

Volatility Calculation using Reinforcement Learning and Deep Learning.

A BS thesis end semester report submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Science
in
Data Science and Engineering
by

Shashank Singh
Roll No.: 19289

Under the Supervision of
Dr. P. B. Sujit



Department of Data Science and Engineering
Indian Institute of Science Education and Research Bhopal
Bhopal - 462066, India

April, 2023

Dedicated to my parents Mr. Dineshwar Kumar Singh,
Mrs. Ranjana Singh, my teachers Shri. Haridatt Seetha,
Shmt. Kanta Seetha and my guide Dr. P. B. Sujit.

Acknowledgements

Looking back at the last four years that I have spent at IISER-Bhopal I have made some amazing memories. Met some great friends and mentors but most importantly this college has helped me to find my true interests. Though I was not a very bright student with exceptional talent of making into Data Science and Engineering yet my professors have been extremely patient towards me. I would like to thank Dr. Kushal Shah and Dr. Tanmay Basu for making me believe in myself, I would like to thank to Dr. Jasabanta Patro for his patience and understanding. I would like to thank Dr. Vaibhav Kumar for guiding me as a fresh department representative of a new department, Dr Vinod Kurmi for his forgiveness and kindness. I would like to thank Dr. Mitradiip Bhattacharjee, Dr. Partibhan Srinivasan and Dr. Biswajit Patra for their many recommendation letters. I would like to thank Dr. Nabamita Banerjee for her constant support. I cannot thank Dr. P. B. Sujit enough for all his help, for letting me explore robotics under him for giving me the freedom to experiment with my career choices and make mistakes, for always forgiving my mistakes and giving me another chance. Meeting him truly was the best thing to happen after joining IISER-Bhopal.

I would like to thank my seniors Akarsh Ralhan, Madhur Mehta, Manav Mishra, Prithvi Poddar Rishabh Bhosle, Ranveer Singh, Aditya Ojha, Pallav Chanda, Chirag Adwani, Atharv Ambekar, Ayush Mathur, Sukrant Bhatt, Vishal Soni, Kasi V, Manavika Khanna, and S Gangotri for their awesome support and guidance.

I would like to thank Prahmarsh, Yuvraj, Raghav, Khush, Rahul, Devesh, Aaryan, Shubham Mulay, Amey, Anurag Yadav, Abhijeet Kumar, Sumit Nag, Deependra Singh, Bhalyo, Tanvi for always motivating me and helping me through depressing and challenging times.

I would like to thank Mrigank, Udit, Jaydev, Jayesh, Shivam Tiwari, Lokendra, Jayesh, Abhinav, Ali, Marut, Meet, Abhishek Khuriyal, Sumit Dangi, Sarthak, Sanjeet, Sai, Athrva, Shaunak and Jatin for teaching me so much. My distant support and all time friends Poornendu, Divyanshu, Pronoy and Anushk to constantly remind me of my worth after every failure and filling me up with motivation.

Lastly I would like to thank Dr. Sheereen Khan Bajpai for always listening to me and making campus life bearable and Mr. Arshi Ali for all his help through the placement process.

I am sorry if I have missed anyone's name but for someone who was filled with self doubt and guilt life would not have been possible without the support of my parents, professors, seniors and most importantly friends. If you have ever interacted with me and helped me I am truly indebted to you.

Thank you.

IISER Bhopal

Shashank Singh

November 4, 2024

Abstract

Volatility estimation is a crucial task in finance as it is used for risk management, option pricing, and portfolio optimization. Traditional methods for calculating volatility rely on statistical models that assume a specific distribution of the underlying asset's returns. However, these methods are often limited in their ability to capture the complex dynamics of financial markets.

In recent years, there has been a growing interest in using machine learning techniques, such as Reinforcement Learning (RL) and Deep Learning (DL), to model and predict financial volatility. RL algorithms can learn directly from market data and optimize a policy to maximize a specific reward, while DL algorithms can learn complex patterns and relationships from large datasets.

Results show that RL and DL can provide accurate and robust volatility estimates, outperforming traditional methods in certain cases. However, there are still challenges and limitations that need to be addressed, such as data quality, model interpretability, and generalization to new market conditions. We conclude that RL and DL hold promise as powerful tools for volatility prediction in finance, but further research is needed to fully realize their potential.

