Forecasting the Value-at-Risk of an Equity Portfolio: A Recurrent Mixture Density Network Approach

Hubert Normandin-Taillon

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

December 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Hubert Normandin-Taillon**

Entitled: **Forecasting the Value-at-Risk of an Equity Portfolio: A Recurrent Mixture Density Network Approach**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Shin Hwei Tan

_____ Examiner
Dr. Abdelhak Bentaleb

_____ Supervisor
Dr. Chun Wang

_____ Co-supervisor
Dr. Frédéric Godin

Approved by _____
Joey Paquet, Chair
Department of Computer Science and Software Engineering

_____ 2023

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Forecasting the Value-at-Risk of an Equity Portfolio: A Recurrent Mixture Density Network Approach

Hubert Normandin-Taillon

The value-at-risk is a useful metric employed by financial institutions to measure the risk of a portfolio. However, accurately forecasting the value-at-risk is difficult, as it requires predicting the returns of the portfolio's assets. Forecasting asset returns is particularly challenging due to their stochastic nature and the presence of 'stylized facts' such as heteroskedasticity, fat tails, and skewness in stock returns series. This thesis considers modeling the assets returns using a recurrent mixture density network, which has been previously proposed to model other financial time-series. In this thesis, we propose an improved recurrent mixture density network architecture, as well as a pretraining method to improve the numerical stability and convergence speed of the model. We also propose the Copula-S-RMDN-GARCH, which extends the current recurrent mixture density network architecture to multivariate settings. We compare the value-at-risk forecast obtained with the Copula-S-RMDN-GARCH with the forecasts obtained from a Copula-AR-GARCH.

Acknowledgments

I would first like to express my gratitude to my advisor Dr. Chun Wang, who gave me great advises and guidance throughout my research. His advises help me become a better researcher, but more importantly his ideas guided me throughout my research project. His teaching and advice allowed me to greatly improve my skills in communication and academic writing. I would also like to express my gratitude to my thesis co-advisor and Mitacs internship supervisor Frédéric Godin. His support and advises were fundamental in conducting my research. I appreciated the guidance and the opportunity to conduct research under your supervision and gain experience through the internship.

I also acknowledged Quantolio and Amine El-Kaouachi for the great learning opportunities during my Mitacs Accelerate internship at Quantolio. I would also like to thank Abdoul Haki Maoude, Waleed Ahmed, and Maryam Ouhadi for their support and help during my Mitac Accelerate internship at Quantolio and beyond. I acknowledged the funding source provided by Mitacs under grant IT23943.

I also acknowledged my father, family, and friends who supported me throughout my whole degree. Their support help me go through difficult time during my degree.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As part of their operations, institutional investors hold large portfolios on behalf of their clients. While financial institutions hold a variety of different types of assets, this thesis focuses on equity. This, in turn, requires the institutions to effectively manage risk. Risk management requires quantifying the risk of a portfolio accurately. This process, part of the portfolio management process, is particularly important, as it will influence all operations of a financial institution. Quantifying the risk of the portfolio, in turn, requires the choice of a risk metric. It is also of the utmost importance that the risk metrics used to quantify the risk be interpretable by the management team. The most commonly used metric for this is the value-at-risk, which can be easily interpreted. While multiple criticisms of the value-at-risk have been formulated, it is still the most widely adopted risk measure. It will, therefore, be the focus of this thesis.

The value-at-risk can be analyzed using three categories of methods: the historical simulation method, the variance-covariance method, and the Monte-Carlo method (Dalbudak, Atan, & Yilmaz, 2017). The historical simulation method analyzes the value-at-risk of a portfolio based on historical data. This method has the advantage of being simple to use and examines the effect of historical data on a specified portfolio. However, it does not incorporate information about the future risk of a portfolio as it only includes information about the past (Dalbudak et al., 2017). The second method employed to analyze the value-at-risk is the variance-covariance method. This method relies on the

variance and covariance matrix to estimate the value-at-risk of a portfolio analytically. It is advantageous as it is computationally efficient, but it is a parametric method that relies on the parameters of a distribution. As the name suggests, the variance-covariance method is not designed to accommodate skewness and kurtosis as it relies on the Gaussian distribution (Dalbudak et al., 2017). Lastly, the Monte-Carlo method is the most flexible method, as it allows modeling the values of assets using different models. The Monte-Carlo method requires sampling multiple series of new observations from which the value-at-risk can be estimated (Dalbudak et al., 2017). This method is advantageous because it can be leveraged to forecast the risk of a portfolio rather than only analyzing it. This is accomplished by forecasting the value of the assets rather than simply modeling their distribution. As we focus on forecasting the value-at-risk, this method will be the focus of this thesis.

In order to forecast the value at risk using the Monte-Carlo method, the value of a portfolio's assets needs to be forecasted. However, forecasting the value of financial assets is complex and requires specialized models, as stock returns have been shown to incorporate highly stylized facts such as heteroskedasticity, skewness, and fat tails. Models that incorporate these characteristics have been widely adopted in the industry. These models include the ARCH/GARCH families of models, the Markov switching model, as well as neural network models. The ARCH/GARCH models were first introduced to model heteroskedasticity. This is achieved by forecasting the conditional variance of a time series as an autoregressive process in the case of the ARCH model and as an autoregressive moving average process in the case of the GARCH model. A different approach is the Markov switching models, which model the economy using multiple states. This allows the Markov switching model to differentiate between periods of growth and periods of loss in the market. Numerous neural network models have been proposed in the literature that can be used to forecast the value-at-risk using different approaches. These approaches include forecasting the price, returns, volatility, or distribution of assets. To this day, the ARCH/GARCH models remain the most widely adopted by financial institutions.

While the ARCH/GARCH models have been widely adopted, they have two major limitations:

(1) The ARCH/GARCH models are parametric models. As such, they make assumptions regarding the underlying time series distribution. This can be problematic in the case of financial time series, as the distribution of financial asset returns is complex and uncertain.

(2) The ARCH/GARCH models are also linear models. They model the conditional variance as a linear combination of the previous conditional variance. This can be problematic, as evidence suggests that stock returns follow a non-linear dynamics (Hinich & Patterson, 1985).

These limitations can be obstacles to obtaining an accurate forecast of the value-at-risk. In its basic form, the ARCH/GARCH models are univariate. This is problematic as the values of assets in a portfolio are co-dependent. Multiple multivariate GARCH models (MGARCH) have been proposed. The multivariate GARCH models include the Constant Conditional Correlation model (CCC) and the Dynamic conditional correlation model (DCC). The CCC model assumes that the conditional correlation is constant through time, while the DCC model dynamically assumes that the conditional correlation is time-varying. The specific dynamic conditional correlation under the DCC model is commonly thought to be a linear MGARCH model. However, it is not as tractable and is inherently guaranteed to be linear due to some peculiarity in its specification (Aielli, 2011). The DCC also requires the estimation of a large number of parameters, making it more difficult to estimate the model in high dimensions. An alternative to the CCC and DCC model is the OGARCH model, which uses dimensionality reduction of the multivariate time series to minimize the number of estimated parameters. However, this method is limited and has been shown to underperform other methods used for extending the GARCH model to multivariate settings (Engle, 2002). Another alternative to the model presented is the copula-GARCH model. This model has the advantage of being flexible, as different copulas can be used for modeling the dependence between assets. However, the multivariate GARCH still relies on the univariate ARCH/GARCH models and is subject to the two major limitations of the ARCH/GARCH models.

Another type of model used to forecast financial time series is neural networks. Multiple specific neural network architectures have been developed to forecast a time series, with the most common being recurrent neural networks such as the fully recurrent neural network, LSTM, and GRU.

Recurrent neural networks have the ability to maintain an internal state, allowing them to consider information about the past. The LSTM and GRU architectures were developed to incorporate longer memory than the fully recurrent neural network. These models have been successful in forecasting financial time series (Di-Giorgi et al., 2023; Fjellström, 2022; Jia & Yang, 2021; Kamalov, Smail, & Gurrib, 2020; Paranhos, 2021). However, the recurrent neural network only forecasts the expected value of the time series. The information provided by the expected value does not allow us to forecast the risk of a financial asset, as the other characteristics of the asset's return distribution are as important as its expected value. An alternative to recurrent neural networks is the recurrent mixture density network, which forecasts the shape of the conditional distribution of a time series. The recurrent mixture density network uses a recurrent neural network to forecast the parameters of a mixture distribution. This allows the model to forecast not only the expected value of the time series but all the information contained in the distribution. This model does not have the limitations of the ARCH/GARCH model. First, the recurrent mixture density network is a semi-parametric method, and hence it does not assume a specific shape of the distribution. The RMDN is limited only by the number of components in the mixture and by the shape of the mixture's underlying distribution. Furthermore, the RMDN can model various distribution shapes that can be represented by a mixture Gaussian distribution, including some forms of skewed distributions. Second, unlike the GARCH models, the RMDN is a neural network, which means that the RMDN can model the non-linear dynamics of the conditional variance and the mean. It can also incorporate heteroskedasticity inherent in stocks returns, as well as time-varying skewness and kurtosis. This allows mimicking the stylized facts inherent in financial time series, such as heteroskedasticity, skewness, and fat tails.

In this thesis, we propose an improved RMDN architecture based on the work of Nikolaev, Tino, and Smirnov (2012) and Schittenkopf, Dorffner, and Dockner (2000). This new architecture enhances the numerical stability of the model. Additionally, we introduce a pretraining method to improve the convergence of the model. Our proposed improvement focuses on the univariate RMDN architecture to forecast the value of each individual asset in the portfolio. Furthermore, we present a method for extending the model to a multivariate setting using a Gaussian copula. Our method enables the scaling of the forecast to high dimensions. We also introduce a method for estimating the value-at-risk using an importance sampling scheme to efficiently sample from the

RMDN.

1.1 Outline of the Thesis

In this thesis, we first present background information on the ARCH/GARCH models, the Markov switching model, and the estimation of value-at-risk. Additionally, we provide an introduction to mixture density networks and their applications. In the following chapter, we present a literature review focused on financial time series forecasting methods used for predicting the value-at-risk, as the value-at-risk can be easily estimated from the forecast. The next chapter outlines the methodology for forecasting stock returns and estimating the value-at-risk. We then analyze the results obtained from our model and compare them with estimates from the GARCH forecasts. Finally, we conclude and present potential future directions in Chapter 6.

Chapter 2

Background

In this chapter, we first introduce key financial concept important for understanding the problematic of this thesis. Namely, the concept of value-at-risk and its estimation. We then proceed to introduce two key financial time series model: the ARCH/GARCH models. The next section introduces the reader to mixture density network and their application. We also introduce the recurrent mixture density network model that will be used throughout this thesis. Finally, we introduce the reader to the concept of copula and to the theoretical foundation surrounding copula.

2.1 Value-at-Risk

The value-at-risk has been central to many financial institutions' risk management operations. The value-at-risk has been first developed at J.P. Morgan in the 1980's. It has since been adopted by many financial institutions. Since its introduction, the VaR has been criticized. While proponents of its use see it as a way to report risk in units that everybody understands, detractors consider it to be flawed as it cannot accurately measure all possible risks associated with a financial instrument (Taleb & Jorion, 1997). While the VaR has both its proponents and detractors, it provides a simple yet effective view of a portfolio's risk. For the purpose of this thesis we define the value-at-risk as the smallest amount of capital that keeps the probability of a negative outcome below the level α

(Föllmer & Schied, 2016). We will use the following mathematical definition of the value-at-risk

$$VaR_\alpha(X) = inf\{m|P[X + m < 0] \leq \alpha\} \tag{1}$$

where α is the maximum level of risk the investor is willing to take. There are three main methods that can be used to estimate the value-at-risk of a portfolio. Namely, the historical simulation method, the analytical method, and Monte Carlo simulation methods. Each of those methods has its advantages and disadvantages. The following section describes each of the main methods and presents their advantages and disadvantages.

In the historical simulation method, the value-at-risk is estimated by applying the weight of the portfolio over the historical returns and estimating the quantile of loss over the resulting portfolio returns (Adamkoa, Spuchľákováb, & Valáškováb, 2015). This method has the advantage of being simple and requiring little computation. It also does not make any distributional assumptions regarding the underlying time time series. While it is easy to use, the historical simulation method require a long horizon to be accurate and to capture the underlying time series characteristics such as fat tails and heteroskedasticity. However, a long horizon might contain events that are irrelevant of the current market conditions. An improvement over the historical simulation method is the bootstrap method, which allows to increase the sample size and retain the time series characteristics (Adamkoa et al., 2015).

The value-at-risk can also be estimated analytically. This is usually done by the variance-covariance method (Adamkoa et al., 2015). The main advantage of this method is that it does not require data and it is computationally efficient. However, this method makes assumptions regarding the distribution of the assets returns and requires estimating the correlation between every assets. The estimation of the correlation matrix can be particularly challenging due to the large number of assets present in a portfolio and to the presence of spurious relationships between assets. This led to multiple methods being developed to estimate the covariance matrix. Multiple methods have been proposed to estimate the covariance matrix including factor models (Fan, Fan, & Lv, 2008),

shrinkage estimators (Ledoit & Wolf, 2003), and PCA based methods (Avellaneda & Serur, 2020).

