Journal of
Risk and Financial Management

MDPI

Article

Encoder–Decoder Based LSTM and GRU Architectures for Stocks and Cryptocurrency Prediction

Joy Dip Das [1,*,†], Ruppa K. Thulasiram [1,*,†], Christopher Henry [1,†] and Aerambamoorthy Thavaneswaran [2]

[1] Department of Computer Science, University of Manitoba, Winnipeg, MB R3T 5V6, Canada; christopher.henry@umanitoba.ca
[2] Department of Statistics, University of Manitoba, Winnipeg, MB R3T 2N2, Canada; aerambamoorthy.thavaneswaran@umanitoba.ca
* Correspondence: dasj@myumanitoba.ca (J.D.D.); tulsi.thulasiram@umanitoba.ca (R.K.T.)
† Current address: E2-445 EITC, 75A Chancellors Circle, Winnipeg, MB R3T 5V6, Canada.

Abstract: This work addresses the intricate task of predicting the prices of diverse financial assets, including stocks, indices, and cryptocurrencies, each exhibiting distinct characteristics and behaviors under varied market conditions. To tackle the challenge effectively, novel encoder–decoder architectures, AE-LSTM and AE-GRU, integrating the encoder–decoder principle with LSTM and GRU, are designed. The experimentation involves multiple activation functions and hyperparameter tuning. With extensive experimentation and enhancements applied to AE-LSTM, the proposed AE-GRU architecture still demonstrates significant superiority in forecasting the annual prices of volatile financial assets from the multiple sectors mentioned above. Thus, the novel AE-GRU architecture emerges as a superior choice for price prediction across diverse sectors and fluctuating volatile market scenarios by extracting important non-linear features of financial data and retaining the long-term context from past observations.

Keywords: autoencoder; LSTM; GRU; hybridization; stocks; stock index; cryptocurrency

Citation: Dip Das, Joy, Ruppa K. Thulasiram, Christopher Henry, and Aerambamoorthy Thavaneswaran. 2024. Encoder–Decoder Based LSTM and GRU Architectures for Stocks and Cryptocurrency Prediction. *Journal of Risk and Financial Management* 17: 200. https://doi.org/10.3390/jrfm17050200

Academic Editor: Jong-Min Kim

Received: 25 March 2024
Revised: 6 May 2024
Accepted: 6 May 2024
Published: 12 May 2024

1. Introduction

Various financial instruments and cryptocurrencies introduce complexities with their volatile nature for buying and selling ownership of shares. The dynamics of the stock and cryptocurrency markets are influenced by a multitude of factors such as political, geographical, and socio-economic considerations. The pronounced variability across these diverse factors contributes to fluctuations in stock market trends.

Predicting stock prices involves traditional methods like analyzing historical data and patterns to inform investment decisions. These commonly involve examining past stock price data through statistical techniques like the use of moving averages (see Metghalchi et al. (2012)), auto regressive integrated moving average (ARIMA) modeling (see Banerjee (2014)), exponential smoothing (see McMillan (2003)), and so forth. These techniques often overlook the temporal dependencies present in stock data, a crucial consideration for accurate stock market prediction. Additionally, numerous factors influence stock trends, making it critical to include all of them for effective predictions. Various studies indicate that machine learning (ML) and deep learning (DL) [1] methodologies exhibit superior performance in stock price forecasting when contrasted with conventional statistical methods. This superiority arises from their adeptness in managing intricate, non-linear patterns and handling extensive datasets Hiransha et al. (2018); Sirisha et al. (2022).

ML algorithms are now being heavily explored in stock price prediction (see Kumbure et al. (2022)). Traditional ML-based regression techniques such as linear regression (see Cakra and Distiawan Trisedya (2015)), support vector regression (SVR) (see Lawal et al. (2020)), decision tree regression (see Hindrayani et al. (2020)), random forest (RF) regression (see Chen (2023)), etc., are used in stock prediction. Moreover, these traditional

techniques have also been pursued in cryptocurrency prediction (see Khedr et al. (2021)). DL algorithms such as artificial neural networks (ANNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), etc., are also used in stock market prediction (see Shahi et al. (2020)). ANNs and CNNs do not consider temporal dependencies, thus RNN algorithms, for example, long short-term memory (LSTM) and gated recurrent unit (GRU) are heavily used in stock market prediction. These DL techniques are also used for highly volatile cryptocurrency price prediction (see Pintelas et al. (2020)). They are good at finding subtle patterns and trends in financial data that other methods might miss.