Contents

1	Introduction	1
2	Background	3
3	Research Gaps	5
4	Objectives	7
5	Methodology	9
5.1	Mathematical Formulation	9
5.2	Getting the Data	10
5.3	Deep Learning	12
5.4	Reinforcement Learning	15
6	Results and Discussion	17
7	Conclusion	21
8	Work Plan	23
9	References	25

List of Tables

Chapter 1

Introduction

Volatility is an important measure of risk and uncertainty in the financial markets. It plays a crucial role in the pricing and hedging of financial instruments such as options and futures contracts. Over the years, various methods have been proposed to estimate volatility, including traditional econometric models and statistical techniques. However, with the increasing availability of data and computational resources, machine learning techniques have emerged as a promising alternative for volatility prediction.

Among these techniques, Reinforcement Learning (RL) and Deep Learning (DL) have gained significant attention in recent years. RL is a subfield of machine learning that focuses on training agents to make decisions based on feedback received from the environment. In the context of volatility prediction, RL can be used to learn optimal trading strategies that take into account the uncertainty and risk associated with different market conditions. On the other hand, DL, which is a type of artificial neural network, has been shown to be effective in capturing complex patterns and dependencies in financial data.

We aim to develop a volatility calculator that uses these techniques to generate accurate and timely estimates of volatility. We will evaluate the performance of our approach using real-world financial data and compare it with existing methods. Overall, our research contributes to the growing body of literature on machine learning in finance and has important implications for risk management and investment decision-making.

Chapter 2

Background

The stock market is a complex and dynamic system that is affected by numerous factors, including economic indicators, geopolitical events, and public sentiment. Volatility, or the degree of variation of prices over time, is a key aspect of the stock market and plays a crucial role in decision making for investors and traders.

Traditional methods for calculating volatility involve using statistical models that assume a certain distribution of returns. However, these models may not accurately capture the non-linear and non-stationary nature of the stock market.

Reinforcement learning and deep learning are emerging fields in artificial intelligence that have shown promise in addressing these challenges. Reinforcement learning is a branch of machine learning that involves an agent learning to make decisions in an environment through trial and error. Deep learning is a subset of machine learning that uses artificial neural networks to model complex patterns in data.

By using reinforcement learning and deep learning techniques, it may be possible to develop more accurate and robust models for calculating volatility in the stock market. These models could take into account a wider range of factors and be better able to adapt to changing market conditions, leading to more informed decision making and potentially higher returns for investors and traders.

Chapter 3

Research Gaps

Traditional approaches to volatility calculation in finance often rely on statistical models, such as GARCH models or stochastic volatility models, which are based on assumptions about the underlying distribution of stock prices and may not always capture the complex and non-linear relationships in financial data. Reinforcement learning and deep learning approaches, on the other hand, can learn directly from the data without making explicit assumptions about the underlying distribution or relationships in the data. This can lead to more accurate and flexible models for volatility prediction, especially in highly dynamic and unpredictable market conditions. Additionally, these methods can incorporate feedback and adjust their predictions in real-time, making them well-suited for decision-making in financial markets.

Chapter 4

Objectives

The primary objective of volatility calculation in finance is to estimate the degree of uncertainty or risk associated with an investment. This can help investors make more informed decisions about their investments and can also help financial institutions better manage their risk exposure.

Traditional approaches to volatility calculation, such as the use of statistical models, have limitations in accurately capturing the complex dynamics of financial markets. Reinforcement learning and deep learning techniques offer an alternative approach that can better capture these dynamics, making them a promising tool for volatility prediction and risk management.

Reinforcement learning allows an agent to learn and make decisions based on feedback from its environment. In the context of volatility calculation, this can involve learning the optimal strategy for trading and hedging in response to changing market conditions. Deep learning, on the other hand, can be used to extract patterns and features from large amounts of financial data, allowing for more accurate predictions of future market trends.

By combining reinforcement learning and deep learning techniques, it is possible to create a more robust and accurate volatility calculator that can better capture the complex dynamics of financial markets. This can ultimately help investors and financial institutions better manage risk and make more informed investment decisions.

Chapter 5

Methodology

5.1 Mathematical Formulation

Volatility over a given time frame can be calculated using the given formulas.

