HIGH SCHOOL EDITION
Journal of Student Research
Volume 11 Issue 3 (2022)

Analyzing Volatility Forecasting Capabilities of Neural Network Enhanced ARCH Models

Priyansh Singh[1] and Erin O'Rourke[#]

[1]Syosset High School, Syosset, NY, USA
[#]Advisor

ABSTRACT

Examines the capabilities of Autoregressive-Conditional-Heteroskedasticity (ARCH) family models (with Artificial Neural Networks) to predict volatility of thirty equities from a five-year fiscal-period. The models underwent the maximization of its parameters through Hessian matrices and were used to predict volatility by maximizing the log-likelihood function. Trained Long-Short-Term-Memory models using Neural-Net-Enhanced-ARCH algorithms and calculated the Root-Mean-Square-Error. Found the RMSE value of the traditional ARCH/GARCH models as 1.1695 as opposed to the algorithm's 0.8763.

Motivation

About eight months ago, I had watched a video segment produced by Yahoo Finance which summarized the happenings of stock market for the past week when, suddenly, the market crashed. The stock prices for Gamestop and AMC Entertainment Holdings, Inc had surged, an increase measuring 104% and 301%, respectively. Upon further investigation regarding the anomaly in the news, I discovered that the New York Stock Exchange had reached insurmountable volatility rates, causing many investors to gain high returns on their investments. This event proved just how volatile and unpredictable a market can be, and how seemingly non-influential variables can change the fate of a stock share. A group of investors pooling funds into a stock and "holding" their shares were able to turn the fortune of other investors and firms overnight and cause the stock market to have one its highest volatility rates.

I was motivated to research mathematical finance and conduct a study that was centered around the volatility of stocks using GARCH family models and AI. After reading a plethora of articles on the event, I found that none of the articles contained any scientific evidence of the oddity, rather just empirical information of share prices. Furthermore, the articles were "projection-centric", discussing the direction of the stock prices instead of addressing the sophisticated evidence of the volatility change in them. In this paper, I present further research on creating and analyzing volatility capabilities of ARCH models using Artificial Neural Networks to further help investors and federal regulation companies forecast volatility more accurately.

Introduction

The notion of prudently forecasting volatility in the stock market has been one that has been widely accepted by those within the realm of financial statistics. Per its definition, volatility, or variance can be defined as the rate at which a stock's price rises or falls over a given period, in a financial data set. Intuitively, more stock price volatility refers to higher risk posed for investment, but they also allow an investor to predict future variations in share price, with almost all financial applications of volatility models including anticipating future returns. To predict the magnitudes of returns, a volatility model, a forecast quantile, or even the whole density

function, of the prices can be employed. With the advent of Artificial Intelligence (AI), specifically Machine Learning, financial estimations can be made more accurately. Artificial Neural Networks, or computational algorithms used for pattern detection, are an integral part of forecasting financial derivative prices and can also be employed to project volatility within the stock market. In this paper, forecasting volatility using accepted theories, G(ARCH) families, as well as Artificial Neural Networks are used to predict volatility of thirty Big-Cap stocks from the S&P 500 stock index.

First put forth by Tim Bollerslev and S.J Taylor in 1986, the Autoregressive Conditional Heteroskedasticity, or the ARCH model, is a class of statistical models that are used to estimate and project the volatility of financial integrants such as options, stocks, bonds, etc. ARCH-type models are also a part of the stochastic volatility model family which are models in which the variance of a stochastic process is randomly distributed. In "A Practical Guide to Volatility Forecasting Through Calm and Storm" by Robert Engle, Bryan Kelly, and Christian Brownlees, every model within the ARCH family is used to forecast the volatility based on data from the years 1990 to 2008. The study examined the way design elements are implemented into a real-time volatility forecasting approach, such as the model type, the quantity of data to utilize in the estimate, the frequency of estimation updates, and the use of heavy-tailed likelihoods for volatility forecasting. The investigation progresses to the experiment stage where a wide variety of domestic and foreign equity indexes and exchange rates are used to test the process in different environments. These environments included the volatility spike in the third and fourth quarters of 2008 (2008 Financial Crisis) to examine if the model works in turbulent times. The findings were that the threshold GARCH model, the simplest asymmetric generalized autoregressive conditional heteroskedasticity (GARCH) specification, is the best forecaster across all asset classes. Furthermore, the researchers concluded that the GARCH family functions performs with the most precision over a long period of time.