In addition to the models mentioned above, the autoencoder is used as a popular DL model in computational linguistics and representation learning (see Liou et al. (2014)). An autoencoder is a non-linear smart data compressor and decompressor. During training an autoencoder will learn to shrink down complex information into a compact representation (encoding), and then, expand it back to its original form (decoding). The advantage of autoencoders is that they capture the essential features of data in a non-linear fashion. This results in a reduction in unnecessary input data details and enables the extraction of patterns necessary to make decisions for a given application or problem. Autoencoders have also been used for constructing asset pricing models by Gu et al. (2021). Amalgamating models from, e.g., the autoencoder family of reduction models, with other DL techniques and architectures is leading the application of ML methods to produce prominent advancement across diverse domains. Overall, hybridization of established or traditional ML methods with newer DL architectures broadens the spectrum of applications and can improve performance across several domains. LSTM-GRU hybridization is being heavily explored in the financial time-series forecasting problem domains. In addition, LSTM-CNN and GRU-CNN also have been explored for different problem domains including finance by Xia et al. (2020). Other than these models, LSTM-GRU-ARIMA, by Pierre et al. (2023), is also a popular hybridized model that can be used in different problem domains. Rubell et al. (2023) conducted experiments employing a multivariate sequential LSTM autoencoder on various stocks, including AAPL, GOOGLE, JPM, JNJ, etc., demonstrating its superiority over univariate sequential LSTM autoencoder, GRU, and generative adversarial networks (GANs). Notably, the authors did not delve into hyperparameter tuning, leaving room for potential enhancements in model performance. Moreover, experimentation with the number of layers is also a necessary investigation that was not considered previously.

In computational finance, hybridization primarily aims to integrate a range of ML and DL models across diverse market contexts. However, their efficacy varies across distinct market conditions. Presently, there is an absence of well-established models that comprehensively account for market dynamics and exhibit performance across diverse market scenarios. Additionally, the stock dataset encompasses a multitude of underlying features necessitating meticulous consideration through diverse statistical measures. This research involves design and experimentation for the novel hybridization of autoencoders with LSTM and GRU models for retaining long-term context and non-linear important features from past observations. Our objective is to augment predictive accuracy, particularly in the context of forecasting stock prices, stock indices, and cryptocurrency values. In this paper, we have introduced two innovative encoder–decoder architectures, AE-LSTM and AE-GRU, based on LSTM and GRU frameworks, respectively. Our proposed architectures exhibit superior performance compared to existing models, demonstrating remarkable consistency in predicting various classes of financial assets across diverse market conditions. Particularly noteworthy is the AE-GRU model, which surpasses the enhanced AE-LSTM counterpart, showcasing exceptional efficacy in effectively forecasting amidst fluctuating levels of market volatility. Moreover, our model demonstrates high predictive accuracy while operating efficiently in resource-constrained environments, thus offering a cost-effective alternative to the computationally intensive transformer architectures.

The subsequent sections of this article are structured as follows: Section 2 provides a comprehensive literature review to establish the contextual background of the research. Section 3 delineates the methodology, elucidating the architectures, algorithms, and hyper-

parameters of the proposed models, alongside insights into grid-search-based hyperparameter tuning, activation functions, and evaluation metrics. In Section 4, we analyze the empirical evidence gleaned from experimentation and assess model performance. Section 5 delves into the practical implications and potential applications of the research findings. Furthermore, Section 6 encapsulates the concluding remarks drawn from our study, while Section 7 outlines potential avenues for future research endeavors.