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$




σ = Standard Deviation

N = Number of Trading Days

x_i = Daily Stock Price

μ = Mean of Prices

Volatility Formula


$$\text{Annualized Volatility} = \text{Standard Deviation} \times \sqrt{252}$$


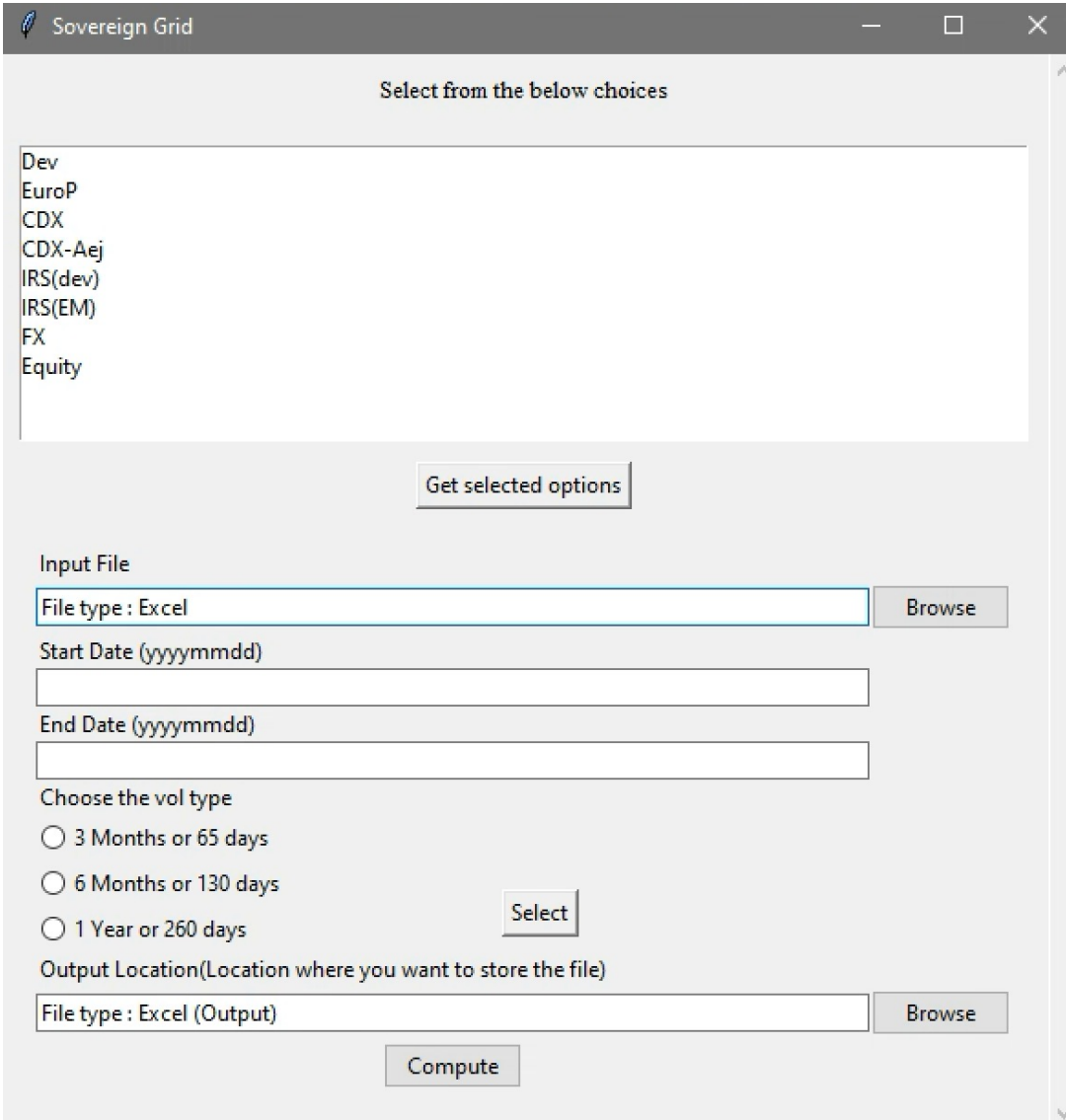
Volatility

$$Volatility_t = volatility_{annual} \times \sqrt{t}$$

$$Volatility_{weekly} = volatility_{annual} \times \sqrt{\frac{1}{52}}$$

$$Volatility_{daily} = volatility_{annual} \times \sqrt{\frac{1}{252}}$$

5.2 Getting the Data



The screenshot shows a web application window titled "Sovereign Grid". Inside, there's a section titled "Select from the below choices" with a list of data sources: Dev, EuroP, CDX, CDX-Aej, IRS(dev), IRS(EM), FX, and Equity. Below this list is a "Get selected options" button. Further down, there's an "Input File" section with a text box containing "File type : Excel" and a "Browse" button. Below that are two empty text boxes for "Start Date (yyyyymmdd)" and "End Date (yyyyymmdd)". Then, there's a "Choose the vol type" section with three radio button options: "3 Months or 65 days", "6 Months or 130 days", and "1 Year or 260 days". A "Select" button is positioned to the right of these options. Below the radio buttons is an "Output Location(Location where you want to store the file)" section with a text box containing "File type : Excel (Output)" and a "Browse" button. At the bottom center is a "Compute" button.