Among the financial derivatives that act as agents of our markets, options are the most are one of the widely used derivatives for forecasting volatility from historical prices. They are considered to be the most helpful in predicting volatility because of their high susceptibility to volatility changes, which can indicate high potential returns. In particular, the implied volatility of an option, which indicates the stock's predicted volatility throughout the option's life, is used by many researchers for analysis. In "Using Neural Networks to Forecast the S&P 100 Implied Volatility" by Linda Salchenberger and Mary Malliaris, the authors present a neural network that was able to successfully predict the most often utilized volatility (implied) by traders. In every scenario studied within the study, neural networks, an algorithm proved to efficiently describe inconsistent interactions, proved as one viable strategy to forecast local options' volatility and hence may be utilized to produce reliable forecasts. The historical volatility was calculated for each trading day in 1992 using a size 30 Index price sample. The authors utilized the Black-Scholes model to generate implied volatility for three different contracts to find the closest at-the-money: current month, one month out, and two months away, with 250 observations in each series of volatility in the experiment. The results were that the portion of successful projections using neural networks had a value of 0.794. With a significance threshold of 0.0001, the future implied volatility and neural network prediction had a correlation of 0.8535, indicating a moderately strong relationship between the two mediums of forecasting.

Another method for foretelling volatility in financial markets includes using Machine Learning in an out-of-sample forecast. In "A Machine Learning Approach to Volatility Forecasting" by Bezirgen Veliyev, Mathias Siggaard, and Kim Christensen, the researchers aimed to find whether a large count of inconsistent variables display conditional heteroskedasticity and serial correlation and prove the accuracy of machine learning (ML) within the experiment. The authors of this research studied the out-of-sample performance of a variety of machine learning approaches for the projection of volatility, including tree-based algorithms such as gradient boosting and random forest, as well as neural networks. The results of this method were then compared to the results yielded by the HAR (traditional volatility model). This paper's findings were organized into five categories. First, the HAR model is compared against off-the-shelf implementations of several ML approaches for

out-of-sample predicting accuracy. Second, if regularization was used to account for over-fitting, significant statistics regarding volatility projection may recover using ML. When more factors are added to the HAR model, the forecast accuracy improves. Third, the authors discovered a nonlinear interaction between variables by looking at the structure of ML approaches. Fourth, the authors use accumulated local effect (ALE) plots to calculate variable relevance. Fifth, the authors look at how machine learning approaches react to pure noise factors. The two new mediums of forecasting were discovered to provide greatly enhanced predictions when compared to HAR. To confuse or distort the signal, the authors added a plethora of noise variables to the information set to check robustness of the models. In contrast to linear regression, the majority of ML approaches were somewhat resistant to this impact. Finally, the authors use ALE plots to examine the data's underlying structure to pinpoint the primary factors that contribute to improved prediction.

The intricacy of predicting volatility is what contributed to its many benefits for investors. With the use of past prices of financial instruments, the volatility of these rates can be determined effectively and accurately. With the improvements being made to the traditional volatility indicator models, namely ARCH models, these results continue to become more accurate. Moreover, with enhancements to Machine Learning and AI, these processes can be done quickly and efficiently.

Data

Thirty equities from the S&P 500 will be selected for the analysis based on their relative weights. The equities will be chosen from a wide range of sectors ranging from Informational Technology to Energy. The stock index for the shares will be computed using a Capitalization Weighted Index, which involves smaller components that are weighted with the relative total market capitalization of the stock. The daily closing prices were collected from the past five years using online platforms, namely MarketWatch. The following formula will compute the price of each stock at the end of the five-year fiscal period using log-returns

$$P_t = P_0(1+r) = P_0 e^R$$

Then divide both sides of the equation by P_0 and apply the natural logarithm (ln), utilizing Euler's number, e, as its base

$$ln\left(\frac{P_t}{P_0}\right) = ln(1+r) = lne^R$$

$$ln\left(\frac{P_t}{P_0}\right) = ln(1+r) = Rlne$$

The functions, $f(x) = lnx$ and $f(x) = e$, equate to 1 as they are inverses, leaving the equation to be

$$ln\left(\frac{P_t}{P_0}\right) = ln(1+r) = R$$

, where R, the value that is continuously compounded which will grow from P_0 to P_t, is the same value as $ln(1 + r)$

Thus, the computed log-returns formula

$$R_i = log(P_i/P_{i-1})$$

and volatility σ_t^2, where

$$\sigma_t^2 = Var\{Z_t^2 | F_{t-1}\}$$

where F_t represents the amount of information set available at time t and X_t is the random process can both be used for the analysis of time-series forecasting.

Autoregressive Time-Series Forecasting

ARCH Family Models

The ARCH family was created by Robert Engle in 1982 and evolved into a sphere of statistical tools used to forecast and analyze the volatility of financial derivatives using a time series. The ARCH family is comprised of 12 family models including the most prominent Autoregressive Conditional Heteroskedasticity (ARCH), Generalized Autoregressive Conditional Heteroskedasticity (GARCH), Integrated Generalized Autoregressive Conditional Heteroskedasticity (IGARCH), and GARCH-in mean (GARCH-M), etc. Throughout the process, all family functions are used for the characterizations of the observed time series.