2. Related Works

Machine learning has grown tremendously in many areas of research and applications. It would be a daunting task to comprehensively cover all work from the literature from multiple areas, and hence, our focus in this section is limited to financial applications.

Dimensionality reduction techniques can reduce the redundancy in stock market data and can result in efficient price prediction. Zhong and Enke (2017) have used dimensionality reduction techniques along with ANNs for daily stock market return. However, their dimensionality reduction techniques utilized different variations of principle component analysis (PCA) which only capture linear/planar intricacies among the data. The stock market is mainly affected by non-linear complexities. Kohli et al. (2019) forecast the movements of the Bombay Stock Exchange (BSE) by considering various factors such as market history, commodity prices, and foreign exchange rates. They found gold prices to affect BSE the most. Their analysis revealed the superiority of the AdaBoost algorithm among others explored.

The application of DL models is rising phenomenally in the arena of stock prediction (see Hu et al. (2021)). Jiang (2021) has summarized the recent progress in stock market prediction using different DL models and provided a general workflow for the stock prediction domain using DL. Nikou et al. (2019) considered the non-linearity and non-stationarity of stock market data when they analyzed different ML and DL algorithms in predicting the daily close price data. Their findings suggest DL models surpass traditional ML algorithms. They also noted that SVR and RF perform well next to DL models such as deep neural networks (DNNs). Chen et al. (2024) employed DNNs to develop an asset pricing model for individual stock returns, utilizing abundant conditioning information and incorporating time variation, thereby achieving superior out-of-sample performance compared to benchmark approaches, and identifying key driving factors for asset prices. Zhou et al. (2023) developed DNN models for equity-premium forecasting and compared their performance with ordinary least squares (OLS) and historical average (HA) models, demonstrating the superior predictive accuracy of DNN models both in- and out-of-sample, along with enhanced performance from additional finance variables. The above-mentioned ML and DL algorithms do not consider temporal dependencies, thus RNN variants such as LSTM and GRU, which consider past temporal values of the stock market, are being explored in stock market behavior prediction.

LSTM is found to be inefficient when working with smaller datasets, which is often considered a drawback by certain researchers, such as Sheth and Shah (2023). The LSTM model has been also hybridized in the prediction of the non-stationary and non-linear stock market (see Ali et al. (2023)). Apart from LSTM, GRU has also been explored in stock prediction quite extensively (see Qi et al. (2023)). LSTM outperforms ML algorithms like SVR in several markets Karmiani et al. (2019). However, the architecture of LSTM can be market-dependent. Jiang et al. (2019) have explored the functionality of LSTM in different markets such as the Chinese, European, and American markets. Their findings suggest the superiority of LSTM in the European and American markets compared to the more rational Chinese market. Furthermore, Fang et al. (2023) have proposed a hybrid forecasting framework using LSTM, a batch normalization layer, a dropout layer, and a binary classifier for trend forecasting of financial time series. Their study used the S&P500, Shanghai Stock Exchange 180, and China Securities Index 300 for their experimentation. From their experimentation, it is evident that the performance of hybridized DL models can be generalized to different datasets. Gupta et al. (2021) have performed stock prediction

based on LSTM and GRU along with incorporating PCA and least absolute shrinkage and selection operator (LASSO) techniques. Kim et al. (2021) carried out a comparative assessment of LSTM and GRU for cryptocurrency price prediction, revealing that GRU outperforms LSTM in specific situations. DL algorithms such as autoencoders are also very useful in financial research.