As the firm has a lot of data across all offices a big challenge was to access the data and use the specific financial data that we need to calculate volatility of.

The pool of Financial assets that were considering are as follows:-

- Dev
- EuroP
- CDX
- CDX-Aej
- IRS(Dev)
- IRS(EM)

- FX
- Equity

Another challenge of extracting data was that volatility over 3 different time intervals were required.

The third challenge was that the data was in the excel format so a need for central user-interface was felt that will fetch the required data and push the data to various functions and models.

To solve the issue we made a Graphical User Interface using the tkinter library.

The screenshot shows a graphical user interface titled "Sovereign Grid". At the top, it says "Select from the below choices". Below this is a list of options: Dev, EuroP, CDX, CDX-Aej, IRS(dev), IRS(EM), FX, and Equity. The options EuroP, CDX-Aej, and IRS(EM) are highlighted in blue. Below the list is a button labeled "Get selected options".

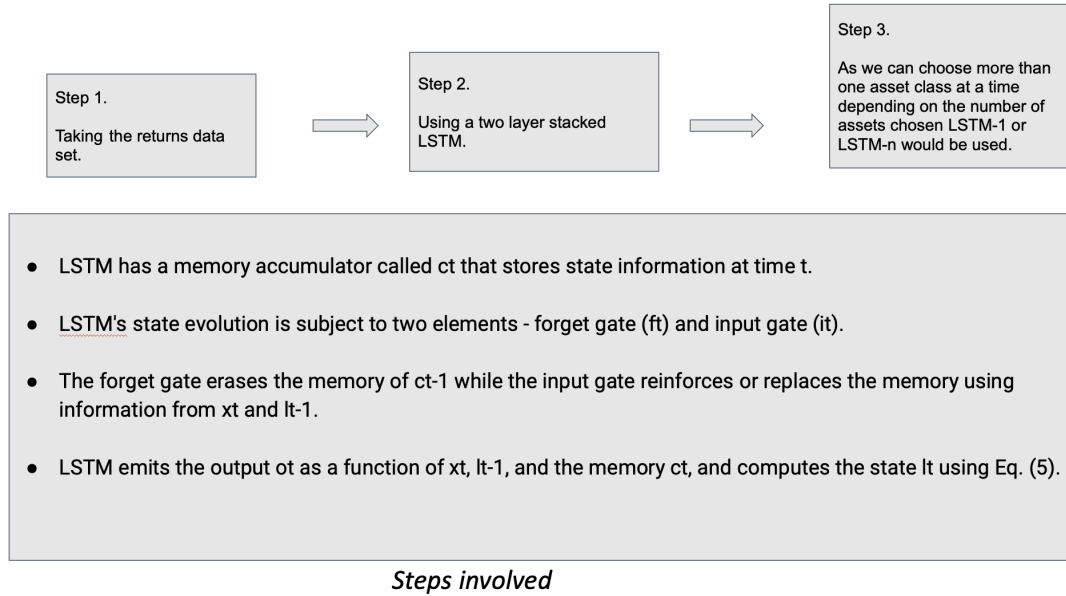
Below the button, there are several input fields and buttons:

- Input File:** A text box containing "EMEAList.xlsm" and a "Browse" button.
- Start Date (yyyymmdd):** A text box containing "2023-03-31".
- End Date (yyyymmdd):** A text box containing "2012-02-24".
- Choose the vol type:** Three radio buttons: "3 Months or 65 days" (selected), "6 Months or 130 days", and "1 Year or 260 days". A "Select" button is next to them.
- Output Location(Location where you want to store the file):** A text box containing "/Desktop" and a "Browse" button.
- A "Compute" button is at the bottom.

The GUI lets the user input multiple of the above choices at once. Then as there are multiple sources of such data we need to select the input location for our data (in excel format). Then we are required to choose the time frame over which we need to calculate the volatility. After this a final destination is chosen where the volatility calculator generates separate excel files for each asset with their volatility and relative volatility.

Now since all the inputs necessary for our work and model training has been extracted we are ready to use it for volatility calculation.