ARCH Model

Proposed by Robert Engle in 1982, the Autoregressive Conditional Heteroskedasticity (ARCH) model utilizes log-returns as a product of Gaussian White Noise and volatility in a conditional variance process with an auto-regressive structure. In the ARCH process, the Gaussian White is described a random, stationary process (with a mean of zero) in which any two values in the set of data are statistically independent. Let ϵ_t be the Gaussian white noise with a unit variance in the data. a_t is the process of ARCH

$$Z_t \equiv \sigma_t \epsilon_t$$

where

$$\sigma_t = \sqrt{\omega + \sum_{i=1}^{p} \alpha_i Z_{t-i}^2}$$

Here, the formula models the heteroskedasticity by correlating the linear combination of squared disturbance in earlier data to the conditional variance of the disturbance term.

G(ARCH) Model

The GARCH model is regarded as Bollerslev's earliest and most fundamental symmetric model. The GARCH model extends the ARCH models by modeling the present variance of financial data at time t using values of previous squared returns and prior variances. As seen below, the GARCH (p,q) model is designated as a linear function of previous squared residuals and lagged conditional variances, where p is notated as a sequence of the series in terms of σ^2 and q of the ARCH in terms of a^2

$$Z_t \equiv \sigma_t \epsilon_t$$

$$(\epsilon_t | \psi_{t-1}) \sim N(0, \sigma_t^2)$$
$$t = 1,2,\ldots,n$$
$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i Z_{t-i}^2 + \sum_{i=1}^{q} \beta_i \sigma_{t-i}^2, \forall_t \in \mathbb{Z}$$

where $a_i \geq 0$, $\beta_j \geq 0$, $\omega > 0$, $i = 1, 2..., q$, $j = 1, 2..., p$, $a_t/I_t \sim (0, \sigma^2)$. a_i and β_i are the coefficients of any unknown parameter in the data, ϵ_t are the error terms, I_t is the set of information available at time t, and a_t is the dependent variable.

ARMA Model

The Autoregressive-Moving-Average model is a parsimonious model theorized by Peter Whittle in 1951 which aims to calculate the stationary stochastic process for two polynomial functions, the autoregression (AR) and the moving average (MA) given a set of data. The Box-Jenkins method, which employs the ARMA model to create a best-fit in order to model a forecasting time series paradigm, is used to estimate the model. The auto-regressive function is then used in order to generate a regression of the past (lagged) values in the data. The

moving-average function is used in order to create a modelling of the error (or residual) as a linear combination. Below, both functions in the ARMA model are represented.

Autoregressive model

Below, the first function in the ARMA model, the Autoregressive Model, is split into its components.

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

where c is the constant, ε_t, is the Gaussian white noise, and $\varphi_1,...,\varphi_p$ are the parameters from the set of data.

Moving-Average Model

Below, the moving-average of the data is modelled with order q

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

where μ is the assumption of X_t, ε_i ... is the Gaussian White Noise, and $\theta_1,...,\theta_q$ are the parameters in the set of data.

Combined ARMA (p,q) Model

Below, the pairing of both the Autoregressive model and the Moving-Average model is presented, thus equating to

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

where p represents the Autoregressive function and q is the Moving-Average function.

ARIMA Model

The Autoregressive Integrated Moving Average model is a generalization of the ARMA model. The model is used in sets of data where non-stationary exists in the mean. The "integrated" component in the ARIMA model is referred the number of times needed for the model to achieve its stationarity. Given data from time t, the formula below models the ARIMA model

$$\left(1 - \sum_{i=1}^{p'} \alpha_i L^i\right) X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

where α_i represents parameters of the AR portion, θ_i represents the parameters of the IMA component, ε_t are the error terms in the set of data, and L is the lag operator. The model is then used to identify any autocorrelation within the data.

Neural Network Volatility Forecasting

A neural network is a type of computer algorithm that uses approaches akin to those used in the human brain, hence its designation "neural." The algorithm is made to operate information as well as recognize sequences. The function is modeled after the components and activities of brain cells, specifically the nodes. The technique employs numerous basic processing components working in tandem to achieve high computing speeds. The

concept of Neural Network Volatility Forecasting has been a profound one developed by Ormoneit and Neuneier (1996). It was first used to predict volatility of financial equities in the Frankfurt Stock Market (FWB). Neural networks have consistently outperformed linear models in a number of situations. They've been especially successful at capturing complicated linkages where linear models fall short.