Various types of autoencoders are used for non-linear feature extraction. Gradxs and Rao (2023) have used deep stacked autoencoder in behavior-based credit card fraud detection. They have combined Harris Grey Wolf Network with a deep balanced stacked autoencoder. Fanai and Abbasimehr (2023) have used hybridized autoencoder and deep classifiers for credit card fraud detection. Their experimentation shows the superiority of autoencoder over PCA. For proper prediction, consideration of multiple features is also very important, thus, hybridization helps in this scenario. Through hybridizing several DL models, multiple features of different algorithms can be incorporated into a single model. This helps to capture previously unnoticed complexities behind the scenes. LSTM has been combined with adaptive boosting (AdaBoost) and k-nearest neighbors (KNNs) Zhao et al. (2023). Compared to traditional ML models, the performance of this combined approach was significant. Kwak and Lim (2021) experimented on the AdaBoost and GRU ensemble model and noticed increased efficiency compared to the LSTM, GRU, and ARIMA models on the KOSPII market. Previously, Shen et al. (2018) have experimented with GRU by replacing the last layer with SVM for trading-signal prediction. Their study showed significant performance enhancement when ML and DL models were hybridized. Hossain et al. (2018) have reported better predictability of prices with LSTM-GRU hybridization. Song and Choi (2023) have explored various hybridizations of DL models and observed a significant rise in efficiency compared to other traditional models. Fang et al. (2023) have proposed a hybrid forecasting framework using LSTM, a batch normalization layer, a dropout layer, and a binary classifier to forecast trends in financial temporal data. From their experimentation, it is evident that the performance of hybridized DL models can be generalized to different datasets. Also, in the realm of cryptocurrency prediction, hybridization is a prominent technique. Petrovic et al. (2021) used a hybridized LSTM-GRU model along with swarm intelligence for cryptocurrency prediction.

Retaining long-term context in financial time-series data is very important since predicting future values of financial assets depends heavily on important past events. Hence, the context of those events should be remembered by the model including important non-linear features of time-series data. In this research, we propose two novel models with an efficient solution through extensive hyperparameter tuning.

3. Methodologies

In this section, we describe our proposed hybridization of the classic LSTM and GRU architectures with the autoencoder architecture. The basic LSTM and GRU architectures, though they are well-known and well-studied, are described in Appendix A for the sake of completeness.

3.1. Autoencoder (AE)

An autoencoder is a neural network architecture designed for unsupervised learning, consisting of an encoder and decoder. It aims to learn a compact representation of input data by reducing the reconstruction error between the original and reformed data. Figure 1 is an illustration of the autoencoder architecture.

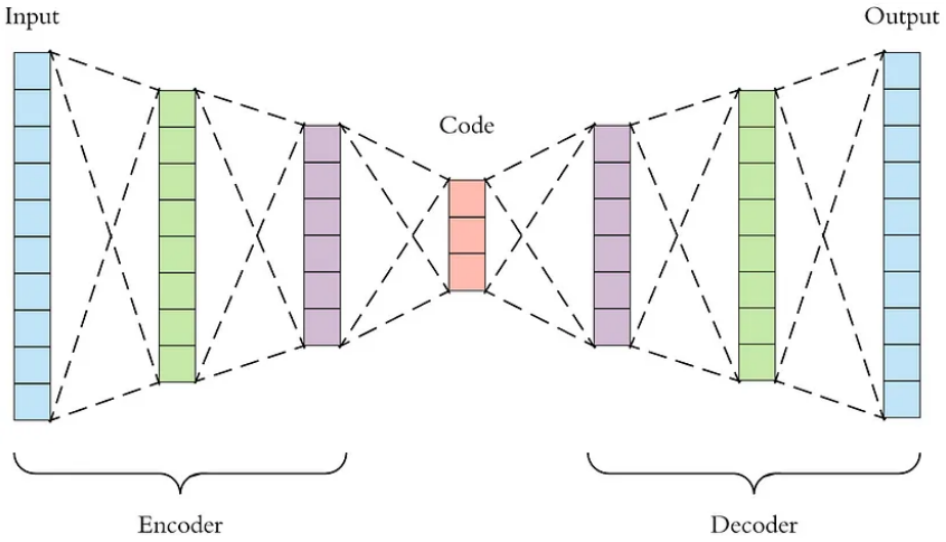


Figure 1. A general autoencoder architecture.

