative finance*, 1(2):223, 2001.

[34] BB Mandelbrot. The variation of certain speculative prices (pp. 392–417). *XXXVI: Journal of Business*, 1963.

[35] Robert Engle and Andrew Patton. What good is a volatility model? *Quantitative Finance*, 1, 02 2001.

[36] Eric Ghysels, Andrew Harvey, and Eric Renault. Stochastic volatility. *Statistical Methods in Finance*, 14:17–19, 12 1995.

[37] Alain Hecq, Luca Margaritella, and Stephan Smeekes. Granger causality testing in high-dimensional vars: a post-double-selection procedure. *arXiv preprint arXiv:1902.10991*, 2019.

[38] Kenneth L Lange, Roderick JA Little, and Jeremy MG Taylor. Robust statistical modeling using the t distribution. *Journal of the American Statistical Association*, 84(408):881–896, 1989.

[39] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[40] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[41] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[42] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[43] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[44] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[46] Abien Fred Agarap. Deep learning using rectified linear units (ReLU). *arXiv preprint arXiv:1803.08375*, 2018.

[47] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[48] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[49] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.

[50] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[51] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[52] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[53] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies*, 6(2):327–343, 1993.

[54] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

[55] Patrick S Hagan, Deep Kumar, Andrew S Lesniewski, and Diana E Woodward. Managing smile risk. *The Best of Wilmott*, 1:249–296, 2002.