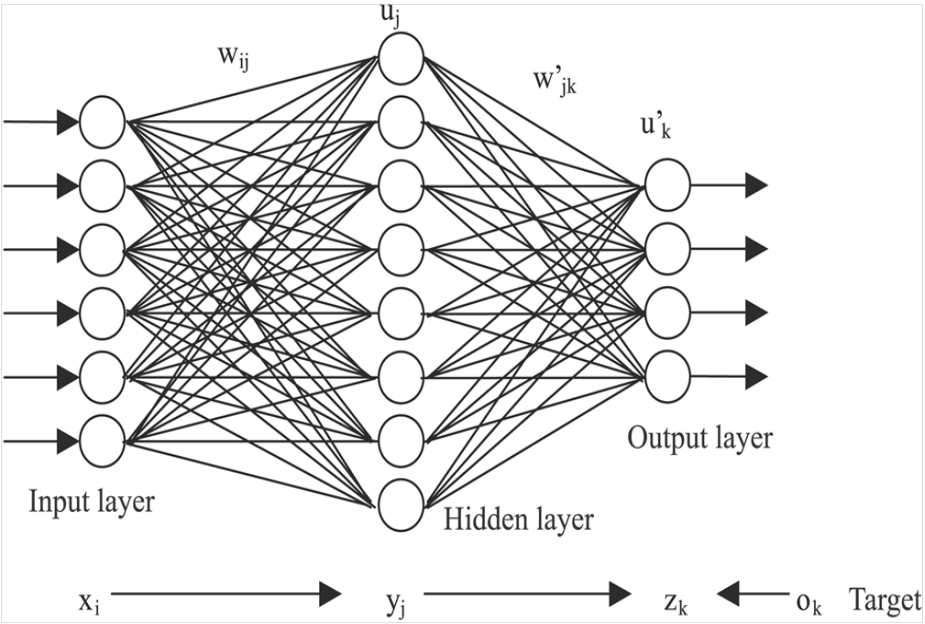

Figure 1. The most popular type of neural network structure for prediction is a feed-forward multilayered network. This form of network has at least three layers. The input layer is where data is presented to the network. The "hidden layer" is the next layer, in which it generates an internal delineation of the data presented to the network. The output layer is the last layer in which each component in its layer instantaneously gets information from neurons in the preceding layer. (Templeton, 2015)

Using LSTM-ANN Models

One of the most frequented types of Recurrent Neural Networks (RNN) utilized is within Long Short-Term Memory (LSTM). A recurrent neural network (RNN) is a type of a artificial neural network that uses time series to train input data, using previous inputs to affect new outputs. This recurrent neural network is suited for projecting time series and is designed to prevent long-term reliance difficulties. The LSTM model, proposed by Jurgen Schmidhuber and Sepp Hochreiter in 1997, is comprised of a distinctive collection of memory cells which restores the RNN's hidden layer nodes. To maintain the condition of the memory cells, the LSTM refines input via the gate structure. It has input, forgotten, and output gates in its door construction. Three sigmoid layers, a layer that decides if fresh data should be up to date and/or ignored, and one *tanh* layer, a layer that has all of the new input's potential values through the making of a vector, make up each memory cell. The *tanh* activation function is utilized in LSTM network to select a potential cell state as well as an upgraded hidden state.

The LSTM unit's forgotten gate selects which cell state data is removed from the model. The memory cell receives the previous moment's output h_{t-1} and the current information x_t as inputs and transforms them (through σ) into a long vector $[h_{t-1}, x_t]$ to become

$$f_t = \sigma(W_f \cdot [h_t - 1, x_t] + b_f)$$

where the weight matrix and bias of the forgetting gate, respectively, are W_f and b_f, and σ is the sigmoid function. The major purpose of the forgotten gate is to keep track of how much the previous moment's cell state C_{t-1} is reserved for the prior moment's cell state C_t. Based on h_{t-1} and x_t, the gate will produce a number between 0 and 1, with 0 denoting complete discard and 1 denoting total reserve.

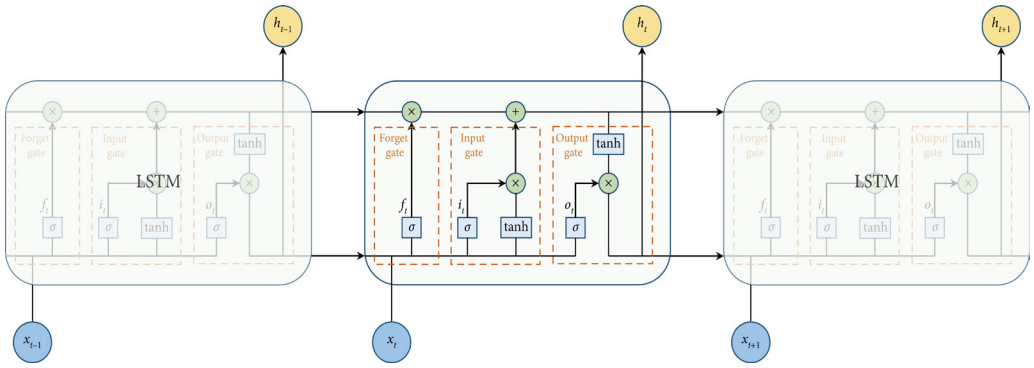

Figure 2. Above, the architecture of the LSTM model and its components are shown. It is composed of four gates: the learn gate, forget gate, remember gate, and use gate. Learn Gate: The instance that the information as well as the STM are combined so relevant information provided via the STM may be applied to the input in the instance. Forget Gate: When LTM goes through the forget gate, it rejects information that is not useful. Remember Gate: LTM information that is pertinent; in the updated LTM, the STM and instance is integrated into the Remember Gate. Use Gate: predicts the output of the current event and acts as a newly modified STM.