A generalized equation of autoencoder is as follows:

$$x \xrightarrow{\text{Encoder}} h = e(x) \xrightarrow{\text{Decoder}} \hat{x} = d(h).$$

Here, an encoder is a function that takes input x and is defined as

$$h = e(x).$$

The architecture of an autoencoder is designed to extract the hidden representation of input data through its encoder component. This representation, which exists in a reduced dimensional space, allows the model to discern non-linear characteristics of the input feature vector. In forecasting time series, these non-linear characteristics can offer valuable insights into the non-linear relationships among data points, which are crucial for forecasting future values. In the given equation, h signifies the hidden representation of the input data as obtained by the encoder component. The decoder component then reconstructs the data, denoted by $\hat{x}$, by minimizing the reconstruction error. This process of reconstruction eliminates noise or outliers that are not correlated with the dataset, leaving behind only the significant features that are relevant to the context of the dataset. Autoencoders have proven to be effective tools for denoising across various fields (see Lu et al. (2013), Saad and Chen (2020)).

3.2. Stacked AE-LSTM and AE-GRU Architectures

In the realm of financial time-series forecasting, the process of extracting non-linear features from a dataset is of paramount importance. Financial time-series data are unique in that it often exhibits non-linear characteristics. This means that the relationships between variables in the data are not simply proportional, but can change in complex ways. These non-linear relationships can be influenced by a multitude of factors, making them challenging to model accurately.

Algorithms designed for prediction tasks, such as those used in machine learning, are capable of learning from these inherent patterns within the data. They do this via a process called training, which adjusts models to minimize the difference between their predictions and the actual data. Over time, this training process allows the algorithm to improve its predictive accuracy.

LSTM and GRU are commonly employed in the prediction of stocks and cryptocurrencies. However, these basic LSTM and GRU structures do not take into account the significance of non-linear features. In natural language processing (NLP), both LSTM and GRU encounter difficulties in predicting words when the sentence length increases

significantly. Consequently, these algorithms struggle to remember the context over the very long term. To address this issue, the Transformer network (TN) was introduced, which uses an attention mechanism in an encoder–decoder architecture to remember the long-term context (see Vaswani et al. (2017)). However, the encoder–decoder architecture of TN does not compress the input vector for learning intricate features from the latent representation. The encoder component of the Transformer handles the processing of the input sequence, while the output sequence is produced by the decoder. However, the implementation of such a complex model is computationally demanding. In the prediction of financial time series, it is crucial to remember the long-term context of past significant market events. This aids in identifying similar trends and predicting future values based on these trends. Therefore, the basic LSTM and GRU architectures are not the optimal solution for this task. Typically, for a one-day-ahead forecast for stocks and cryptocurrencies, the training dataset is usually very large. Consequently, running such a large dataset with a TN architecture becomes computationally prohibitive. Encoder–decoder-based RNN architectures demonstrated effective performance when an autoencoder-based LSTM was proposed by Rubell et al. (2023). To a certain degree, the encoded latent representation of the input data preserved the long-term contextual information about past observations, which helped to enhance the predictive accuracy of the basic LSTM architecture. In this study, we found that significant improvements in predictive accuracy could be achieved through extensive experimentation with the model's hyperparameters. However, in some studies, the GRU outperformed the LSTM in terms of achieving higher predictive accuracy (see Pirani et al. (2022), Salimath et al. (2021)). Therefore, this research also examines the accuracy of such an encoder–decoder architecture with GRU and proposes a new model with improved performance.

In this paper, we introduce innovative frameworks comprising stacked autoencoder-based long short-term memory (AE-LSTM) and gated recurrent unit (AE-GRU). The foundational structure of the proposed algorithm is elucidated in Figure 2. While both AE-LSTM and AE-GRU share a common architecture, distinct hyperparameters and activation functions tailored to each architecture were employed for optimal performance after strenuous investigation.