5.3 Deep Learning



Steps involved

$$i_t = \sigma(W_{xi}x_t + W_{li}l_{t-1} + W_{ci}c_{t-1} + b_i), \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{lf}l_{t-1} + W_{cf}c_{t-1} + b_f), \quad (2)$$

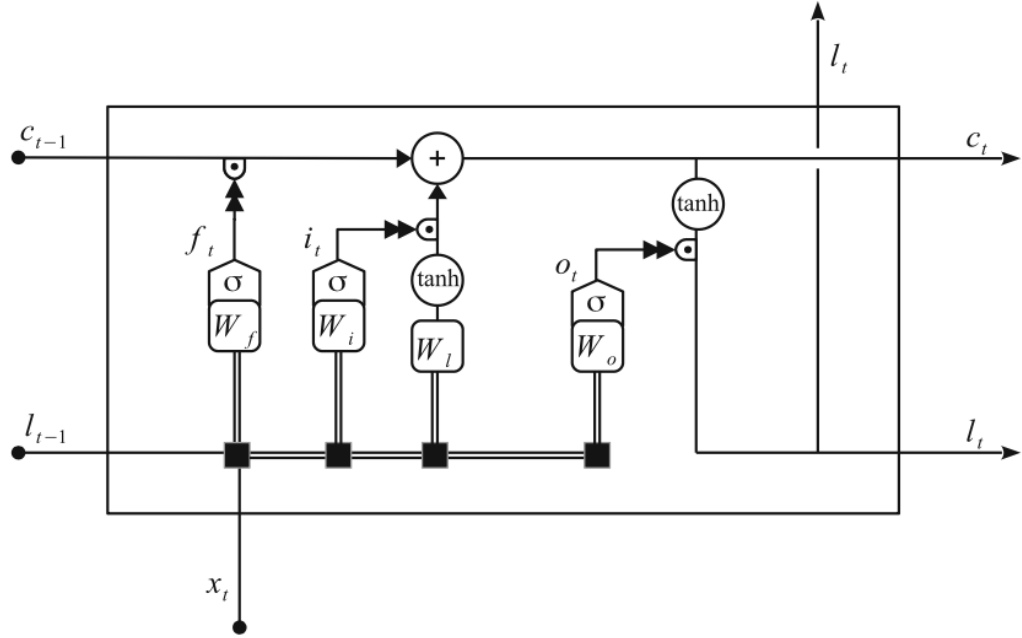
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{lc}l_{t-1} + b_c), \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{lo}l_{t-1} + W_{co}c_t + b_o), \quad (4)$$

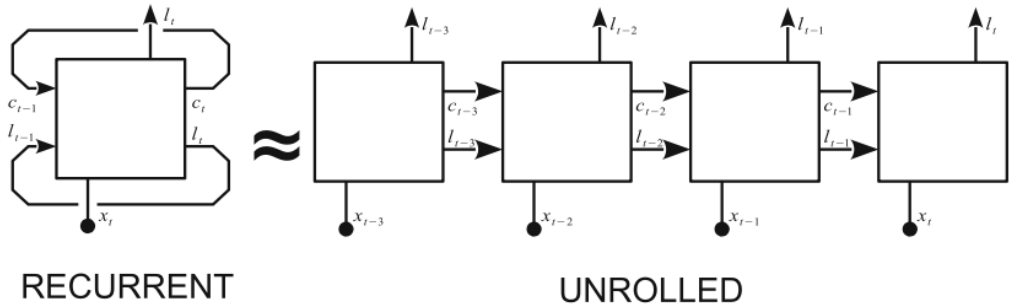
$$l_t = o_t \tanh(c_t). \quad (5)$$

The core of the LSTM is represented by the ct which acts as a memory accumulator of the state information at time t . The state evolves according to Eq. (3), subject to two elements—the “forget gate” and the “input gate”, represented at time t by the variables ft and it , respectively. The role of ft is to erase the memory $ct-1$ according to the current input xt (comprising open-close return

and volatility), the state l_t and the memory c_t (Eq. (2)). The forget gate is counterbalanced by the input gate (Eq. (1)) that making use of the same information has instead the role of reinforcing or replacing the memory by activating a combination x_t and l_t (Eq. (3)). These last functions, as those governing the activation of f_t and i_t are learned as single-layer perceptrons using the logistic function (Eq. (1) and Eq. (2)), or the tanh function (Eq. (3)) as activation, where b_i , b_f and b_c are the respective biases. Once the memory is recomputed at time t , the LSTM emits the output o_t as a function of x_t , l_t and the memory c_t (Eq. (4)). This latter function is also learned as a single-layer perceptron and finally, the LSTM computes the state l_t as given by Eq. (5). Figure 1b shows how a sequence is propagated through the LSTM.