The input gate controls the quantity of the present instance input, x_t, is retained for the cell state C_t, preventing irrelevant data from reaching the memory cells. It serves two purposes: to determine the condition of the cell that has to be modified; the sigmoid layer selects the value to be updated. The other option is to update the information to reflect the current status of the cell. To regulate how much current information is aggregated, the *tanh* layer creates a candidate vector, $\hat{C}_t$.

$$i_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_i),$$

$$\hat{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c),$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t$$

A sigmoid layer determines the output information initially, then is refined by *tanh* and paired with the sigmoid layer output to generate a final output portion

$$O_t = \sigma(W_\sigma \cdot [h_{t-1}, x_t] + b_o)$$

This gives a final output cell value as

$$h_t = O_t \cdot \tanh(C_t)$$

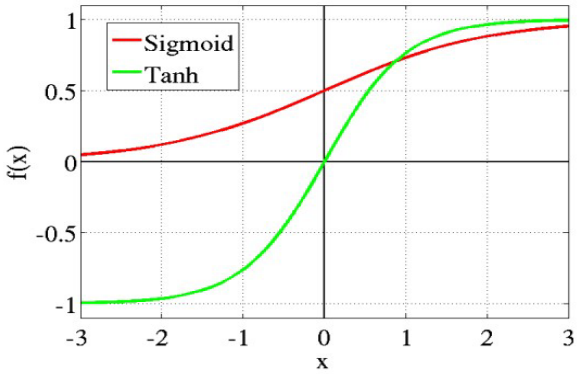

Figure 3. The sigmoid function is widely utilized since it contains values between 0 and 1, which contains all probability values. The tanh function contains values between -1 and 1 (Sharma, 2017)

Maximum Log-Likelihood Function

In the event that data in a set has heteroskedasticity, the Ordinary Least Square Estimation (OLS) as well as the co-variance matrix proves to be inefficient in calculating the maximization of parameters. In the ARCH family processes, the maximum log-likelihood function is used to maximize the parameters of the respective models. Maximum Likelihood estimation (MLE) chooses the model parameter values that have a higher likelihood of generating the data than any other parameter value. By definition, the realization of the conditional variance h_t is

$$L = \prod_{i=1}^{T} \sqrt{\frac{1}{2\pi h_t}}^{\left(\frac{-\varepsilon_t^2}{2h_t}\right)}$$

The conditional maximum likelihood (CML) approach is commonly utilized to approximate difficulty parameters of the log-likelihood function. The strategy is based on maximizing of these parameters' conditional likelihood given a set of minimal enough statistics for the ability parameters. Through this, the conditional log-likelihood is then defined as

$$\frac{1}{T-1}\log f(x_2 \ldots\ x_T|x_1;\ \theta) = \frac{1}{T-1}\log L\,(\theta|x_1, x_2 \ldots x_T\,)$$

$$= -\frac{1}{2}\log(2\pi)\frac{1}{2(T-1)}\sum_{t=2}^{T}\log\sigma_t^2\ -\ \frac{1}{2(T-1)}\sum_{t=2}^{T}\frac{\varepsilon_t^2}{\sigma_t^2}$$

However, in order to optimize the ARCH or GARCH processes' parameters, α and β, the Maximum Likelihood estimator (MLE) function is used. Any estimate, $\hat{\theta}_n$, that optimizes the likelihood function inside any parameter space (set of data), Θ, is referred to as an ML estimator, where

$$\hat{\theta}_n = argmax_{\theta\in\Theta} L_n(\theta)$$

$$= argmax\left\{-\frac{1}{2}\log(2\pi)\ -\ \frac{1}{2(T-1)}\sum_{t=2}^{T}\log\sigma_t^2\ -\ \frac{1}{2(T-1)}\sum_{t=2}^{T}\frac{\varepsilon_t^2}{\sigma_t^2}\right\}$$

By definition, the multiplication of the marginal densities is congruent to the joint-density of a sample where

$$f(x_1,\ldots,x_T;\theta) = f(x_1;\theta)\ \cdots\ f(x_T;\ \theta) = \prod_{i=1}^{T} f(\,x_T;\ \theta)$$

T, the joint density of the data $(x_1,\cdots,x_T)$, given the parameter θ, satisfies