The last method used to estimate VaR is the Monte Carlo method. This method is the most flexible as it allows for the inclusion of financial instrument with a non-linear payoff. This is the reason why it is the most commonly used by financial institutions today. However, this method is computationally intensive and requires scenarios based on assumptions about the financial assets. Those assumptions can result in biased estimation of the value-at-risk (Adamkoa et al., 2015). Two of the most commonly used models for generating scenarios is the ARCH/GARCH family of models and the regime switching models.

2.2 ARCH/GARCH Models

Heteroskedasticity is a recurring problem encountered when analysing time series. Financial time series are particularly prone to this problem (Schwert & Seguin, 1989). Most time series forecasting models such as AR and ARIMA do not take into account that the conditional variance of the model depends on past observations. The ARCH/GARCH models have been proposed to model the conditional variance. The ARCH model solves this problem by modeling the conditional variance as an AR process (Engle, 1982). The GARCH model generalizes this by extending the AR model to an ARMA model. The following two sections introduce the two models.

2.2.1 ARCH

To understand the ARCH model, we first need to consider a first-order autoregressive model

$$y_t = \gamma y_{t-1} + \epsilon_t \tag{2}$$

where ϵ is white noise with variance σ^2 and γy_{t-1} is the conditional mean of y_t. The conditional variance of y_t is therefore σ^2 and the unconditional variance of y_t is $\frac{\sigma^2}{1-\gamma^2}$.

Given the set of all information available at time t ψ_t, we can define an ARCH process as

$$\begin{aligned} y_t|\psi_{t-1} &\sim \mathcal{N}(0, h_t) \\ h_t &= \alpha_0 + \alpha_1 y_{t-1}^2 \end{aligned} \tag{3}$$

the conditional variance dynamic can be generalized by defining the conditional variance h_t as the function h such that

$$h_t = h(y_{t-1}, y_{t-2}, ..., y_{t-p}, \alpha) \tag{4}$$

This generalization allows for the specification of the GARCH model and further extension of the model.

2.2.2 GARCH

The GARCH model first appeared in the article "Generalized Autoregressive Conditional Heteroskedasticity" (Bollerslev, 1986). The GARCH model generalized the ARCH model published in Engle (1982). The GARCH(p, q) model reduces to a ARCH(q) model when p = 0. The GARCH(p, q) model's main advantage over the ARCH(q) model is its introduction of a moving average component in the specification of the conditional variance. This make the GARCH model more flexible and introduces longer memory. In turn, this allows the GARCH model to account for persistent volatility and volatility clustering. A GARCH model of the log returns of the price y_t can be defined using the following equations

$$\begin{aligned} y_t|\psi_{t-1} &\sim \mathcal{N}(0, h_t) \\ h_t &= \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \end{aligned} \tag{5}$$

We can observe that the difference between the GARCH(p, q) model and the ARCH(q) model is the addition of the moving average in the second part of the equation. This addition is akin to the change made from an AR model to an ARMA model on the conditional variance.

2.3 Mixture Density Network

The mixture density network neural network architecture was first introduced by Bishop (1994), to model conditional probabilities of the target data. This contrasts the usual approach for training a neural network, which models the conditional average of the target data. This is accomplished by modelling the shape of the conditional probability distribution directly. In a mixture density network, the shape of the conditional probability distribution is approximated using a mixture distribution of N components. While the mixture can be composed of any distribution, the most commonly used is a mixture of Gaussian distributions. The parameters of each component in the mixture are estimated using a neural network, which is trained by minimizing the negative log-likelihood.

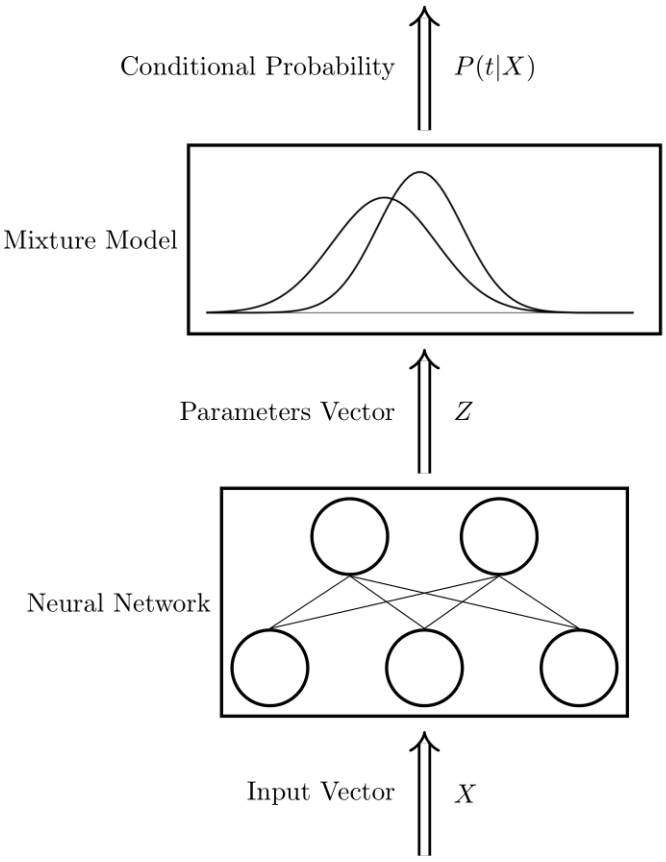

Figure 2.1: General Architecture of the Mixture Density Network

2.3.1 RMDN

Schittenkopf et al. (2000) present a recurrent mixture density network model for forecasting the distribution of assets returns. A similar model was proposed by Nikolaev et al. (2012), who called it an "RMDN-GARCH", as it models the heteroskedasticity of the data in a similar fashion as a GARCH model. The RMDN-GARCH also reduces to a GARCH model under specific circumstances. While the RMDN has some similarity to the GARCH model, it has two clear advantages, which make it well suited to model financial time series. First, the RMDN models the volatility non-linearly. Second, as the resulting time-conditional density forecast is a mixture, the RMDN-GARCH model is able to model the skewness and the kurtosis in a time-varying fashion (Schittenkopf et al., 2000).

The time-conditional density of the distribution is defined as

$$p(r_t|R_{t-1}) = \sum_{i=1}^{N} \eta_{i,t}\phi_i(\mu_{i,t}, \sigma_{i,t}^2) \tag{6}$$

where we define the weight of each distribution in the mixture as $\eta_{i,t} \equiv \eta_{i,t}(r_{t-1})$. The parameter $\mu_{i,t}$ is the mean of the i^{th} mixing distribution and the parameter $\sigma_{i,t}^2$ is the variance of for the i^{th} mixand distribution. The conditional density function of the i^{th} mixing distribution is defined as ϕ_i. R_{t-1} is the information available at time $t-1$. The authors in Nikolaev et al. (2012) consider the Gaussian and the Student-T distributions as candidates for the mixing distributions. If the conditional density follows a Student-T distribution, the degree of freedom of the distribution ν also needs to be estimated. Unlike the other parameters of the distribution, the degree of freedom parameter ν is estimated outside the RMDN by MCMC (Nikolaev et al., 2012). The instantaneous log-likelihood of the time conditional density is defined as

$$L_t = \log p(r_t|R_{t-1}) = \log \sum_{i=1}^{N} \eta_{i,t}\phi_i(\mu_{i,t}, \sigma_{i,t}) \tag{7}$$

The parameters of the neural network can be estimated by minimizing the negative log-likelihood. The authors in Nikolaev et al. (2012) choose the BFGS algorithm due to its fast convergence and stability. In the next section, we describe the neural networks used to estimate the parameters of the

RMDN.

2.3.2 Neural Network Architecture

In order to estimate the three parameters of the RMDN-GARCH, three interlinked neural networks are used; each estimating different parameters. The authors in Nikolaev et al. (2012) called those three networks the mixing network, the mean-level network and the variance recurrent network. The mixing network is used to estimate the mixing coefficient vector η_t, the mean-level network is used to estimate the mean vector μ_t, and the variance recurrent network is used to estimate the conditional variance vector σ_t. The authors in Nikolaev et al. (2012) specify a feedforward architecture for the mixing network and the mean-level network. The variance recurrent network is specified as a recurrent neural network. This parallels the GARCH model, as it models time dependencies. The three different neural networks make the bulk of the architecture of the RMDN-GARCH. However, to properly estimate each parameter of the RMDN-GARCH model, the links between the different neural network are as important as the architecture of each individual neural network as such links allows to model heteroskedasticity.

The input layer of the mixing network and the mean-level network takes for input the log-returns of the assets. The input layer of each network is composed of 3 nodes followed by an activation layer containing one linear activation function and two hyperbolic tangent activation functions. We will refer, throughout this thesis, to an activation layer with two or more different activation functions as an heterogeneous activation layer. This type of activation layer allows the neural network to reduced to a GARCH model under certain circumstances. This will be shown in Section 4.3. The variance recurrent network takes for input the error e^2_{t-l} and the previous estimate for the conditional variance h_{t-1}. The error e^2_{t-l} is defined as $e^2_{t-l} = (y_{t-l} - \mu^2_{t-l})$. The fitting error is computed using the mean of the mixture distribution and the log returns of the assets r_{t-l}. It is also followed by and heterogeneous activation layer. All three networks are followed by an output layer, which is designed to ensure that the property of each parameters are respected.

For the mixing network, a softmax activation function is used to estimate the mixing coefficient.

This ensures that the sum of the mixing coefficient is equal to 1. The mean-level network output activation layer is a linear activation function, This ensured that the mean of each component is unrestricted. To properly estimate the variance σ_t^2 parameters, we need to define activation functions that will ensure that the conditional variance is always superior to 0. The authors in Nikolaev et al. (2012) specified an absolute function which ensures that the output of variance recurrence network is always positive. The following diagram describes the architecture of the neural network

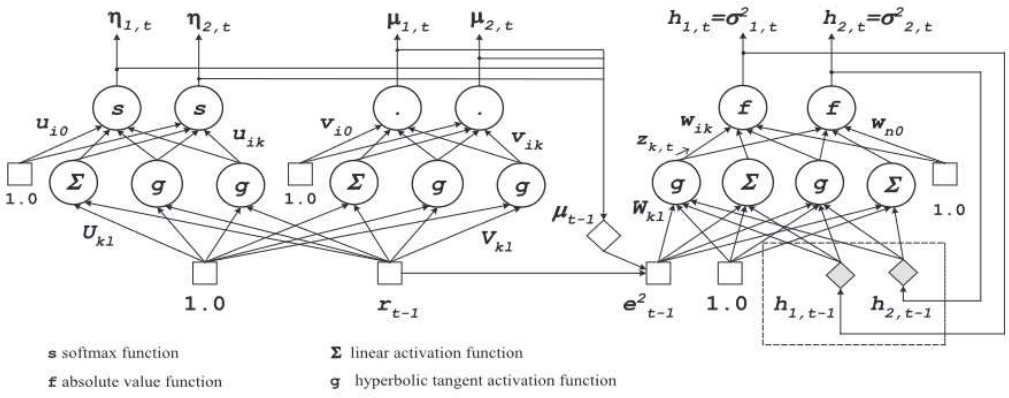

Figure 2.2: Architecture of the RMDN-GARCH

In Diagram 4.2, the round nodes correspond to neural network nodes, while the square nodes are input nodes. The input of the neural networks are the stock returns denoted as r_{t-l}, the error denoted as e_{t-l}^2, the feedback denoted $h_{1,t-l}$ and $h_{2,t-l}$, and the bias denoted as 1.0. The estimates for the mixing coefficients are denoted $\eta_{1,t}$, $\eta_{2,t}$, the estimates for the means denoted as $\mu_{1,t}$, $\mu_{2,t}$, and the estimates for the conditional variance are denoted as $\sigma_{1,t}$ and $\sigma_{2,t}$. The latter output are then used as input for the memory neuron of the recurrent network used.

2.4 Summary

This chapter introduces the reader to important concepts that will be central to this thesis. We first introduce the reader to the value-at-risk and the methods employed to estimate the value-at-risk, such as the historical method, variance-covariance method, and the Monte-Carlo method. We

then proceed to introduce the ARCH/GARCH family of models that can be used to forecast the value-at-risk. We also introduce the reader to the mixture density network and the recurrent mixture density network. The recurrent mixture density network will be the main forecasting method used throughout this thesis.

Chapter 3

Literature Review

As forecasting the value-at-risk requires predicting the returns of a stock portfolio, we provide a detailed summary of the literature related to time series forecasting used in financial contexts. The current financial time series forecasting literature addresses both univariate and multivariate forecasting problems. We specifically focus on the univariate time series forecasting literature adapted to financial forecasting problems, which we divide into two main groups. Our focus on financial forecasting problems is crucial, as certain phenomena, known as ”stylized facts” in the literature, are specific to financial time series. Three main stylized facts have been reported in the literature. Heteroskedasticity is commonly observed and is described as time-varying variance (Cont, 2001). The second stylized fact commonly reported is the presence of heavy tails in the distribution of stock returns (Cont, 2001). The last stylized fact is the ”leverage effect,” which can be described as the negative correlation between volatility measures and the assets' returns (Cont, 2001). The first group of models (a) comprises linear time series models such as the ARCH/GARCH family of models and regime switching models. The second group (b) comprises neural network models used for time series forecasting, such as recurrent neural networks, convolutional neural networks, and hybrids between the two different architectures. This group also includes the mixture density network and recurrent mixture density network, which is the focus of this thesis. The classification of financial time series forecasting methods is summarized in Table 3.3. We analyze several time series forecasting methods previously proposed in the literature to provide a broad view of the current state of the literature.