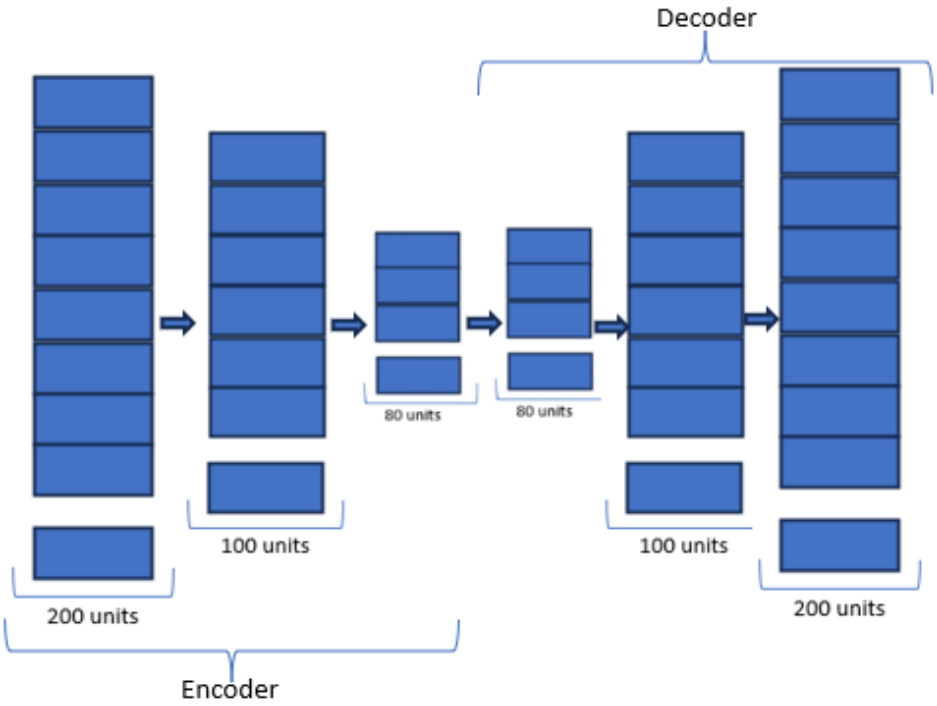


Figure 2. Encoder–decoder based novel LSTM and GRU architectures.

3.2.1. AE-LSTM

AE-LSTM, or autoencoder-based LSTM, is a hybrid neural architecture that combines the capabilities of an autoencoder with LSTM units. Autoencoders are employed to learn a compressed representation of input data, and this encoded information is then fed into LSTM layers, enabling the model to capture and leverage long-term dependencies in sequential data. AE-LSTM is particularly effective in tasks involving time-series data, such as stock price prediction, where both feature learning and sequential context preservation are crucial for accurate forecasting. Figure 2 represents the model structure of the proposed improved AE-LSTM architecture. The encoder–decoder architecture of AE-LSTM can be expressed as the following equations:

$$\text{Encoder:} \quad x \overset{\text{LSTM }200}{\longrightarrow} h_1 \overset{\text{LSTM }100}{\longrightarrow} h_2 \overset{\text{LSTM }80}{\longrightarrow} z,$$

$$\text{Decoder:} \quad z \overset{\text{LSTM }80}{\longrightarrow} h_3 \overset{\text{LSTM }100}{\longrightarrow} h_4 \overset{\text{LSTM }200}{\longrightarrow} \hat{x}.$$

Given the equation above, it is evident that the encoder structure employs several stacked LSTM layers to identify latent features at various compression levels. With each level of compression, the architecture learns more prominent features. However, additional compression leads to information loss. Such a deep architecture results in a higher reconstruction loss and is susceptible to the vanishing gradient problem. The initial compression occurs when the number of units in the second LSTM layer is reduced from 200 to 100. Over time, each layer experiences a reduction in units until the latent representation z is finally extracted from the LSTM layer with 80 units.

Similarly, the decoder layer incrementally increases the units in the subsequent LSTM layers. The entire structure is trained by minimizing the reconstruction loss. The output of the decoder is the reconstructed data that encapsulates the internal non-linear complexities of the financial time series. Ultimately, a dense layer is employed to obtain the predicted output.

Algorithm 1 shows an algorithmic view of the proposed novel improved encoder–decoder architecture of LSTM, which outperforms the existing models.