HIGH SCHOOL EDITION
Journal of Student Research

Volume 11 Issue 3 (2022)

$$f(x_1, \ldots, x_T; \theta) \geq 0,$$

$$\int \cdots \int f(x_1, \ldots, x_T; \theta) dx_1 \cdots dx_T = 1$$

The MLE function was then used in Python through MATLAB, a programming language that plots a plethora of functions given different sets with volatile data. The code below illustrates that the Maximum Likelihood estimator converges to the true values after a Monte Carlo simulation. Furthermore, the GARCH procedure is implemented using the student-t innovation.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Report Matlab code for Maximum Likelihood estimation of the GARCH model; report a Monte Carlo simulation which shows that the Maximum Likelih
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% number of repetitions for the Monte Carlo simulation:
repetitions = 2000;

% initialize matrices for coefficients (ML estimator):
MLE_alpha= NaN(1,repetitions);
MLE_beta= NaN(1,repetitions);

for w = 1: repetitions

%% sample size for the GARCH process is set to 1000:
N=1000;

%% generate the t5-student innovation for the GARCH process
t5_distr = trnd(5,[N,1])';
% generate t5-student innovation with mean zero and variance 1:
epsilon =  (t5_distr  - mean(t5_distr) )./ (sqrt(var(t5_distr)  ));

%% parameter values (GARCH volatility equation):
beta0= 0.8;
alfa0=0.1;

%% generate the data:
h=zeros(1,N);
y=zeros(1,N);
h(1)= 1./ (1-alfa0-beta0);
y(1)= ((h(1)).^0.5) * (epsilon(1));

for  i=1: (N-1);
    h(i+1)= 1+ beta0 .* h(i) + alfa0 .* ((y(i)).^2);
    y(i+1)=  ( (h(i+1)).^0.5) * (epsilon(i+1));
end

%% starting values for the estimation of parameters (maximum likelihood estimation):
startingvalues = [0.0001;0.0001; 0.0001];
lowerbound = [  0 ;  0  ; 0];
upperbound = [  1;  1; 10];

%% estimate the parameters using MLE (maximum likelihood estimation)
likelihood = @(x) MLE_t5_NEW(x(1),x(2),x(3), y,h );

options    = optimset('fmincon');
options    = optimset(options, 'TolFun', 1e-006);
options    = optimset(options, 'LargeScale', 'off');
options    = optimset(options, 'MaxFunEvals', 1000);
options    = optimset(options, 'MaxIter', 400);
options = optimset('maxfunevals',20000);

[PARMLE, fval] = fmincon(likelihood,startingvalues,[],[],[],[],lowerbound ,upperbound , @mycon ,options);

MLE_alpha(1,w)= PARMLE(1,1);
MLE_beta(1,w)= PARMLE(2,1);
MLE_sigma(1,w)= PARMLE(3,1);