3.1 Classical Financial Time Series Model

Heteroskedasticity is a feature inherent in financial time series. The general approach to modeling heteroskedasticity is to model the conditional variance using a time series model such as the AR or ARMA model. This concept was first introduced within the ARCH model by Engle (1982), which involves modeling the conditional variance as an autoregressive process. Subsequently, the GARCH model was introduced by Bollerslev (1986) as a generalization of the ARCH model. The GARCH model improves on the ARCH model by modeling the conditional variance as an ARMA process (Bollerslev, 1986). This allows past realizations of conditional variance to influence the conditional variance equation. The GARCH model also sparked a large literature focusing on extensions of the GARCH model. These extensions aim to improve the GARCH model by modeling additional stylized facts in the conditional variance equation. Mainly, they attempt to model the "leverage effect," which is defined as negative returns having a larger impact on conditional volatility than positive returns (Engle & Ng, 1993).

The EGARCH model was first proposed by Nelson (1991). It allows for the conditional variance to account for asymmetry in the relationship between the returns and the variance. This stylized fact is called the leverage effect. Multiple models were developed to model the leverage effect. Those models include the GQARCH model (Sentana, 1995), the GARCH-M model (Engle, Lilien, & Robins, 1987), the GJR-GARCH model (Glosten, Jagannathan, & Runkle, 1993), and the TGARCH model (Zakoian, 1994). The GQARCH model generalizes the QARCH model as the GARCH model generalizes ARCH. The GQARCH models the asymmetry as an additive term, instead of modeling the asymmetry as a multiplicative heteroskedasticity term as in the EGARCH model (Sentana, 1995). The GARCH-M model introduces a heteroskedastic term to the mean, allowing the conditional variance to affect the mean. However, this model is only appropriate for time series that are autocorrelated (Engle et al., 1987). The GJR-GARCH improves on the GARCH-M model by integrating the nominal interest in the conditional variance equation, January and October dummy variables, and the unanticipated part of the stock returns (Glosten et al., 1993). The TGARCH model proposes to model the conditional variance as a sum of negative and positive

functions (Zakoian, 1994). Another extension to the GARCH model is the AP-ARCH model. The AP-ARCH model allows for longer memory in the conditional volatility dynamic by integrating the absolute returns rather than the square returns. It also models the asymmetry in the conditional volatility (Ding, Granger, & Engle, 1993).

Another type of model used extensively to forecast financial time series is the regime-switching models, also known as Markov switching models. First introduced by Goldfeld and Quandt (1973), the regime-switching model assumes that the time series can be decomposed into at least two different regimes, that is, two different states of the economy (growth, recession). Such models have gained traction in the literature since the seminal work of Hamilton (1989), who showed empirically that Markov switching models were well-suited for accurately estimating economic states. Newer models have been developed that combine the regime-switching dynamics with the GARCH, producing regime-switching GARCH models. Notably, Gray (1996) proposes using a Markov process to switch between two different regimes, each governed by a GARCH model. The model does not lead to path dependency, which would otherwise cause issues during estimation. A further improvement is the proposal of Klaassen (2002), which allows the model to incorporate more information in the GARCH process.

While the ARCH/GARCH family of models and the regime-switching models can provide reliable forecasts of the conditional variance, they are parametric models. This means that they require the specification of an error distribution, which in the case of the GARCH model is generally Gaussian. However, assuming a Gaussian error distribution implies that the time series is symmetric. It is also difficult to model some of the excess kurtosis or asymmetry in the distribution as is often observed in financial time series. To model time series with excess kurtosis and an asymmetric distribution, other error distributions have been proposed (Zivot, 2008). Namely, the student-t distribution (Bollerslev, 1987) and the asymmetric student-t distribution (Fernandez & Steel, 1998) have been proposed to model the error distribution. However, the choice of a distribution is specific to a particular time series and should be appropriate for the time series (Zivot, 2008). The choice of a distribution is also static in that it does not change over time other than with the conditional variance.

3.2 Neural Networks Financial Time Series Model

Multiple neural network architectures have been proposed to forecast univariate financial time series. Notably, recurrent neural networks (RNN) (Fjellström, 2022; Paranhos, 2021), convolutional neural networks (CNN) (J.-F. Chen, Chen, Huang, Huang, & Chen, 2016; Zeng et al., 2023), and more complex neural networks (Lu, Li, Li, Sun, & Wang, 2020) have been successfully used to forecast financial time series. Recurrent neural networks are well-suited for this task as they were initially developed to analyze sequential data and require minimal modification to forecast time series (Ge, Lalbakhsh, Isai, Lenskiy, & Suominen, 2022). The recurrent neural network has since been modified to incorporate longer memory first with the long-short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and then with the gated recurrent unit (GRU) architecture (Cho et al., 2014). The LSTM architecture has been commonly applied to volatility forecasting problems (Di-Giorgi et al., 2023; Jia & Yang, 2021), inflation forecasting problems (Paranhos, 2021), and stock price predictions (Kamalov et al., 2020). CNN, on the other hand, requires extensive modifications and/or preprocessing to be used for time series, making it a less attractive option for time series forecasting. Initially developed for image processing applications, single-dimension convolutions and transformation of the time series to 2 dimensions have been proposed to model time series using CNN (Ge et al., 2022). Notably, S. Chen and He (2018) proposed using convolutional neural networks to forecast the prices of stocks listed on the Chinese market. Similar financial forecasting applications with CNN, such as stock returns forecasting, have also been proposed (Vijh, Pandey, Vijh, & Kumar, 2021). Hybrid approaches have combined the CNN and LSTM architecture for forecasting financial time series (Liu, Zhang, & Ma, 2017; Vidal & Kristjanpoller, 2020). However, the neural network approach mentioned above only forecasts the expected value of the variable being studied. Newer models proposed include the Deep AR model (Salinas, Flunkert, Gasthaus, & Januschowski, 2020), the Generative Adversarial Network (Kumar et al., 2021; Tovar, 2020), and transformer based model (Muhammad et al., 2023; Zeng et al., 2023). Those approaches have been successfully used for various forecasting applications including demand forecasting and stock price

forecasting.

Another approach to time series forecasting is the mixture density network model (MDN). Unlike RNN and CNN, mixture density networks forecast the shape of the conditional distribution. First introduced by Bishop (1994), they were originally developed to model arbitrary conditional distributions of data for non-linear regression. Most time series forecasting methods incorporate a linear layer as output; the mixture density network differs in that regard, as the output is a mixture layer that outputs a probability distribution. Since its introduction, the mixture density network has been extended to model serial dependence in the data using a recurrent neural network in the input layer. This approach, called a recurrent mixture density network (RMDN), has led to its adoption for time series forecasting. Notably, Schittenkopf et al. (2000) introduced an architecture that nests the GARCH model and applied it to the FTSE 100 index returns. Other financial applications of recurrent mixture density networks include forecasting volatility (Mostafa & Dillon, 2008) and energy prices (Brusaferri, Matteucci, Ramin, Spinelli, & Vitali, 2020). The recurrent mixture density network method is attractive for forecasting financial time series due to its semi-parametric nature and its ability to model non-constant variance and discontinuity in the data. These features are particularly useful for time series forecasting and parallel the ARCH/GARCH family of models and the regime-switching models. However, mixture density networks are notoriously difficult to implement and are prone to instability during training due to underflow and the persistent NaN problem (Guillaumes, 2017). The simplification of the loss function using the log-sum-exp trick (Blanchard, Higham, & Higham, 2019) has been proposed to resolve issues with underflow. The persistent NaN problem is, however, more complex to solve and requires a multi-pronged approach, which includes replacing the output activation function with an exponential linear unit (ELU)(Clevert, Unterthiner, & Hochreiter, 2015), applying weight regularization to the variance network, and simplifying the distribution function (Guillaumes, 2017). These strategies are explored in further detail in this thesis and are discussed further in Chapter 4.

3.3 Summary

This chapter surveys the most important models in the literature related to time series forecasting methods and their applications to financial time series, covering linear time series to recent developments with neural network-based forecasting methods. We discuss the challenges of using neural networks for forecasting time series and promising methods such as mixture density networks, which bridge the gap left by linear time series methods by leveraging neural networks and also avoiding some of the pitfalls commonly encountered when forecasting time series. This thesis focuses on forecasting financial time series using recurrent mixture density networks. More specifically, we propose an effective method for accurately forecasting the value-at-risk of an equities portfolio using recurrent mixture density networks. This method was selected to avoid the pitfall of selecting a distribution and dynamics for the conditional variance, thus producing more accurate results than currently proposed methods.

Ref.	Specific Objective	Category
(a) Linear Time Series Models		
(Engle, 1982) (Bollerslev, 1986)	Model heteroskedastic time series	ARCH/GARCH model
(Engle et al., 1987) (Sentana, 1995) (Glosten et al., 1993) (Zakoian, 1994) (Nelson, 1991) (Higgins & Bera, 1992)	Model heteroskedastic time series and the "leverage" effect	ARCH/GARCH model
(Goldfeld & Quandt, 1973) (Hamilton, 1989) (Gray, 1996) (Klaassen, 2002)	Model regime change in time series	Regime-Switching model
(b) Neural Network Models		
(Fjellström, 2022) (Paranhos, 2021) (Luo et al., 2018) (Rodikov & Antulov-Fantulin, 2022) (Datta, 2022) (Di-Giorgi et al., 2023) (Jia & Yang, 2021)	Model volatility of financial time series	Recurrent Neural Network
(Vijh et al., 2021) (S. Chen & He, 2018)	Model financial time series	Convolutional Neural Network
(Vidal & Kristjanpoller, 2020) (Liu et al., 2017) (Lu et al., 2020)	Model financial time series	Hybrid Neural Network
(Salinas et al., 2020) (Kumar et al., 2021) (Tovar, 2020) (Zeng et al., 2023) (Muhammad et al., 2023)	Model financial time series	Misc. Neural Network
(Schittenkopf et al., 2000) (Nikolaev et al., 2012) (Mostafa & Dillon, 2008) (Brusaferri et al., 2020)	Models the conditional distribution of financial time series	Mixture Density Network

Table 3.1: Classification of the literature

Chapter 4

Value-at-Risk Forecasting Model and Methodology

This chapter outlines the models and methodology utilized in this thesis. We commence by introducing the previously proposed RMDN architecture, which forms the foundation for the model proposed in this work. The initial section delineates the neural network architecture of the RMDN employed to predict the parameters of the mixture. Subsequently, we elaborate on the training procedure implemented for the neural network. We then introduce the copula model designed to account for interdependence among the assets within each independent univariate RMDN model. Finally, we provide a detailed explanation of the sampling method utilized for estimating the value-at-risk.

4.1 Base RMDN Architecture

In this section, we introduce the architecture of two interconnected recurrent mixture density networks proposed previously. Subsequently, we discuss the limitations associated with these architectures.

The first recurrent mixture density network architecture proposed for modeling financial time series is the architecture of Schittenkopf et al. (2000). Their model implements the ideas surrounding

the GARCH model, while being capable of modeling a variety of continuous conditional densities and higher-order moments. The RMDN proposed by Schittenkopf et al. (2000) estimates the conditional density

$$f(r_{t+1}|R_t) = \sum_{i=1}^{N} \hat{\eta}_{i,t+1} k(\hat{\mu}_{i,t+1}, \hat{\sigma}^2_{i,t+1}) \tag{8}$$

where the parameters $\hat{\eta}_{i,t+1}$, $\hat{\mu}_{i,t+1}$, and $\hat{\sigma}^2_{i,t+1}$ are, respectively, the mixing coefficients, the mean, and the conditional variance of the component i of a mixture of N distributions. Each distribution is described by the density function $k(\mu, \sigma^2)$. We define R_t is the information available at time t in the returns series.

Each of these parameters is estimated using a multi-layer perceptron. The authors specify the use of two layers for each neural network. The first layer of each neural network is followed by a tanh activation function. However, there are slight differences in the input and output layers of each neural network. The first neural network is employed to estimate the parameter $\eta_{i,t+1}$, taking the returns r_t as input. The hidden layer comprises three nodes, and the output layer consists of N nodes. It concludes with a softmax activation function, ensuring that the sum of the weights equals 1. The second neural network is used to estimate the parameter $\mu_{i,t+1}$, taking returns r_t as input. Similar to the first neural network, its hidden layer has three nodes, and the output layer contains N nodes, concluding with a linear activation function. The linear activation function is chosen as the mean is defined on the set of real numbers $\mathbb{R}$.The third neural network is employed to estimate the parameter $\hat{\sigma} i, t+1^2$, taking the previous error $\varepsilon_t^2 = (r_t - \overline{\mu} t)^2$ as input. Additionally, the variance recurrent network takes the previous estimates for the conditional variance $h_{i,t} = \hat{\sigma}^2_{i,t}$ as input, introducing recurrence into the architecture. The hidden layer consists of 4 nodes, and the output layer contains n nodes. The output layer is passed through an absolute function to ensure that the variance is positive. Figure 4.1 provides a detailed illustration of the architecture and interconnections.