Algorithm 1 AE-LSTM

```
data ← parse("stock_name")
normalized_data ← normalize(data)
train_data ← normalized_data[start : end]
test_data ← normalized_data[start : end]
preprocessed_train ← preprocess(train)
encoder ← LSTM(units = 200, activation =' activation')(preprocessed_train)
encoder ← LSTM(units = 100, activation =' activation')(encoder)
encoder ← LSTM(units = 80, activation =' activation')(encoder)
decoder ← LSTM(units = 80, activation =' activation')(encoder)
decoder ← LSTM(units = 100, activation =' activation')(decoder)
decoder ← LSTM(units = 200, activation =' activation')(decoder)
model ← Model(encoder, decoder)
model.compile()
predict ← model.predict(test_data)
```

3.2.2. AE-GRU

AE-GRU, or autoencoder–gated recurrent unit, is a hybrid neural network architecture that combines the capabilities of an autoencoder with a gated recurrent unit (GRU). This innovative model integrates the feature learning abilities of autoencoders with the sequential modeling strengths of GRU, aiming to enhance the representation and prediction capabilities in various applications such as time-series analysis and sequential data processing. The AE-GRU architecture involves the use of a stacked autoencoder for feature extraction,

feeding the encoded features into GRU cells for capturing temporal dependencies and patterns. This combination leverages both unsupervised learning and RNN architectures to achieve improved performance in tasks requiring sequential data understanding and prediction. To our knowledge, this hybridization is a novel approach to financial asset price prediction. Figure 2 represents the model structure of the proposed novel AE-GRU algorithm. The AE-GRU architecture can be expressed as the following equations:

$$\text{Encoder:} \quad x \xrightarrow{\text{GRU 200}} h_1 \xrightarrow{\text{GRU 100}} h_2 \xrightarrow{\text{GRU 80}} z,$$

$$\text{Decoder:} \quad z \xrightarrow{\text{GRU 80}} h_3 \xrightarrow{\text{GRU 100}} h_4 \xrightarrow{\text{GRU 200}} \hat{x}.$$

In the above equations, we can observe that AE-GRU follows a similar architecture as AE-LSTM. Multiple encoder layers are layered to condense the feature vector into a representation in the latent space. After compressing to a feature vector, z, with a GRU layer consisting of 80 units, it is observed that further compression leads to a loss of information.

Several GRU layers, each with varying units, are stacked in a descending order to construct the encoder component of the AE-GRU architecture. This compression into the latent space z preserves valuable non-linear features of the financial time series. This latent feature vector encapsulates the long-term context derived from past observations.

Similarly, the decoder component incrementally increases the units in the GRU layers, resulting in an accurate reconstruction of the input data. This process eliminates outliers and extracts the non-linear contextual features from the financial data. Subsequently, a dense layer comprising a single unit is employed to predict future values.

Algorithm 2 shows an algorithmic view of the proposed novel encoder–decoder architecture of GRU, which outperforms the existing models and even the improved AE-LSTM designed in this study. This encoder–decoder-based GRU architecture is novel and tested rigorously under varying market conditions. This proposed novel model outperforms existing architectures by retaining long-term non-linear contextual features.

Algorithm 2 AE-GRU

$data \leftarrow parse(\text{“}stock\_name''\text{)}$
$normalized\_data \leftarrow normalize(data)$
$train\_data \leftarrow normalized\_data[start : end]$
$test\_data \leftarrow normalized\_data[start : end]$
$preprocessed\_train \leftarrow preprocess(train)$
$encoder \leftarrow GRU(units = 200, activation =' activation')(preprocessed\_train)$
$encoder \leftarrow GRU(units = 100, activation =' activation')(encoder)$
$encoder \leftarrow GRU(units = 80, activation =' activation')(encoder)$
$decoder \leftarrow GRU(units = 80, activation =' activation')(encoder)$
$decoder \leftarrow GRU(units = 100, activation =' activation')(decoder)$
$decoder \leftarrow GRU(units = 200, activation =' activation')(decoder)$
$model \leftarrow Model(encoder, decoder)$
$model.compile()$
$predict \leftarrow model.predict(test\_data)$

3.2.3. Hyperparameter Consideration

The process of tuning hyperparameters is a critical step in the development of ML or DL algorithms. Hyperparameters are distinct from other parameters in that their values are determined before the learning process begins. These are not derived from the data during training but are manually set to guide the learning process. The selection of these hyperparameters significantly influences the functioning of the algorithm. Incorrect choices could lead to sub-optimal performance, even for a model with high potential.