% means and standard deviation of the GARCH parameters using the ML
% estimator:
Mean_alpha_MLE= mean(MLE_alpha,2)
Mean_beta_MLE= mean(MLE_beta,2)
Std_dev_alpha_MLE =std(MLE_alpha)
Std_dev_beta_MLE= std(MLE_beta)
Mean_sigma_MLE= mean(MLE_sigma,2)
```

Figure 4. Algorithmic code used through MATLAB for the Maximum Log-Likelihood for GARCH and ARCH parameters, α and β. Uses a Monte Carlo simulation as well as a student-t innovation. (Created by Student Researcher)

Hessian Matrix

Another approach to maximizing the ARCH family processes' parameters is through the Hessian Matrix, a scalar field that comprises a square matrix of second-order partial derivatives of a scalar-valued function. The Hessian matrix function, notated as **H**, is modeled as

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

which can simplify (based off its coefficients) to

$$(\mathbf{H}_f)_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

This matrix is employed within the algorithm and is used to compared to the maximum likelihood function in order to fully optimize the parameters.

Econometric Results

After putting the data under their respective conditions, it was found that the most effective way of calculating whether the ARCH or GARCH model best fits the financial data was through the use of mean error calculations. These calculations are mainly used to evaluate how well a collection of data represents the population as a whole. Below, three of the mean error formulas are shown that were used throughout the duration of this project.

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

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{n}}$$

$$MAD = \frac{\sum|y_i - \hat{y}_i|}{n}$$

Over the 5-year fiscal period, from January 1, 2014 to January 1, 2019, a total of 1,770 prices, *n*, (from monthly data) were used through the use of database MarketWatch. In the table below, the respective mean error calculations were done for the ARCH/GARCH models.

Table 1. Mean Errors for Traditional ARCH/GARCH Tests (Created by Student Researcher)

MSE	RMSE	MAD
1.3677	1.1695	0.8537

This data presented shows that the ARCH/GARCH tests done, on the sample size of 1770, performed fairly well indicating a moderately strong RMSE value.

The neural network process underwent the same ARCH/GARCH process, except with addition of the maximization of the log-likelihood function and LSTM training model. Below are the results of the constant mean model, the mean model, and volatility model, respectively.

Table 2. GARCH Model Constant Mean Results (Created by Student Researcher)

Mean Model	Constant Mean	**R-Squared**	-0.122
Vol. Model	GARCH	**Log-Likelihood**	-13339.2
Method	Maximum Likelihood	**No. Observations**	1770
Distribution	Normal	**Dep. Variable**	Volatility

Table 3. GARCH Mean Model Results (Created by Student Researcher)

	Coefficient	**Std. Err.**	**95.0 Conf. Int.**
mu	21.6127	0.586	(21.59, 21.64)

Table 4. GARCH Volatility Model Results (Created by Student Researcher)

	Coefficient	**Std. Err.**	**95.0 Conf. Int.**
Omega	0.8342	0.217	(0.824, 0.844)
Alpha	0.8765	0.0582	(0.874, 0.879)
Beta	0.1471	0.0505	(0.145, 0.149)

In the neural network process, the LSTM model was utilized to further train the data using epochs in a cycle of time. The epochs underwent training to further increase the accuracy of the model. Below, the graph illustrates the training of 150 epochs using the data.

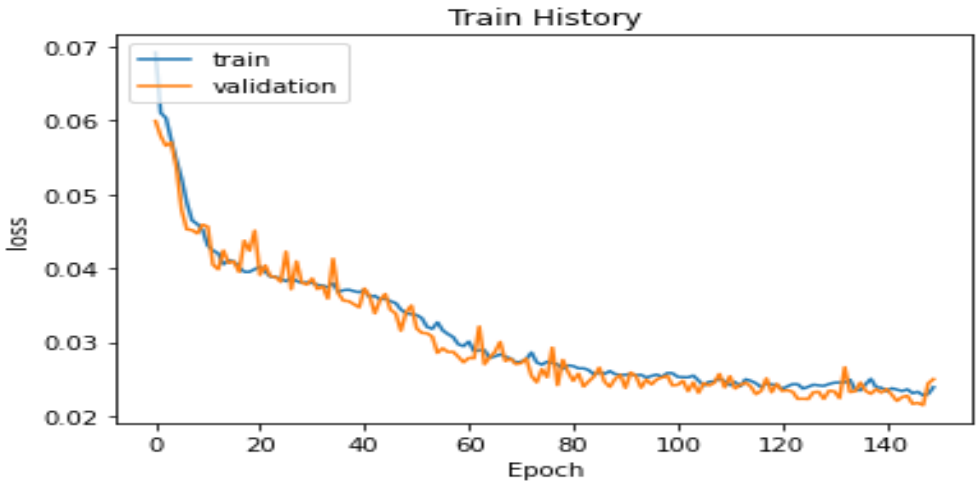

Figure 5. Illustrates the training of the LSTM model in the GARCH neural network with 150 epochs. (Created by Student Researcher)

After the data set had been trained, the model, where $n = 1770$, was compared to the real rolling volatility of the 30 equities. As shown below, the LSTM model is graphed on the actual stock volatility.

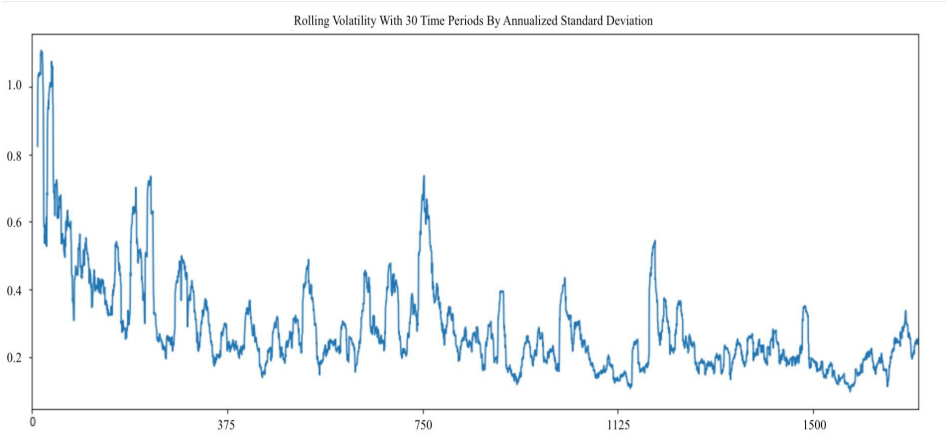

Figure 6. The real rolling volatility of the 30 equities characterized by annualized standard deviations. (Created by Student Researcher)

Lastly, the combination of the LSTM model and the GARCH model produces a graph that is mapped on to each other to illustrate the efficacy of both forecasting methods.

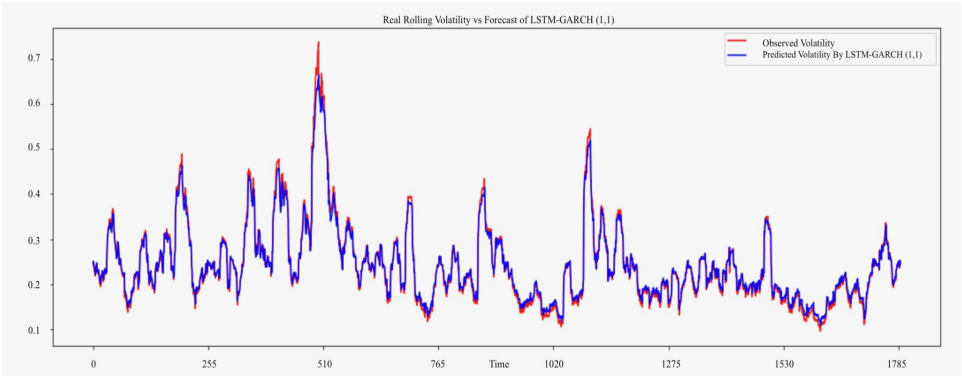