The second architecture proposed in the literature is the RMDN-GARCH (Nikolaev et al., 2012). Building upon the architecture proposed by Schittenkopf et al. (2000), the RMDN-GARCH enhanced the RMDN by introducing linear nodes in the hidden layer. This modification enables

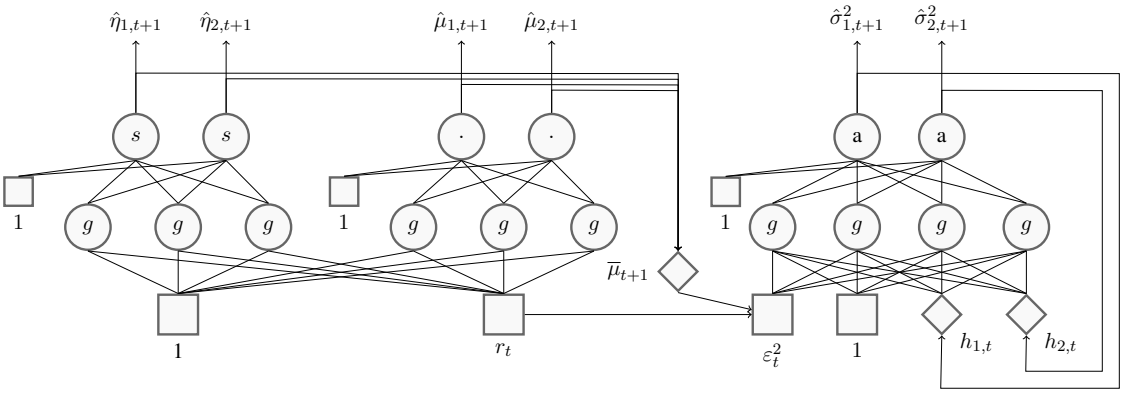

Figure 4.1: Architecture of the original RMDN

the model to encompass the AR-GARCH model, as illustrated in Figure 4.2. Another significant distinction is the training algorithm employed. The RMDN-GARCH is trained using the RTRL algorithm (Williams & Zipser, 1989, 1995). In contrast, the preceding model is trained through backpropagation through time. The key difference between these two algorithms lies in the fact that the RTRL algorithm allows for online learning, while the backpropagation through time algorithm is limited to offline learning.

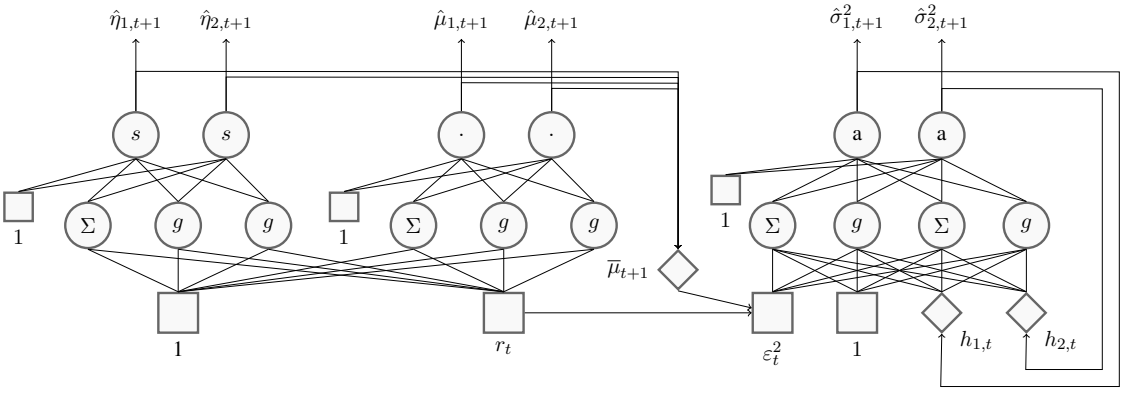

Figure 4.2: Architecture of the RMDN-GARCH

The RMDN-GARCH architecture has been shown to perform well in comparison with the

GARCH model (Nikolaev et al., 2012). However, early tests on a selected number of stocks indicated inconsistent results across different datasets. In some datasets, we were unable to train the RMDN-GARCH as it converged to NaN. The results of these preliminary tests are reported in Table 4.1. The outcomes suggest numerical stability issues during training, a problem previously identified with mixture density networks (Guillaumes, 2017). Another issue with the RMDN-GARCH was its convergence to a lower log-likelihood than the GARCH model. As the RMDN-GARCH model nests the GARCH model, the log-likelihood of the RMDN-GARCH should be at least equal to the log-likelihood of the AR-GARCH model. This indicates that the RMDN-GARCH converges, in some instances, to a local minimum. In the following section, we propose a new architecture based on the RMDN-GARCH that improves the numerical stability of the model. Additionally, we introduce a pretraining method in Section 4.3 to enhance the convergence of the model.

Stock Ticker	Not Converged	Converged
AKAM	7	3
CBRE	6	4
EA	7	3
EMN	6	4
K	7	3
LOW	7	3
MKTX	7	3
NWSA	7	3
OKE	7	3
UAL	8	2
Total	69	31
Total%	69%	31%

Table 4.1: In-Sample Test

4.2 Stable-RMDN-GARCH

In this section, we present enhancements to the RMDN-GARCH architecture. The RMDN-GARCH architecture, chosen for its inclusion of linear nodes in the hidden layer, possesses a crucial advantage, detailed in Section 4.3. We introduce the improvements aimed at enhancing the numerical stability of the loss function, and we henceforth refer to this refined architecture as Stable-RMDN-GARCH (S-RMDN-GARCH).

Following the approach of Nikolaev et al. (2012), we devise a recurrent mixture density network architecture to forecast the conditional density of stock returns through mixtures of Gaussian distributions. The proposed architecture, while closely aligned with the RMDN-GARCH, incorporates recent advancements in mixture density network architecture (Guillaumes, 2017). The S-RMDN-GARCH architecture comprises three modules, each dedicated to estimating different parameters of a mixture of N components. In our consideration, we focus on a mixture of Gaussian distributions, characterized by three sets of parameters: the weight of each component $\hat{\eta}t+1 = (\hat{\eta}1, t+1, \hat{\eta}2, t+1, ..., \hat{\eta}N, t+1)$, the mean of each component $\hat{\mu}t+1 = (\hat{\mu}1, t+1, \hat{\mu}2, t+1, ..., \hat{\mu}N, t+1)$, and the variance of each component $\hat{\sigma}t+1^2 = (\hat{\sigma}1, t+1^2, \hat{\sigma}2, t+1^2, ..., \hat{\sigma}N, t+1^2)$. The probability density function of a Gaussian mixture distribution with N components is defined as

$$f(r_t|R_{t-1}) = \sum_{i=1}^{N} \eta_i \phi(r_t; \mu_i, \sigma_i^2) \tag{9}$$

where ϕ is the probability density function of the univariate Gaussian distribution. Similarly, the cumulative distribution function of the Gaussian mixture density distribution is defined

$$F(r_t|R_{t-1}) = \sum_{i=1}^{N} \eta_i \Phi(r_t; \mu_i, \sigma_i^2) \tag{10}$$

where Φ is the cumulative density function of the univariate Gaussian distribution.

The conditional mean and variance of the model are respectively given by $\overline{\mu}_{t+1} = \sum_{i=1}^{N} \hat{\eta}_{i,t+1}\hat{\mu}_{i,t+1}$ and $\overline{\sigma}_{t+1}^2 = \sum_{i=1}^{N} \hat{\eta}_{i,t+1}(\hat{\sigma}_{i,t+1}^2 + (\hat{\mu}_{i,t+1} - \overline{\mu}_{t+1})^2)$. The input of the S-RMDN-GARCH is a time series $\{r_t\}_{t=1}^{T}$. The first part of the S-RMDN-GARCH is the mixing network, which takes for input the last observation r_{t-1} and estimates the parameters weights of each mixture component $\hat{\eta}_{t+1}$. In order to ensure that the condition $\sum_{i=1}^{N} \hat{\eta}_{i,t+1} = 1$ is satisfied, this output layer of this network is followed by a softmax activation function. The softmax activation function is defined as

$$s_i(y) = \frac{e^{y_i}}{\sum_{j=1}^{N} e^{y_j}} \tag{11}$$

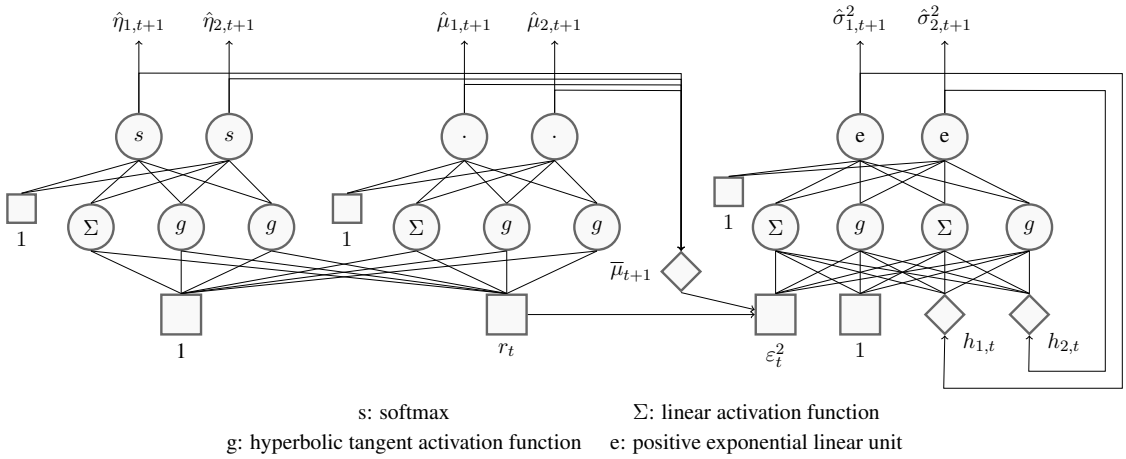

Figure 4.3: Architecture of the proposed RMDN

for every i in $1, ..., N$ (M. Bishop, 2006). The estimate $\hat{\eta}_{t+1}$ is thus defined as

$$\hat{\eta}_{i,t+1} = s_i \left(u_{n,1}(U_{1,1} r_t + U_{1,0}) + \sum_{k=2}^{K} u_{n,k} \tanh(U_{k,1} r_t + U_{k,0}) + u_{n,0},\ n = 1, \ldots, N \right) \quad (12)$$

where U are the input-to-hidden layer weight and u are the hidden-to-output layer weight. The second part of the model is the mean-level network, which takes for input r_t and estimates the mean $\hat{\mu}_{t+1}$ of each mixture component. The estimate $\hat{\mu}_{t+1}$ is thus defined as

$$\hat{\mu}_{i,t+1} = v_{i,1}(V_{1,1} r_t + V_{1,0}) + \sum_{k=2}^{K} v_{i,k} \tanh(V_{k,1} r_t + V_{k,0}) + v_{i,0} \quad (13)$$

where V are the input-to-hidden layer weight and v are the hidden-to-output layer weight. The third part of the S-RMDN-GARCH is the variance recurrent network, which estimates the variance of each component of the mixture $\hat{\sigma}^2_{t+1}$. As in the GARCH model, this network takes for input the innovation $\varepsilon^2_t = (r_t - \overline{\mu}_t)^2$ and the last estimate for the conditional variance $\hat{\sigma}^2_t$. The output layer is fed through what we refer to as the positive exponential linear unit, which is defined as $e(x, \alpha) = ELU(x, \alpha) + 1 + \epsilon$ where ϵ is a small number and $ELU(x, \alpha)$ is the exponential linear unit defined as

$$ELU(x, \alpha) = \begin{cases} x & \text{if } x > 0, \\ \alpha(e^x - 1) & \text{otherwise.} \end{cases} \quad (14)$$

The hyperparameter α controls the saturation value for negative inputs (Clevert et al., 2015). The ELU activation function improves the numerical stability of MDN and ensures that the estimated variance is always greater than 0 (Guillaumes, 2017; Joseph, 2021). We refer to this activation function as the "positive exponential linear activation unit" throughout this paper. The estimate for $\hat{\sigma}_{i,t+1}$ is defined as

$$\begin{aligned}\hat{\sigma}_{i,t+1} = e\Bigg(& w_{i,1}(W_{1,1}\varepsilon_t^2 + W_{1,0}) + \sum_{k=2}^{K} w_{i,k}\tanh(W_{k,1}\varepsilon_t^2 + W_{k,0}) \\ & + w_{i,K+1}(W_{K+1,1}\hat{\sigma}_{i,t}^2 + W_{K+1,0}) + \sum_{k=K+2}^{2K} w_{i,k}\tanh(W_{k,1}\hat{\sigma}_{i,t}^2 + W_{k,0}) + w_{i,0}\Bigg)\end{aligned} \tag{15}$$

where w are the hidden-to-output weights and W are the input-to-hidden weights. A detailed graphical representation of the architecture is presented in Figure 4.3. The model is implemented using PyTorch (Paszke et al., 2017). As in the RMDN-GARCH, the S-RMDN-GARCH is trained by minimizing the negative log-likelihood. Assuming a mixture of N Gaussian components and denoting by $\phi(\cdot, \mu, \sigma^2)$ the density of a Gaussian distribution with mean μ and variance σ^2, the negative log-likelihood $-\ell$ is given by

$$\begin{aligned}-\ell &= \sum_{t=1}^{T} -\log \sum_{i=1}^{N} \hat{\eta}_{i,t}\phi(r_t, \hat{\mu}_{i,t}, \hat{\sigma}_{i,t}^2) \\ &= \sum_{t=1}^{T} -\log\left(\sum_{i=1}^{N} exp\left[\log(\hat{\eta}_{i,t}) - \frac{1}{2}\log(2\pi) - \log(\hat{\sigma}_{i,t}) - \frac{1}{2}\frac{(r_t - \hat{\mu}_{i,t})^2}{\hat{\sigma}_{i,t}^2}\right]\right).\end{aligned} \tag{16}$$

The formulation (16) makes it possible to use the *logsumexp* trick (Blanchard et al., 2019) to avoid underflow issues. We use the *logsumexp* implementation available in the PyTorch library (Paszke et al., 2017) as a direct replacement of the term $log\sum_{i=1}^{N} exp$ in the loss function. The S-RMDN-GARCH is trained with the Adam optimizer (Kingma & Ba, 2014) without weight regularization. However, conventional training method do not guarantee that the model will converge optimally.