(a) LSTM cell

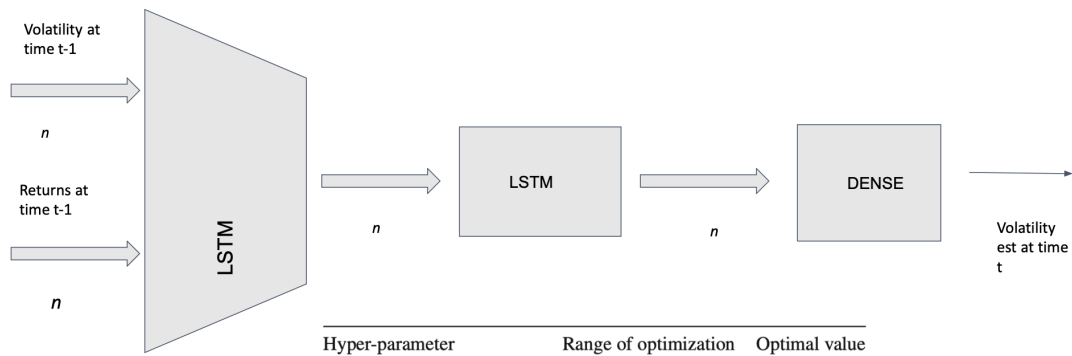


(b) unfolding of LSTM

The main advantage of this architecture is that the memory c_t is refreshed under the control of gates so that the gradient is limited to the last stage and

prevented from vanishing too quickly. This latter issue is a critical one and a well-known limitation of RNN based on older architectures, such as the Elman’s and Jordan’s reference models.

Because the LSTM learning function can be decomposed into multiple intermediate steps, LSTMs can be “stacked” such that the information produced by one LSTM step becomes the input to another. This “stacked” architecture has been applied to many real-world sequence modeling problems.



Hyper-parameter	Range of optimization	Optimal value
Dropout	[0, 0.6] every 0.1	0.2
Number of training epochs on pre-training data	[100, 400] every 50	300
Number of training epochs on rolling window	[10, 30] every 5	20
Look-back	[5, 100] every 5	20
Loss function	MSE and QLIKE	QLIKE

LSTM-1 has a univariate architecture (one model independently trained for each asset) so takes as input only one asset at a time (open-close return and volatility for a chosen past window $(rt_k, \dots, rt_1, vt_k, \dots, vt_1)$) and produces as output the one-step-ahead volatility (vt).

LSTM-n has a multivariate architecture, where a single model is trained using all the assets. This version takes as input the daily returns and volatilities for a given past window $(rt_{ik}, \dots, rt_{i1}, vt_{ik}, \dots, vt_{i1})$, $i = 1, \dots, n$ and outputs the n one-step-ahead volatility (vt_i , $i = 1, \dots, n$).

For each volatility duration the model is pre-trained and updated on data available starting 2008-01-01. With a look back window of 20 days (on average the number of trading days in a month).

5.4 Reinforcement Learning

Environment - All the assets available in the above 8 choices.

Goal is to map the current state(pf) to an optimal future state (p0) that maximizes the expected reward.

Based on the input from the user-interface the RL algorithm receives observations of the current state and then chooses action based on a policy that has been learned during the training.

The policy is updated iteratively based on the reward signals provided by the environment.