Figure 7. The real rolling volatility of the 30 equities mapped on the LSTM-GARCH model of 1770 observations. (Created by Student Researcher)

With the new addition of the LSTM model, the mean error statistics greatly improved, suggesting a more accurate forecast.

Table 5. Mean Errors for Neural Network Enhanced LSTM GARCH Models (Created by Student Researcher)

MSE	**RMSE**	**MAD**
0.7679	0.8763	0.6396

Conclusion

In this study, various modes of predicting volatility were utilized, from traditional ARCH and GARCH family models, to Neural Network enhanced algorithms. Upon statistical analysis, it was found that the Neural Network model greatly outperformed the traditional ARCH and GARCH models. With the neural network's RMSE value of 0.873 and the traditional ARCH/GARCH's RMSE value of 1.1695, it can be concluded that the neural network LSTM model outperformed the traditional ARCH/GARCH model. With its higher accuracy rate as well as efficiency rate, it was concluded that the new and improved neural network LSTM model would be able to detect more anomalies in volatility and further forecast the data, while helping institutions and individual investors choose their portfolios of stocks based on their relative fluctuations and volatility.

The new concept of financial forecasting is one that is very extensive. New forms of methodologies are constantly being created in order to create more accurate models. These creations yield many benefits and aid individual investors as well as financial institutions. Moreover, with the direct relation to the Markov Chain and GARCH model, it would be a feasible endeavor to further explore the nexus of these two spheres through Markov Switching Models. This is a division of nonlinear time series models that are used to project asset returns by blending heterogeneous duration of a data set with stochastic volatility components (Xie. Y, 2007). This family relies its accurate predictions on the expectation maximization algorithm, which focuses on data analysis involving an unobserved variable. Moreover, the adaptability of the model contributes to its usefulness when attempting to represent data that seems to cycle through different time periods. For future work, Markov Switching Models can be compared to neural networks in a test for efficacy. Both models would be put through a series of scenarios that revolve around financial fluctuations and be examined for accuracy when it pertains to sensitivity to change.

Acknowledgments

I would like thank my advisor for the valuable insight provided to me on this topic.

References

Bollerslev, T. (2008). Glossary to arch (garch). *CREATES Research paper*, *49*.

BookDown . (n.d.). https://bookdown.org/compfinezbook/introcompfinr/maximum -likelihood-estimation.html#the-likelihood-function. (Accessed: 2022-0130)

Brownlees, C. T., Engle, R. F., & Kelly, B. T. (2011). A practical guide to volatility forecasting through calm and storm. *Available at SSRN 1502915*.

Christensen, K., Siggaard, M., Veliyev, B., et al. (2021). *A machine learning approach to volatility forecasting* (Vol. 3). Department of Economics and Business Economics, Aarhus University.

GitHub. (n.d.). https://github.com/. (Accessed: 2022-02-01)

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Jia, F., & Yang, B. (2021). Forecasting volatility of stock index: deep learning model with likelihood-based loss function. *Complexity*, *2021*.

Malliaris, M., & Salchenberger, L. (1996). Using neural networks to forecast the s&p 100 implied volatility. *Neurocomputing*, *10*(2), 183–195.

MarketWatch . (n.d.). https://www.marketwatch.com/. (Accessed: 2022-01-36)

Ormoneit, D., & Neuneier, R. (1996). Experiments in predicting the german stock index dax with density estimating neural networks. In *Ieee/iafe 1996 conference on computational intelligence for financial engineering (cifer)* (pp. 66–71).

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *towards data science*, *6*(12), 310–316.

Templeton, G. (2018). Artificial neural networks are changing the world. *What are they*.

Xie, Y. (2007). *Maximum likelihood estimation and forecasting for garch, markov switching, and locally stationary wavelet processes* (Vol. 2007) (No. 2007: 107).