4.3 Proposed Linear Reduction Pretraining

As previously noted, a well-known issue with mixture density networks is their tendency to converge to local minima, which may be suboptimal (Hepp, Zierk, Rauh, Metzler, & Seitz, 2022). This section outlines a pretraining method for the S-RMDN-GARCH aimed at circumventing local minima that would yield a log-likelihood lower than that of the GARCH model. The S-RMDN-GARCH model is a generalization of the Gaussian AR-GARCH model, with the latter being nested within the former. Consequently, the optimized log-likelihood of the S-RMDN-GARCH(1, 1, N) should always be greater than or equal to that of an AR(1)-GARCH(1,1).[1] However, in initial experiments, the S-RMDN-GARCH(p, q, N) did not consistently converge to a minimum with a log-likelihood greater than that of the AR(d)-GARCH(p, q). The model exhibited significant instability, leading to the so-called 'persistent NaN problem,' a common challenge with MDN as defined by Guillaumes (2017). In this section, we first present the mathematical relationship between the AR(1)-GARCH(1,1) and the S-RMDN-GARCH. We then introduce a pretraining method that leverages this relationship to enhance the convergence of the S-RMDN-GARCH.

We first show the implications that using a linear node in the hidden layer implies that the AR(1)-GARCH(1, 1) model is a particular case of the S-RMDN-GARCH model. Consider an AR(1)-GARCH(1, 1) defined as

$$
\begin{aligned}
r_{t+1} &= \mu_{t+1} + \sigma_{t+1}\varepsilon_{t+1}, \\
\mu_{t+1} &= a_0 + a_1 r_t, \\
\sigma^2_{t+1} &= \alpha_0 + \alpha_1 \varepsilon^2_t + \beta_1 \sigma^2_t,
\end{aligned}
\tag{17}
$$

where $\varepsilon_t \sim N(0,1)$ denotes the innovation. Now consider an S-RMDN-GARCH with only 1 component and with only the linear nodes in the hidden layer of each network being used. We achieve this by setting $V_i = 0$ for $1 < i \leq K$ and $W_i = 0$ for $1 < i \leq K$ and for $K + 2 < i \leq 2K$.

[1] The proposed method can easily extend to S-RMDN-GARCH(p, q, N) models.

Since only one component is considered, we set the mixture weight to $\hat{\eta}_{t+1} = (1, 0)$. The RMDN-GARCH can then be reformulated mathematically as

$$
\begin{aligned}
r_{t+1} &= \mu_{t+1} + \sigma_{t+1}\varepsilon_{t+1} \\
\hat{\mu}_{t+1} &= v_1(V_1 r_t + V_0) + v_0 \\
\hat{\sigma}^2_{t+1} &= e(w_1(W_1\varepsilon_t^2 + W_0) + w_2(W_2\hat{\sigma}_t^2 + W_3) + w_0) + 1.001
\end{aligned}
\tag{18}
$$

For simplicity we set the parameter ϵ of the positive ELU activation function to 0.001. The positive ELU activation function can be reformulated to $ELU(x) + 1$. Such representation of the variance $\hat{\sigma}^2_{t+1}$ is equivalent to the conditional variance of a GARCH(1,1) when the input of the ELU is greater than 0. The conditional mean $\hat{\mu}_{t+1}$ also simplifies to an expression equivalent to an AR(1) model. This implies that the RMDN-GARCH(1, 1) reduces to an AR(1)-GARCH(1, 1) when the input of the ELU is greater than 0[2].

We propose a pretraining approach for the S-RMDN-GARCH that leverages the linear nodes in the hidden layer of each neural network. The rationale behind this method is rooted in the belief that the parameters of a linear S-RMDN-GARCH could serve as a robust starting point for the parameters of a non-linear S-RMDN-GARCH. This assumption is based on the notion that an optimal linear S-RMDN-GARCH model should be equivalent to an AR(1)-GARCH(1,1) under certain conditions. This approach mirrors the strategy employed by Nikolaev et al. (2012), who used the parameters of a GARCH model as the initial values for the RMDN-GARCH. However, this method is not suitable for the S-RMDN-GARCH due to its specific relationship with the GARCH model, which is subject to restrictions. Moreover, our pretraining method is more practical, requiring no estimation of another model.

To initiate the pretraining process for the S-RMDN-GARCH, we commence by randomly initializing all linear nodes based on a normal distribution. Concurrently, the non-linear nodes are set to zero to exclude them from the pretraining phase, as they should not be considered at this stage.

[2]Without loss of generality we can set v_1, w_1,w_2 to 1 and V_0, W_0, W_3 to 0.

Throughout the pretraining phase, the non-linear nodes in the hidden layer of each neural network remain frozen. This is achieved by consistently setting the gradients of the nodes with non-linear activation functions to zero whenever gradients are computed. The purpose of this step is to ensure that the parameters of the non-linear activation nodes remain unaltered during training. Subsequently, the model undergoes training employing all nodes in the hidden layer. A detailed analysis comparing the training approach with our proposed pretraining method is presented in Section 5.2.

4.4 Multivariate Extension of the Proposed RMDN

The estimation of the value-at-risk for a portfolio necessitates multivariate forecasts. However, the S-RMDN-GARCH operates as a univariate model. To address this limitation, we propose an extension that incorporates a Gaussian copula. This extension allows us to preserve the semi-parametric characteristics of the S-RMDN-GARCH while introducing the co-dependence of assets within the portfolio into the forecasts. This approach is scalable to high dimensions, making it suitable for modeling large portfolios. We refer to this extended model as the copula-S-RMDN-GARCH.

Copulas, initially introduced by Sklar (1959), provide a framework for modeling joint distributions using a function C and marginal distributions $F_1, F_2, ..., F_n$. They are employed to specify the marginal distribution of a vector of random variables independently of the interdependence between each random variable. Given a vector of random variables $X = (X_1, X_2, ..., X_n)$, the joint distribution can be expressed in terms of its marginal distribution F and a copula C defined on the interval $[0, 1]$ as

$$G(X_1, X_2, ..., X_n) = C(F_1(X_1), F_2(X_2), ..., F_n(X_n)) \tag{19}$$

The copula C needs to satisfy two conditions:

(1) $C(0, 0, ..., 0) = 0$,

(2) $C(1, ..., 1, \alpha, 1, ..., 1) = \alpha$.

There are two main families of copula, the elliptical copulas and the Archimedean copulas. The

elliptical family comprises the Gaussian copula and the Student-T copula. Among the Archimedean family comprises the Gumbel, Frank and Clayton copulas are the most common. Our focus is on the Gaussian copula as it will be used throughout this thesis. Given a vector of random variable $X = (X_1, X_2, ..., X_n)$, the marginal probability of $U = (U_1, U_2, ..., U_n)$ is obtained by the probability integral transform defined as

$$(U_1, U_2, ..., U_n) = (F_1(x_1), F_2(x_2), ..., F(x_n)). \tag{20}$$

The Gaussian copula C is defined as

$$C(u) = \Phi_R \left(\Phi^{-1}(u_1), ..., \Phi^{-1}(u_n)\right) \tag{21}$$

where Φ_R is the cumulative distribution function of a joint Gaussian distribution with mean vector equal to 0 and covariance matrix equal to the correlation matrix R, Φ^{-1} is the cumulative distribution function of a Gaussian distribution with mean equal to 0 and variance equal to 1 (Xue-Kun Song, 2000). The density of the Gaussian copula is defined as

$$c(u, R) = \frac{1}{\sqrt{|R|}} e^{-\frac{1}{2} \cdot q^T (R^{-1} - I) q} \tag{22}$$

where $q = (\Phi^{-1}(u_1), ..., \Phi^{-1}(u_n))^T$ and I is an $n \times n$ identity matrix (Xue-Kun Song, 2000). The probability density function of a multivariate mixture distribution with Gaussian copula can be defined as

$$f_c = c(u, R) \cdot \prod_{i=1}^{n} f(x; W_i, M_i, V_i) \tag{23}$$

where f is the probability density function of a mixture of Gaussian distribution and W, M, and V are set the set of parameters of for all the univariate mixture . We define those parameters$W = (\eta_1, ..., \eta_n)$, $M = (\mu_1, ..., \mu_n)$, and $V = (\sigma_1^2, ..., \sigma_n^2)$.

The remainder of this sections is divided into two parts. In the first part, we present a parameter estimation method, while the second part presents the algorithm for directly sampling from the multivariate model.

4.4.1 Copula Parameter Estimation

To estimate the parameters of a Gaussian copula, we first need to forecast the conditional distribution of asset returns using the RMDN. The set of all the asset returns conditional distribution is the marginal distribution . Let $R = (R_1, R_2, ..., R_N)$ be a set of time series with marginal cumulative distribution function F_i. To estimate the parameter matrix ρ of a Gaussian copula, each time series $R_i \in \mathbb{R}$ need to be transformed to a i.i.d. normally distributed set of observation. This is achieved with the Rosenblatt transform. Let's define the time series $R_i = \{r_{i,t}\}_{t=1}^{T}$ with marginal cumulative distribution F_i and its copula C, the Rosenblatt transform Ω is defined as

$$\Omega(R) = \begin{Bmatrix} \Phi^{-1}(F_{1,1}(r_1)) \\ \Phi^{-1}(F_{1,2}(r_2|r_1)) \\ ... \\ \Phi^{-1}(F_{1,T}(r_T|r_1, ..., r_{T-1})) \end{Bmatrix} \tag{24}$$

where Φ^{-1} is the inverse cumulative distribution function of a Gaussian distribution with the mean equal to 0 and the variance equal to 1 (Rosenblatt, 1952). This definition of the Rosenblatt transform is appropriate for non-Markovian models. However, the RMDN is Markovian as the conditional probability distribution estimated is defined as $F(r_{t+1}|r_t)$. The appropriate definition of the Rosenblatt transform for a Markovian model is defined as

$$\Omega(R) = \begin{Bmatrix} \Phi^{-1}(F_1(r_1)) \\ \Phi^{-1}(F_2(r_2|r_1)) \\ ... \\ \Phi^{-1}(F_n(r_n|r_{n-1}, ..., r_{n-l})) \end{Bmatrix} \tag{25}$$

where l is the maximal lag used in the model. After transforming the time series to normally distributed random variates, we estimate the correlation matrix ρ using Pearson's correlation coefficient. The sample correlation between a pair of random variable x and y is estimated with the following formula $\rho(x, y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x}^2)\sum_{i=1}^{n}(y_i - \overline{y})^2}}$ where n is the number of observations for each variable and $\overline{x}$ and $\overline{y}$ are the estimates for the mean of x and y. Let $F(r_t|r_{t-1}; w, \mu, \sigma^2)$ be the

cumulative distribution function of the mixture of the RMDN, where w is the weight of all distributions in the mixture, μ is the mean of all distributions and σ^2 is the conditional variance for all distributions. The Rosenblatt transform can be reformulate to

$$\Omega(R; W, M, V) = \begin{Bmatrix} \Phi^{-1}(F(r_1; W_1, M_1, V_1)) \\ \Phi^{-1}(F_2(r_2|r_1; W_2, M_2, V_2)) \\ ... \\ \Phi^{-1}(F_n(r_n|r_{n-1}, ..., r_{n-l}; W_n, M_n, V_n)) \end{Bmatrix} \tag{26}$$

where W, M, V are vectors of weights, means, and the conditional variances respectively. The algorithm for the estimation of the Gaussian copula is available in algorithm 1.

Algorithm 1: Estimation of the Copula's Parameters

Data: X, a multivariate time series $\{x_0\}_0^T$, ..., $\{x_N\}_0^T$
Data: σ_0^2, the initial hidden state of the S-RMDN-GARCH at time $t = 0$
Data: R, a set of N trained RMDN on each time series in X
Result: ρ
$Z = [\ ]$;
for $i \leftarrow 0$ **to** N **do**
    $W \leftarrow [\ ]$;
    $M \leftarrow [\ ]$;
    $V \leftarrow [\ ]$;
    for $t \leftarrow 0$ **to** T **do**
        $\eta,\ \mu,\ \sigma^2 \leftarrow$ S-RMDN-GARCH$(X_{i,t}, \sigma^2)$;
        $W_t \leftarrow \eta$;
        $M_t \leftarrow \mu$;
        $V_t \leftarrow \sigma^2$;
    end
    $Z_i \leftarrow \Omega(X_i, W, M, V)$;
end
$\rho \leftarrow corr(Z)$;

4.4.2 Sampling

To sample from the copula-S-RMDN-GARCH, we need to perform the inverse operation compared to estimating the copula's parameters. This operation is carried out sequentially, considering the model's Markovian nature and its dependence on previous estimates. Given an initial set of

observations X, we can generate a sample path using Algorithm 2.