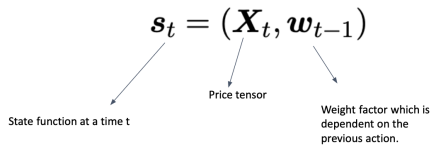


Figure 5.1: State Function

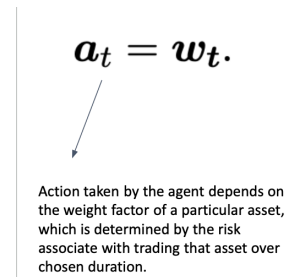


Figure 5.2: Action

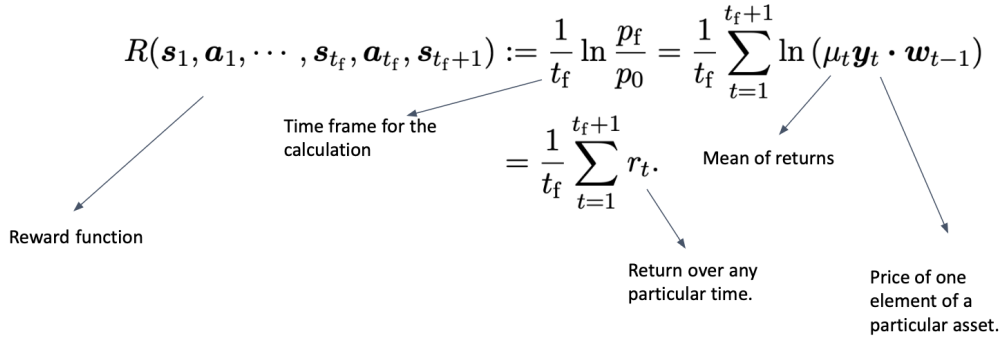


Figure 5.3: Reward

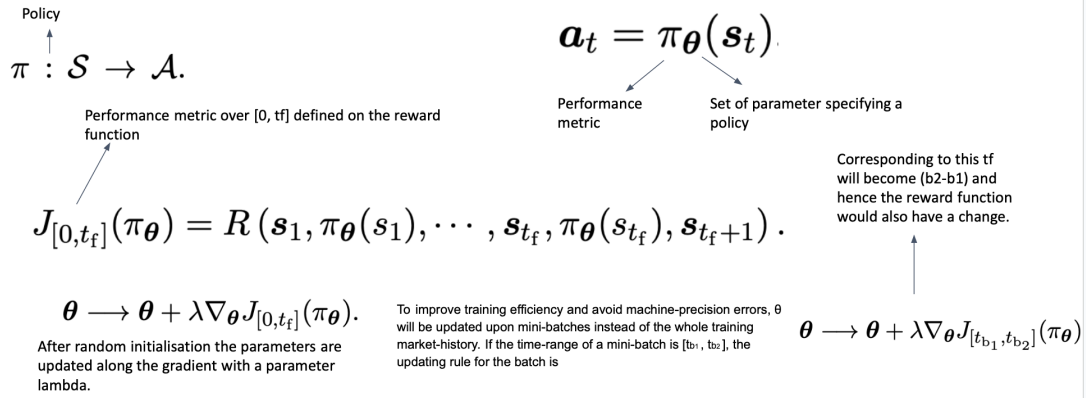


Figure 5.4: Steps in RL

Chapter 6

Results and Discussion

For analysing how good our approach is as compared the traditional models like GARCH models we have used two performance matrices.

We have considered both online and offline evaluations. In the online evaluation case, two alternative loss functions have been used to train the LSTM models: the widely accepted Mean Squared Error (MSE) for regression and forecasting tasks; and the QLIKE function, particularly suitable for volatility forecasting (Patton 2011). For the offline evaluation case, we have considered a test data set (not used for training) for out-of-sample evaluation using MSE, QLIKE

1. MSE (Mean Squared Error) -

Measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate.

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (\hat{Y}_i - Y_i)^2,$$

2. QLIKE (Quasi-Likelihood Information Criterion) -

Is a goodness-of-fit measure used in statistical modeling. It is a modification of the Akaike Information Criterion (AIC) that is commonly used to evaluate the relative quality of statistical models for a given set of data. QLIKE is used for models that have been estimated using quasi-likelihood methods, which are a class of statistical methods used to analyze data that does not satisfy the normality assumptions of traditional methods. In finance, QLIKE is commonly used to evaluate the performance of volatility models, such as GARCH models. A lower QLIKE value indicates a better fit of the model to the data.

$$\text{QLIKE} = \frac{1}{N} \sum_{i=1}^N (\log(\hat{Y}_i) + \frac{Y_i}{\hat{Y}_i}).$$

Here \hat{Y} represents the predicted volatility.

Final Results after running the model

Asset Classes	MSE	QLIKE	MSE	QLIKE
	(LSTM-1)	(LSTM-1)	(GARCH)	(GARCH)
CDX		3.1	1.735	4.6
				2.9
	(LSTM-3, n=3)	(LSTM-3, n=3)	(GARCH)	(GARCH)
CDX-Aej, EuroP, Dev		2.46	1.177	3.53
				1.91

As it can be seen the Deep Learning outperforms the Reinforcement Learning model by a huge margin.

For evaluating the Reinforcement Learning model we use MSE and MDD

3. MDD (Maximum Drawdown) -

It is a risk metric that measures the largest single drop from peak to bottom in the value of a portfolio over a particular time period. MDD is important because it gives investors an idea of the largest loss they might suffer during a

certain investment period. It is calculated by taking the ratio of the difference between the peak value and the trough value of the portfolio divided by the peak value. **MDD is the biggest loss from a peak to a trough, and mathematically**

$$D = \max_{\substack{\tau > t \\ t}} \frac{p_t - p_\tau}{p_t}$$

Since upward volatility contributes to positive returns and downward volatility to negative returns in order to highlight this MDD (Maximum Drawdown) is considered instead of QLIKE.