Algorithm 2: Single Path Sampling Algorithm

Data: ρ, the correlation matrix of U
Data: σ_0^2, the initial hidden state of the S-RMDN-GARCH at time $t = 0$
Data: X, the initial set of observations at time $t = 0$
Data: T, the horizon over which we want to sample
Result: Y, the samples generated
$Y \leftarrow [\ ]$;
for $i \leftarrow 1$ **to** n **do**
  $Y_{i,0} \leftarrow X_i$;
end
for $t \leftarrow 1$ **to** T **do**
  $Z \leftarrow randn()$, sample from $N(0, R)$;
  for $i \leftarrow 1$ **to** n **do**
    $U_i \leftarrow \Phi(Z_i)$;
    $\eta,\ \mu,\ \sigma^2 \leftarrow$ S-RMDN-GARCH$_i(Y_{i,t-1}, \sigma^2)$;
    $Y_{i,t} \leftarrow F^{-1}(U_i;\ \eta, \mu, \sigma^2)$
  end
end

The method can be repeated multiple time to generate more sample paths. However, this method requires computing the inverse CDF of the mixture. As this is done numerically, it is not an efficient algorithm for generating a large number of samples. The estimation of a large portfolio requires sampling a large number of samples. We proposed an importance sampling algorithm in the next section to estimate the value-at-risk more efficiently.

4.5 Efficient Sampling of the copula-S-RMDN-GARCH and Estimation of the Value-at-Risk

In order to estimate the value-at-risk, a large number of samples are required. The previous method proposed for sampling requires computing the inverse cumulative distribution function of the mixture distribution, which can only be done numerically. The inverse cumulative distribution function needs to be computed for each path generated $N \cdot M$ times, where N is the number of assets and M is the number of periods in each path. As a large number of paths are required to

estimate the value-at-risk and the estimation of the inverse cumulative distribution functions is done numerically, sampling directly from the copula-S-RMDN-GARCH is not practical. We present an importance sampling algorithm for estimating the value-at-risk efficiently.

Importance sampling is a variance reduction technique that allows estimating the properties of a distribution (the target distribution) from samples generated from a different distribution (the design distribution). This method can improve the convergence of a Monte Carlo estimate by specifying a distribution with higher density in the area of interest. It can also be used to sample from a distribution that is difficult or inefficient to sample from, as is the case with the copula-S-RMDN-GARCH. To apply this variance reduction technique to the copula-S-RMDN-GARCH, we propose sampling from a multivariate Gaussian distribution. The multivariate Gaussian distribution, as it has the same copula as the copula-S-RMDN-GARCH, can be sampled from efficiently, thereby improving the efficiency of the sampling algorithm.

Let X be a random variable with the pdf f_X and h be a function defined on $\mathbb{R}$. To estimate the expectation of $h(X)$ noted $E[h(X)]$, a Monte-Carlo estimator can be defined as

$$E[h(X)] = \int h(x) f_X(x) dx \approx \frac{1}{N} \sum_{i=1}^{N} h(X_i) \tag{27}$$

where $X_1, ..., X_N$ is a sample drawn from f_X and N is the number of sample. Let Z be a random variable with with the pdf f_Z. An unbiased estimator of $E[h(X)]$ can be constructed from the samples $Z_1, ..., Z_N$ by using the following relationship

$$E[h(X)] = \int h(x) f(x) dx = \int h(x) \frac{f_X(x)}{f_Z(x)} f_Z(x) dx \approx \frac{1}{N} \sum_{i=1}^{N} h(Z_i) \frac{f_X(Z_i)}{f_Z(Z_i)} \tag{28}$$

If distribution of f_Z is chosen appropriately, the new unbiased estimator will reduced the variance of the Monte-Carlo estimates. Therefore, reducing the number of samples required to get an accurate estimate for $E[X]$ (Glasserman, 2004).

As the definition of the value-at-risk differs from the expectation of the random variable X, a different importance sampling estimator is needed. First, lets define the value-at-risk for a given random variable X as

$$VaR_\alpha(X) = inf\{x \in \mathbb{R} | P[X > x] \leq \alpha\} \tag{29}$$

where α is the level of the value-at-risk (Föllmer & Schied, 2016). Given this definition, the importance sampling estimator for the value-at-risk of X from a sample $Z_1, ..., Z_N$ drawn from the distribution Z is defined as

$$VaR_\alpha(X) \approx inf\{z \in \mathbb{R} | \frac{1}{N} \sum_{i=1}^{N} I(Z < z)q \leq \alpha\} \tag{30}$$

where q is the likelihood ratio. Let $Z(0, \rho)$ be a multivariate Gaussian distribution with pdf f_Z and let $U(, \rho, \eta, \mu, \sigma^2)$ be a multivariate mixture distribution with a Gaussian copula with pdf f_U. The likelihood ratio for the copula-S-RMDN-GARCH is specified as

$$q = \frac{f_U(Z; \rho, \eta, \mu, \sigma^2)}{f_Z(Z; 0, \rho)} \tag{31}$$

In order to estimate the value-at-risk at a given time-step t, we sample and estimate the likelihood ratio q_t. This is done in a two steps process. First, we forecast the parameters of the mixture distribution η, μ and σ^2. We then generate P samples from our design distribution Z_i. The second step consist in calculating the likelihood ratio q. The last step consist in evaluating the value at risk from the adjusted samples. The complete algorithm for estimating all the steps for time-step T is explained in Algorithm 3.

The second step in the value-at-risk estimation is the generation of the portfolio returns for each path. The returns of a portfolio can be calculated using the following formula

$$R_p = \omega^T \cdot g(r) \tag{32}$$

where ω is the vector of the weight of each assets, g is a vector of function that defines the payoff of

Algorithm 3: Importance Sampling Algorithm

Data: ρ, the correlation matrix of U
Data: X, the initial set of observations at time $t = 0$
Data: $\overline{\mu}$, the initial μ of the S-RMDN-GARCH at time $t = 0$
Data: σ^2, the initial hidden state of the S-RMDN-GARCH at time $t = 0$
Data: T, the number of time-step for which to sample and calculate the likelihood ratio q
Data: P, the number of samples to generate
Data: n, the number of assets in the portfolio
Result: Z

```
Ŷ ← [ ];
q ← [ ];
Z ← [ ];
for i ← 1 to n do
    Y_{0,i} ← X[i];
end
for t ← 1 to T + 1 do
    W ← [ ];
    M ← [ ];
    S ← [ ];
    for i ← 1 to n do
        U_i ← Φ(Z_i);
        η, μ, σ² ← S-RMDN-GARCH_i(X_{t−1,i}; μ̄_i, σ_i²);
        W_i ← η;
        M_i ← μ;
        S_i ← σ²;
        μ̄_i ← η^T · μ
    end
    Z_t ← randn(),    generate P samples from N(0, ρ);
    q_t ← f_V(Z_t;ρ,W,M,V) / f_Z(Z_t),    calculate the vector of weights;
end
```

each assets, and r is the vector of returns of each assets. In the case of a stock portfolio, the payoff of an assets is equal to its return. The vector g is relevant for assets with more complex payoff such as financial derivatives. In the case of a stock portfolio the returns of a portfolio calculated from the following formula

$$R_p = \omega^T \cdot r \tag{33}$$

The next step is the estimation of the value-at-risk from the computed portfolio returns. The value-at-risk can be estimated by finding the value x at which $\alpha = P(X > x)$. We present the algorithm for estimating the value-at-risk at over the time frame $(0, T)$ in algorithm 4.

Algorithm 4: Estimation of the value-at-risk

Data: Z, the samples from the model
Data: w, the weights of each samples from the importance sampling algorithm
Data: α, the level of the value-at-risk
Data: N_p, the number of paths generated
Result: V
$S \leftarrow sort(Z_t)$;
for $i \leftarrow 1$ **to** N_p **do**
    $p_v = \frac{1}{P}(Z_t \leq S_i))^T \cdot w_i$, Approximate probability of $P(Z_t \leq S_i)$;
    if $p_v >= \alpha$ **then**
        break, break when $P(Z_t \leq S_i) >= \alpha$;
end
$V \leftarrow S_i$, VaR is the value of S at index i;

4.6 Summary

In this chapter, we introduce the reader to the methodology proposed to forecast the value-at-risk. First, we present the current RMDN architecture for univariate forecasting, including the architectures of the RMDN and the RMDN-GARCH. We then introduce an improved recurrent mixture density network architecture, referred to as the Stable-RMDN-GARCH (S-RMDN-GARCH). This new architecture builds on the previously proposed model by enhancing the stability of the gradient of the model. Additionally, we propose the use of a novel pretraining method, the linear reduction pretraining, to improve the convergence of the S-RMDN-GARCH. The S-RMDN-GARCH is further extended to a multivariate model called the copula-S-RMDN-GARCH. We present an approach for estimating the parameters of the copula-S-RMDN-GARCH and for sampling the model. Furthermore, we propose an importance sampling method to enhance the efficiency of sampling from the copula-S-RMDN-GARCH. Finally, the algorithm for estimating the value-at-risk is introduced.

Chapter 5

Analysis

This chapter details the methods employed to establish the performance of the copula-S-RMDN-GARCH. We first present the data used to train and test the model. We then present the analysis of the performance of the pretraining method presented in Section 4.3. The following sections analyze the application of the univariate RMDN-GARCH model on the dataset. The same is done for the multivariate copula-ELU-RMDN.

5.1 Data and Parameters

In this section, we introduce the dataset used to train and test the model. We also present the parameters used for training and testing the model.

We selected the stocks comprised in the SP 500 index for testing the model. The SP 500 was chosen as the index comprised of the stocks of the 500 largest market capitalizations in the United States. Large market capitalization stocks tend to have a long price history, ensuring sufficient data to train the model and enabling it to perform well under various market conditions. Daily stock price data was collected for a period covering several years, starting from January 1, 2015, to December 31, 2019. More recent data collected for the years 2020 to 2023 were omitted from the dataset due to disruptions in the financial markets caused by the COVID-19 pandemic, making the out-of-sample period difficult to predict. Stocks whose data didn't cover the tested period were removed

from the dataset. The data was separated into two periods: the in-sample period and the out-of-sample period. The in-sample period starts on January 1, 2015, and ends on December 31, 2017. The in-sample period was used to train and validate linear reduction pretraining. The out-of-sample period starts on January 1, 2018, and ends on December 31, 2019. The out-of-sample period was used solely for testing the model and for comparison with the GARCH-based model. We collected the data from Yahoo Finance.

The S-RMDN-GARCH was trained for 300 epochs with a learning rate of 0.0001. In the test where linear reduction pretraining was performed, we pretrained the model for 20 epochs before training the model. The parameters of the S-RMDN-GARCH were initialized randomly, with the exception of the output nodes of the variance recurrent network and the bias of the network, which were initialized to 1. When pretraining is employed, the parameters of the non-linear nodes are initialized to zero. The models were trained on a Dell Precision T7510 workstation with 2 Intel Xeon E5-2630 v3 processors and 16GB of DDR3 ECC memory.

5.2 Analysis of the Performance of the Linear Reduction Pretraining Method

We present an analysis of the linear reduction pretraining method proposed in Section 4.3. To validate the efficacy of the pretraining method, we selected 10 different stocks randomly from the SP 500 components present in our dataset. For each stock, we trained 20 RMDN-GARCH models split equally between two groups: the models with pretraining (S-RMDN-GARCH) and the models without pretraining (RMDN-GARCH). Each model was initialized with 10 different seeds that were randomly selected. Each group uses the same seed. We then estimated the number of models that had converged during training for both groups.

The results from the test performed indicate that the linear reduction pretraining method is effective for improving the convergence of the model, as all the models in the group with pretraining converged to a satisfactory minimum. This contrasts with the groups of models trained without

linear reduction pretraining, which only converged for 31% of the models. The results of the pretraining test are presented in Table 5.1.

	Without pretraining		With pretraining	
Stock Ticker	Not Converged	Converged	Not Converged	Converged
AKAM	7	3	0	10
CBRE	6	4	0	10
EA	7	3	0	10
EMN	6	4	0	10
K	7	3	0	10
LOW	7	3	0	10
MKTX	7	3	0	10
NWSA	7	3	0	10
OKE	7	3	0	10
UAL	8	2	0	10
Total	69	31	0	100
Total%	69%	31%	0%	100%

Table 5.1: In-Sample Pretraining Test

The results presented in Table 5.2 show that using pretraining leads, on average, to a log-likelihood greater than that of the GARCH model, with the exception of the training runs performed on the EMN stock. However, after retraining the model on the EMN return series with a smaller learning rate of 0.00005, the model was able to converge well below the log-likelihood of the GARCH. This suggests that the choice of learning rate is also an important factor related to the convergence of the S-RMDN-GARCH. The tests previously presented allow us to conclude that

Stock Ticker	GARCH	With Pretraining	Without Pretraining
AKAM	-1999.45	-1875.84	-1898.40
CBRE	-1973.91	-1934.19	-1951.72
EA	-2038.76	-1995.17	-2078.30
EMN	-2025.97	-2027.82	-2033.02
K	-1782.75	-1695.07	-1700.97
LOW	-1993.17	-1906.67	-1914.93
MKTX	-2071.61	-2022.83	-2040.89
NWSA	-2041.73	-1968.67	-1989.55
OKE	-2054.26	-2006.29	-2022.70
UAL	-2349.71	-2303.32	-2335.57

Table 5.2: In-Sample Log-Likelihood

the linear reduction pretraining method improves the convergence of the S-RMDN-GARCH model during training. The pretraining method reduces the number of failed trainings but also allows the model to converge to a greater optimum, thereby improving the model's forecasting power.

5.3 Analysis of the Value-at-Risk Forecast

To establish the accuracy of the copula-SP-RMDN-GARCH, we present a comparative analysis of the value-at-risk forecasting accuracy of the Copula-GARCH model and the copula-S-RMDN-GARCH. This section comprises two parts: the methodology and the analysis of the results.

5.3.1 Testing Methodology

To estimate the accuracy of the model without introducing bias, we generate 50 equally weighted portfolios composed of 50 randomly selected stocks. The number of stocks per portfolio was selected as it is representative of the size of an average portfolio. The model wasn't developed for large portfolio with hundreds of assets. We then forecast the 1-day ahead value-at-risk for each portfolio for each date in the testing period. For each portfolio, we sample 25000 paths of 100 time steps. We compute the failure rate of the model and three different statistical tests on the value-at-risk estimates: the binomial distribution test, Kupiec's probability of failure test, and Christopherson's interval forecast test.

The failure rate measures the number of forecasts for which the value-at-risk was greater than the realized returns on that day. A perfectly accurate model should have a failure rate equal to the α of the value-at-risk. For each forecast, the presence of a failure is defined as

$$I_t = \begin{cases} 1, & \text{if } y_t \leq VaR_\alpha \\ 0, & \text{if } y_t > VaR_\alpha \end{cases} \tag{34}$$

The total number of failures is defined as $z = \sum_{t=1}^{T} I_t$. The total failure rate is defined as

$$\text{failure rate} = \frac{1}{T}\sum_{t=1}^{T} I_t \tag{35}$$

where T is the number of time steps (Christoffersen, 1998). To ensure that the estimates for the value-at-risk are consistent through the distribution of the stock returns we estimates the failure rates for 7 different value-at-risk levels. The failure rate was estimated for the following value-at-risk level: 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95.

To validate the accuracy of the value-at-risk estimated obtained with the copula-S-RMDN-GARCH model, we first perform the binomial distribution test. The test states that if the number of failures $\{I_t\}$ are independently and identically distributed the total number of failures should follow a Binomial distribution. Given a number of violation x, the test statistics is defined as

$$Z_{bin} = \frac{x - Np}{\sqrt{Np(1-p)}} \tag{36}$$

where N is the total number of observations and p is the confidence level (Zhang & Nadarajah, 2018). The test was performed with the following critical value: 0.01, 0.05, 0.1.

The second test we perform is the Kupiec proportion of failure test. This test differs from the Binomial test previously introduced as it measures how close the observed number of failures is to the probability implied by the value-at-risk level α (Kupiec, 1995). The test is a likelihood ratio distributed according to a chi-square variable with 1 degree of freedom. We define the likelihood ratio as

$$LR_{POF} = -2log\left(\frac{(1-p)^{n-x}p^x}{(1-\frac{x}{N})^{n-x}(\frac{x}{N})^x}\right) \tag{37}$$

where N is the number of observations and $p = 1 - \alpha$. Under the null hypothesis, the probability of a failure is equal to the VaR level α. We reject the value-at-risk model if the likelihood ratio is greater than a critical value p. The test was performed with the following critical values: 0.01, 0.05, 0.1.

The third test performed is Christofferson's interval forecast test. This test differs from Kupiec's proportion of failure test, as it integrate information about the interval time between value-at-risk breaches and their clustering (Christoffersen, 1998). To perform this test, we first consider a binary first-order Markov chain with transition matrix

$$\Pi_1 = \begin{bmatrix} 1-\pi_{01} & \pi_{01} \\ 1-\pi_{11} & \pi_{11} \end{bmatrix} \tag{38}$$

where the transition probabilities are defined as

$$\pi_{ij} = P(I_t = j | I_{t-1} = i). \tag{39}$$

Using the observed failures $\{I_t\}$, we can approximate the transition matrix Π_1 as

$$\hat{\Pi}_1 = \begin{bmatrix} 1-\pi_{01} = \frac{n_{00}}{n_{00}+n_{01}} & \pi_{01} = \frac{n_{01}}{n_{00}+n_{01}} \\ 1-\pi_{11} = \frac{n_{10}}{n_{10}+n_{11}} & \pi_{11} = \frac{n_{11}}{n_{10}+n_{11}} \end{bmatrix} \tag{40}$$

where n_{ij} is the number of observations where we observe a failure at time $t-1$ if $i = 1$ and a failure at time t if $j = 1$. For example, n_{01} corresponds to the number of observations for which $I_{t-1} = 0$ and $I_t = 1$. The likelihood ratio for the Christofferson's interval forecast test is defined as

$$LR_{CCI} = -2log\left(\frac{(1-\pi_2)^{n_{00}+n_{10}}\pi_2^{n_{01}+n_{11}}}{(1-\pi_0)^{n_{00}}\pi_0^{n_{01}}(1-\pi_1)^{n_{10}}\pi_1^{n_{11}}}\right) \tag{41}$$

where π_0 is the sample probability of having a failure on period t, given that no failure occurred on period $t-1$, π_1 is the probability of the model failing on period t, given that the model failed on $t-1$, and π_2 is the probability of having a failure at time t. We define π_{01} as

$$\pi_{01} = \frac{n_{01}}{n_{00}+n_{01}} \tag{42}$$

Similarly, π_1 can be defined as

$$\pi_1 = \frac{n_{11}}{n_{10} + n_{11}} \tag{43}$$

The definition of π_2 is defined as

$$\pi_2 = \frac{n_{01} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}} \tag{44}$$

Christofferson's interval forecast test statistic is distributed as a chi-square variable with 1 degree of freedom. We reject the value-at-risk model if the likelihood ratio is greater than a critical value. The test was performed with the following critical values: 0.01, 0.05, 0.1.

5.3.2 Results

We compare the results obtained during the estimation of the value-at-risk with the copula-S-RMDN-GARCH and the copula-AR-GARCH models. To compare the results, we compile descriptive statistics for the binomial, Kupiec, and Christofferson tests. The summary statistics for both models are presented in Table 5.3. For brevity, the names of the copula-S-RMDN-GARCH and the copula-AR-GARCH are abbreviated to RMDN and GARCH, respectively.

	copula-S-RMDN-GARCH			copula-AR-GARCH		
	Binomial	Kupiec	Christofferson	Binomial	Kupiec	Christofferson
μ	1.48	1.99	2.12	2.50	5.13	1.90
σ	0.50	1.31	0.86	0.80	2.86	1.56
Min	0.46	0.20	0.20	1.38	1.62	0.06
Max	3.21	7.54	3.36	4.13	11.71	5.78

Table 5.3: Descriptive statistics of the likelihood ratio for the Binomial, Kupiec, and Christofferson Tests

While the statistics indicate that both models fail on average for all tests, the copula-S-RMDN-GARCH seems to be a superior model, in part due to its lower average binomial, Kupiec, and Christofferson test statistics. However, when comparing the results for each portfolio individually in Table 5.4, the copula-AR-GARCH seems to outperform the copula-S-RMDN-GARCH for the

Christofferson test. This indicates that the copula-AR-GARCH outperforms the copula-S-RMDN-GARCH marginally for most portfolios, but that the copula-S-RMDN-GARCH outperforms the copula-AR-GARCH substantially in a few cases.

The results of the tests were evaluated at three different critical values: 0.01, 0.05 and 0.1. The

	RMDN > GARCH	RMDN > GARCH%	RMDN = GARCH	RMDN = GARCH%
Binomial	43	86%	4	8%
Kupiec	43	86%	4	8%
Christofferson	17	34%	4	8%

Table 5.4: Descriptive statistics for the Binomial, Kupiec, and Christofferson Test

number of tests that were accepted are reported in Table 5.5. We can observe that the number of

		copula-S-RMDN-GARCH		copula-AR-GARCH		Difference
		NB Accepted	% Accepted	NB Accepted	% Accepted	
Binomial	0.01	48	96%	25	50%	46%
	0.05	46	92%	16	32%	60%
	0.1	33	66%	9	18%	48%
Kupiec	0.01	49	98%	38	76%	22%
	0.05	46	92%	16	32%	60%
	0.1	33	66%	9	18%	48%
Christofferson	0.01	50	100%	50	100%	0%
	0.05	50	100%	45	90%	10%
	0.1	40	80%	40	80%	0%

Table 5.5: Summary of the Results of the Binomial, Kupiec and Christofferson Tests

tests being accepted for the copula-S-RMDN-GARCH is higher than the number of tests being accepted for the copula-AR-GARCH. This shows that the copula-S-RMDN-GARCH performs better than the copula-AR-GARCH. It also indicates that the copula-S-RMDN-GARCH can be used to forecast the value-at-risk effectively.

We also compare the failure rate of the model over multiple value-at-risk levels. This allows us to determine if the forecasts are accurate across the distribution of the portfolio's returns. To compare the copula-S-RMDN-GARCH and the copula-AR-GARCH failure rate, we compute the

absolute error from the failure rate. We define the absolute error of the failure rate as

$$\varphi = |\text{failure rate} - \alpha| \tag{45}$$

where α is the VaR level. We compare the absolute error rate of the copula-S-RMDN-GARCH with the absolute error rate off the copula-AR-GARCH in Table 5.6. We report the average, minimum, and maximum failure rate for both model as well as the number of copula-S-RMDN-GARCH, which have a lower error rate than the copula-AR-GARCH model. For brevity, the name of the copula-S-RMDN-GARCH and the copula-AR-GARCH are abbreviated to RMDN and GARCH, respectively. We can observe that the copula-S-RMDN-GARCH outperforms the copula-AR-GARCH model for most VaR levels alpha. However, the higher VaR levels see less improvement than the smaller VaR level.

	Average		Min		Max		RMDN > GARCH	RMDN > GARCH%
	GARCH	RMDN	GARCH	RMDN	GARCH	RMDN		
5%	0.0544	0.0322	0.03	0.01	0.09	0.07	43	86%
10%	0.07	0.0298	0.01	0	0.1	0.07	49	98%
25%	0.037	0.0144	0	0	0.08	0.05	42	84%
50%	0.022	0.018	0	0	0.05	0.07	28	56%
75%	0.0324	0.0248	0	0	0.07	0.07	29	58%
90%	0.0566	0.043	0.02	0	0.09	0.09	29	58%
95%	0.036	0.0228	0	0	0.06	0.05	34	68%

Table 5.6: Descriptive Statistics for Failure Rates for all Portfolios

The average execution time of the experiments, as well as the minimum and maximum execution times are reported for the copula-S-RMDN-GARCH and the copula-AR-GARCH in Table 5.7. Overall, the copula-AR-GARCH exhibits a faster execution time than the copula-S-RMDN-

	copula-S-RMDN-GARCH			copula-AR-GARCH		
	Copula	Sampling	VaR	Copula	Sampling	VaR
Average	83.26	87.92	86.96	1.12	122.14	0.05
Min	78.31	80.74	75.04	1.09	113.99	0.05
Max	94.16	104.11	94.70	1.18	127.80	0.05

Table 5.7: Execution Time for the Copula-SP-RMDN-GARCH and the Copula-AR-GARCH

GARCH. However, the sampling process of the copula-S-RMDN-GARCH is quicker than that of the copula-AR-GARCH, suggesting that the importance sampling developed is more efficient than the direct sampling algorithm of the copula-AR-GARCH. This efficiency is somewhat offset in the computation of the value-at-risk, as the importance sampling algorithm of the copula-S-RMDN-GARCH requires additional computations. Nevertheless, the estimation of the copula for the copula-AR-GARCH is considerably faster than the estimation of the copula for the copula-S-RMDN-GARCH. This indicates that the algorithm for estimating the copula's parameters in the copula-S-RMDN-GARCH is not as efficient as the current algorithm used in the copula-AR-GARCH.

5.4 Summary

In this analysis, we first show that the linear-reduction pretraining method is able to improve the convergence of the model, but also that the S-RMDN-GARCH outperforms the GARCH model in-sample. Our analysis showed that the copula-S-RMDN-GARCH generally outperforms the copula-AR-GARCH when used to estimate the value-at-risk. From our analysis, we can conclude that the copula-S-RMDN-GARCH is a viable alternative to the copula-AR-GARCH for estimating the model to forecast the value-at-risk.

Chapter 6

Conclusion

In this thesis, we presented a new methodology for forecasting the value-at-risk of a stock portfolio. We first discussed the background of the value-at-risk forecasting problem and then provided a summary of the literature available for forecasting time series. Next, we presented the approach that we developed to forecast multivariate time series and estimate the value-at-risk. Finally, we compared our model to the Copula-AR-GARCH model and presented the advantages and limitations of our model.