Final Results after running the model

Asset Classes	MSE	MDD
	RL	RL
CDX	8.1	0.224
	RL	RL
CDX-Aej, EuroP, Dev	29.46	0.477

Chapter 7

Conclusion

In conclusion, Reinforcement Learning (RL) has shown promise in developing trading algorithms that can learn from experience and adapt to changing market conditions. Compared to traditional trading models, RL-based models can potentially achieve higher levels of performance and adaptability.

However, it is important to note that developing RL-based trading algorithms can be complex and requires a deep understanding of both machine learning and finance. Additionally, the performance of RL-based models can be highly dependent on the quality of the data used for training and the parameters chosen during model development. Therefore, careful analysis and evaluation of the algorithm's performance on historical data and in live trading environments are necessary before deploying an RL-based trading algorithm.

Overall, RL-based trading algorithms have the potential to revolutionize the financial industry, but their development and deployment require a thorough understanding of both machine learning and finance, and careful analysis and evaluation of the algorithm's performance.

Chapter 8

Work Plan

1. Gather and preprocess data: Collect historical stock market data, including daily stock prices and other relevant financial indicators.
2. Preprocess the data to prepare it for input into the models. Choose and train a Reinforcement Learning model: Select a suitable Reinforcement Learning algorithm to use for predicting volatility.
3. Train the model using the preprocessed data, adjusting the hyperparameters as needed to optimize performance.
4. Choose and train a Deep Learning model: Select a suitable Deep Learning architecture, such as a convolutional neural network or a long short-term memory network. Train the model using the preprocessed data, adjusting the hyperparameters as needed to optimize performance.
5. Evaluate model performance: Evaluate the performance of the Reinforcement Learning and Deep Learning models using various metrics, such as mean squared error or root mean squared error.
6. Compare results to traditional methods: Compare the performance of the Reinforcement Learning and Deep Learning models to traditional methods for calculating volatility, such as the GARCH model or other statistical approaches.

7. **Refine models:** Refine the models as needed to improve their accuracy and performance, using techniques such as transfer learning or ensemble methods.
8. **Test models on new data:** Test the final models on new, unseen data to confirm their performance and accuracy.
9. **Implement models in a practical setting:** Implement the final models in a practical setting, such as a trading platform or financial institution, to help predict stock market volatility and inform investment decisions.

Chapter 9

References

- [1] Hongyang Yang, Yifan Cao, Fengmin Xu, and Hanqing Qiu (2018), *"Deep Reinforcement Learning for Volatility Trading."*

- [2] Stefan Zohren, Bryan Lim, Stephen Roberts, *"Enhancing Time Series Momentum Strategies Using Deep Neural Networks."*

- [3] Xiang Yu, Yan Liu, Zihao Wang, and Rui Yang, *"Deep Reinforcement Learning for Portfolio Optimization."*

- [4] Zihao Zhang, Stefan Zohren, Stephen Roberts, *"Deep Reinforcement Learning for Trading."*

- [5] Taylan Kabbani, Ekrem Duman, *"Deep Reinforcement Learning Approach for Trading Automation in The Stock Market."*

- [6] Frensi Zejnullahu, Maurice Moser, Joerg Osterrieder, *"Applications of Reinforcement Learning in Finance – Trading with a Double Deep Q-Network."*

- [7] Jinjing Liang, Xiaolin Wang, and Junbo Zhang, "*Reinforcement Learning for Financial Portfolio Management: A Deep Neural Network Approach.*"
- [8] Kamil Mizgier, Paweł Lachowicz, and Łukasz Maślanka, "*Reinforcement Learning for Trading and Risk Management.*"
- [9] Benjamin Löwe, Stefan Feuerriegel, and Dirk Neumann, "*Deep Reinforcement Learning for Volatility Prediction.*"
- [10] Shuaiqiang Liu, Cornelis W. Oosterlee, and Sander M.Bohte, "*Pricing options and computing implied volatilities using neural networks.*"
- [11] Jay Cao, Jacky Chen, John Hull, Zissis Poulos, "*Deep Learning for exotic option evaluation.*"
- [12] Shengli Chen, Zili Zhang, "*Forecasting Implied Volatility Smile Surface via Deep Learning and Attention Mechanism1.*"