In this thesis, we focus on probabilistic forecasts as we propose an updated RMDN-GARCH architecture to improve the stability of the model, which we refer to as the S-RMDN-GARCH. We also introduce the linear-reduction pretraining method to further enhance the reliability of the S-RMDN-GARCH during training. Additionally, we propose the Copula-S-RMDN-GARCH model, which extends the S-RMDN-GARCH to multivariate settings. We first demonstrate that the S-RMDN-GARCH and linear-reduction pretraining are able to improve the convergence of the RMDN-GARCH. We conducted a comparative analysis of the Copula-S-RMDN-GARCH, which is compared with the Copula-AR-GARCH model. Our analysis shows that the Copula-S-RMDN-GARCH estimates of the value-at-risk are statistically superior to the estimates of the value-at-risk obtained from the Copula-AR-GARCH forecasts. This indicates that the Copula-S-RMDN-GARCH is a viable alternative to the GARCH model.

References

Adamkoa, P., Spuchľákováb, E., & Valáškováb, K. (2015, July). The history and ideas behind VaR.

Aielli, G. P. (2011, July 14). Dynamic conditional correlation: On properties and estimation. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1507743

Avellaneda, M., & Serur, J. A. (2020). *Hierarchical PCA and modeling asset correlations.*

Bishop, C. M. (1994, February). *Mixture density network* (Tech. Rep.). Aston University.

Blanchard, P., Higham, D. J., & Higham, N. J. (2019). *Accurate computation of the log-sum-exp and softmax functions.* arXiv. Retrieved from https://arxiv.org/abs/1909.03469

Bollerslev, T. (1986). Generalized autoregressive conditional heteroscedasticity.

Bollerslev, T. (1987). A conditionally heteroskedastic time series model for speculative prices and rates of return. *The Review of Economics and Statistics*, *69*(3), 542–547.

Brusaferri, A., Matteucci, M., Ramin, D., Spinelli, S., & Vitali, A. (2020). Probabilistic day-ahead energy price forecast by a mixture density recurrent neural network. In *2020 7th international conference on control, decision and information technologies (codit)* (Vol. 1, p. 523-528). doi: 10.1109/CoDIT49905.2020.9263898

Chen, J.-F., Chen, W.-L., Huang, C.-P., Huang, S.-H., & Chen, A.-P. (2016). Financial time-series data analysis using deep convolutional neural networks. In *2016 7th international conference on cloud computing and big data (ccbd)* (p. 87-92). doi: 10.1109/CCBD.2016.027

Chen, S., & He, H. (2018, October). Stock prediction using convolutional neural network. *IOP Conference Series: Materials Science and Engineering*, *435*(1), 012026. doi: 10.1088/1757-899X/435/1/012026

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning phrase representations using RNN encoder-decoder for statistical machine translation.*

Christoffersen, P. F. (1998). Evaluating interval forecasts. *International Economic Review*, *39*(4), 841–862.

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). *Fast and accurate deep network learning by exponential linear units (ELUs).* arXiv. Retrieved from https://arxiv.org/abs/1511.07289

Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, *1*(2), 223-236. doi: 10.1080/713665670

Dalbudak, Z. I., Atan, M., & Yilmaz, V. (2017). Comparison of value at risk methods: Application of ise 30. *Journal of Business, Economics and Finance*, *6*, 254-263.

Datta, S. (2022, November 2). *Modelling asset price volatility over time using LSTM.* (Available at SSRN: https://ssrn.com/abstract=4265186 or http://dx.doi.org/10.2139/ssrn.4265186)

Di-Giorgi, G., Salas, R., Avaria, R., Ubal, C., Rosas, H., & Torres, R. (2023, April). Volatility forecasting using deep recurrent neural networks as GARCH models. *Computational Statistics*. doi: 10.1007/s00180-023-01349-1

Ding, Z., Granger, C. W., & Engle, R. F. (1993). A long memory property of stock market returns and a new model. *Journal of Empirical Finance*, *1*(1), 83-106. doi: https://doi.org/10.1016/0927-5398(93)90006-D

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation.

Engle, R. F. (2002). Dynamic conditional correlation. *Journal of Business & Economic Statistics*, *20*(3), 339-350. doi: 10.1198/073500102288618487

Engle, R. F., Lilien, D. M., & Robins, R. P. (1987). Estimating time varying risk premia in the term structure: The arch-m model. *Econometrica*, *55*(2), 391–407.

Engle, R. F., & Ng, V. K. (1993). Measuring and testing the impact of news on volatility. *The Journal of Finance*, *48*(5), 1749–1778.

Fan, J., Fan, Y., & Lv, J. (2008). High dimensional covariance matrix estimation using a factor

model. *Journal of Econometrics*, *147*(1), 186-197. (Econometric modelling in finance and risk management: An overview) doi: https://doi.org/10.1016/j.jeconom.2008.09.017

Fernandez, C., & Steel, M. F. J. (1998). On bayesian modeling of fat tails and skewness. *Journal of the American Statistical Association*, *93*(441), 359–371.

Fjellström, C. (2022). *Long short-term memory neural network for financial time series.*

Föllmer, H., & Schied, A. (2016). *Stochastic Finance An Introduction in Discrete Time* (4th rev. ed. ed.). Berlin/Boston: De Gruyter. doi: 10.1515/9783110463453

Ge, W., Lalbakhsh, P., Isai, L., Lenskiy, A., & Suominen, H. (2022, January). Neural network–based financial volatility forecasting: A systematic review. *ACM Comput. Surv.*, *55*(1). doi: 10 .1145/3483596

Glasserman, P. (2004). *Monte Carlo methods in financial engineering*. New York: Springer. (Section: xiii, 596 pages : illustrations ; 25 cm.)

Glosten, L. R., Jagannathan, R., & Runkle, D. E. (1993). On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance*, *48*(5), 1779-1801. doi: https://doi.org/10.1111/j.1540-6261.1993.tb05128.x

Goldfeld, S. M., & Quandt, R. E. (1973). A Markov model for switching regressions. *Journal of Econometrics*, *1*(1), 3-15. doi: https://doi.org/10.1016/0304-4076(73)90002-X

Gray, S. F. (1996). Modeling the conditional distribution of interest rates as a regime-switching process. *Journal of Financial Economics*, *42*(1), 27–62.

Guillaumes, A. B. (2017). *Mixture density networks for distribution and uncertainty estimation.* Retrieved from https://upcommons.upc.edu/bitstream/handle/2117/100566/122527.pdf

Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the econometric society*, 357–384.

Hepp, T., Zierk, J., Rauh, M., Metzler, M., & Seitz, S. (2022, July). Mixture density networks for the indirect estimation of reference intervals. *BMC Bioinformatics*, *23*(1), 307.

Higgins, M. L., & Bera, A. K. (1992). A class of nonlinear ARCH models. *International Economic Review*, *33*(1), 137–158.

Hinich, M. J., & Patterson, D. M. (1985). Evidence of nonlinearity in daily stock returns. *Journal*

of Business & Economic Statistics, *3*(1), 69–77.

Hochreiter, S., & Schmidhuber, J. (1997, November). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735-1780. doi: 10.1162/neco.1997.9.8.1735

Jia, F., & Yang, B. (2021, February). Forecasting Volatility of Stock Index: Deep Learning Model with Likelihood-Based Loss Function. *Complexity*, *2021*, 5511802. (Publisher: Hindawi) doi: 10.1155/2021/5511802

Joseph, M. (2021, March 20). *Mixture density networks: Probabilistic regression for uncertainty estimation.* Retrieved from https://deep-and-shallow.com/2021/03/20/mixture-density-networks-probabilistic-regression-for-uncertainty-estimation/

Kamalov, F., Smail, L., & Gurrib, I. (2020, November). Stock price forecast with deep learning. In *2020 international conference on decision aid sciences and application (DASA).* IEEE. doi: 10.1109/dasa51403.2020.9317260

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization.* arXiv. Retrieved from https://arxiv.org/abs/1412.6980

Klaassen, F. (2002). *Improving GARCH volatility forecasts with regime-switching GARCH*. Springer.

Kumar, A., Alsadoon, A., Prasad, P. W. C., Abdullah, S., Rashid, T. A., Pham, D. T. H., & Nguyen, T. Q. V. (2021, November). Generative adversarial network (gan) and enhanced root mean square error (ermse): deep learning for stock price movement prediction. *Multimedia Tools and Applications*, *81*(3), 3995–4013. Retrieved from http://dx.doi.org/10.1007/s11042-021-11670-w doi: 10.1007/s11042-021-11670-w

Kupiec, P. H. (1995). Techniques for verifying the accuracy of risk measurement models. *The Journal of Derivatives*, *3*(2), 73-84. Retrieved from [] doi: 10.3905/jod.1995.407942

Ledoit, O., & Wolf, M. (2003, November). *Honey, i shrunk the sample covariance matrix.* Retrieved from http://www.ledoit.net/honey.pdf

Liu, S., Zhang, C., & Ma, J. (2017). CNN-LSTM neural network model for quantitative strategy analysis in stock markets. In (p. 198–206).

Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020, November). A CNN-LSTM-Based Model to

Forecast Stock Prices. *Complexity*, *2020*. (Publisher: Hindawi) doi: 10.1155/2020/6622927

Luo, R., Zhang, W., Xu, X., & Wang, J. (2018). *A neural stochastic volatility model.*

M. Bishop, C. (2006). *Pattern recognition and machine learning*.

Mostafa, F., & Dillon, T. (2008). Modelling volatility with mixture density networks. In *2008 ieee international conference on granular computing* (p. 501-505). doi: 10.1109/GRC.2008.4664673

Muhammad, T., Aftab, A. B., Ibrahim, M., Ahsan, M. M., Muhu, M. M., Khan, S. I., & Alam, M. S. (2023, April). Transformer-based deep learning model for stock price prediction: A case study on bangladesh stock market. *International Journal of Computational Intelligence and Applications*, *22*(03). doi: 10.1142/s146902682350013x

Nelson, D. B. (1991). Conditional heteroskedasticity in asset returns: A new approach. *Econometrica*, *59*(2), 347–370.

Nikolaev, N., Tino, P., & Smirnov, E. (2012). Time-dependent series variance learning with recurrent mixture density networks. *Neurocomputing*, *122*, 501-512.

Paranhos, L. (2021). *Predicting inflation with neural networks.* arXiv. Retrieved from https://arxiv.org/abs/2104.03757

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... Lerer, A. (2017). Automatic differentiation in PyTorch. Retrieved from https://arxiv.org/pdf/1912.01703.pdf

Rodikov, G., & Antulov-Fantulin, N. (2022). *Can LSTM outperform volatility-econometric models?*

Rosenblatt, M. (1952). Remarks on a multivariate transformation. *The Annals of Mathematical Statistics*, *23*(3), 470–472.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181-1191. doi: https://doi.org/10.1016/j.ijforecast.2019.07.001

Schittenkopf, C., Dorffner, G., & Dockner, E. J. (2000). Forecasting time-dependent conditional densities: a semi non-parametric neural network approach. *Journal of Forecasting*, *19*(4), 355-374. doi: 10.1002/1099-131X(200007)19:4⟨355::AID-FOR778⟩3.0.CO;2-Z

Schwert, G. W., & Seguin, P. J. (1989, May). *Heteroskedasticity in stock returns* (Working Paper No. 2956). National Bureau of Economic Research. doi: 10.3386/w2956

Sentana, E. (1995). Quadratic ARCH models. *The Review of Economic Studies*, *62*(4), 639–661.

Sklar, M. (1959). Fonctions de répartition à N dimensions et leurs marges. *Annales de l'ISUP*, *VIII*(3), 229-231.

Taleb, N. N., & Jorion, P. (1997, April). *The Jorion-Taleb debate.* Retrieved from https://web.archive.org/web/20170706074507/http://www.derivativesstrategy.com/magazine/archive/1997/0497fea2.asp

Tovar, W. (2020). *Deep learning based on generative adversarial and convolutional neural networks for financial time series predictions.*

Vidal, A., & Kristjanpoller, W. (2020, May 4). Gold volatility prediction using a CNN-LSTM approach. *Expert Systems with Applications*.

Vijh, S., Pandey, A. K., Vijh, G., & Kumar, S. (2021). Stock forecasting for time series data using convolutional neural network. In *2021 11th international conference on cloud computing, data science & engineering (confluence)* (p. 866-870). doi: 10.1109/Confluence51648.2021.9377128

Williams, R. J., & Zipser, D. (1989, June). A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, *1*(2), 270-280. doi: 10.1162/neco.1989.1.2.270

Williams, R. J., & Zipser, D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. In *Backpropagation: Theory, architectures, and applications* (p. 433–486). USA: L. Erlbaum Associates Inc.

Xue-Kun Song, P. (2000). Multivariate dispersion models generated from gaussian copula. *Scandinavian Journal of Statistics*, *27*(2), 305-320. doi: https://doi.org/10.1111/1467-9469.00191

Zakoian, J.-M. (1994). Threshold heteroskedastic models. *Journal of Economic Dynamics and Control*, *18*(5), 931-955. doi: https://doi.org/10.1016/0165-1889(94)90039-6

Zeng, Z., Kaur, R., Siddagangappa, S., Rahimi, S., Balch, T., & Veloso, M. (2023). *Financial time series forecasting using CNN and transformer.*

Zhang, Y., & Nadarajah, S. (2018). A review of backtesting for value at risk. *Communications in*

Statistics - Theory and Methods, *47*(15), 3616-3639. doi: 10.1080/03610926.2017.1361984

Zivot, E. (2008). *Practical issues in the analysis of univariate GARCH models* (Working Papers No. UWEC-2008-03-FC). University of Washington, Department of Economics.