Number of Layers: We experimented with various configurations of encoder and decoder architectures and while multiple layer configurations were explored, exceeding a depth

of three layers led to a notable deterioration in model performance. In such instances, the efficacy of the model was significantly compromised. Thus, after thorough experimentation, we kept three layers in both the encoder and decoder. Increasing the number of layers increases the complexity of optimization.

Number of Units: Different numbers of units were tested for the encoder and decoder architecture. But the combination of units in the order illustrated in Figure 2 yields the best accuracy.

Activation Functions: In major financial research, three activation functions are most used in training the DL models: hyperbolic tanh, rectified linear unit (ReLU), and exponential linear unit (ELU). We tried combinations of different activation functions for each category of financial assets in this study. Details about the activation functions can be obtained from Section 3.3.

After hyperparameter tuning, from Table 1 it can be observed that different activation functions perform best for different financial assets. For general stocks across various categories, the proposed AE-LSTM model demonstrates its maximum efficacy when employing the ELU as the activation function. Conversely, the ReLU exhibits superior performance when integrated into the AE-GRU architecture. When focusing on relatively stable stock indices, such as the S&P, AE-LSTM exhibits improved performance with ReLU activation, while AE-GRU demonstrates higher accuracy levels with the tanh activation function. In the case of highly volatile cryptocurrencies, like Bitcoin, Table 1 illustrates that AE-LSTM, utilizing ELU activation, achieves maximum efficiency, whereas AE-GRU, with ReLU activation, displays effective predictive accuracy. Thus, it becomes evident that the activation functions selected for the proposed architectures exhibit similarities for moderately volatile stocks and highly volatile cryptocurrencies. This observation underscores the importance of hyperparameter tuning with respect to activation functions, revealing that the choice of activation function is contingent upon the inherent volatility of the financial assets under consideration.

Table 1. Activation functions for different financial assets in AE-LSTM and AE-GRU models.

Financial Asset	AE-LSTM Activation Function	AE-GRU Activation Function
General Stocks	ELU	ReLU
Stock Index	ReLU	tanh
Cryptocurrency	ELU	ReLU

Optimizers: An optimizer is a crucial algorithmic step to optimize the ML and DL algorithms so that the cost function is reduced, and proper weights are selected through backpropagation. In this study, we experimented with 4 major optimizers:

(a) Stochastic gradient descent (SGD): For this optimizer, the hyperparameters considered for optimization are learning rate and momentum.
(b) Adagrad: For this optimizer, only the learning rate was considered for hyperparameter tuning.
(c) RMSprop: For this optimizer, the learning rate and the decay rate were considered for hyperparameter tuning.
(d) Adam: For this optimizer, learning rate, and first- and second-moment exponential decay rates are considered for hyperparameter tuning.

After extensive hyperparameter tuning conducted via grid search, the Adam optimizer showed the highest predictive efficacy. This was achieved when configured with a learning rate, denoted by the symbol α, set to 0.001; a first-moment exponential decay rate, represented by β_1, set to 0.9; and a second-moment exponential decay rate, denoted by β_2, set to 0.999. This configuration was found to be optimal across various categories of financial assets.

Epochs and Batch Size: We selected various values of epochs for experimentation, including 60, 70, 100, 200, and 300 iterations. Following hyperparameter tuning through

grid search, the results indicated that selecting the number of epochs as 100 yielded the highest predictive accuracy. When we selected epochs as 200, the predictive accuracy slightly differed in one machine. Similarly, among multiple values considered for batch size, hyperparameter tuning suggested using 32 and 64 as the optimal batch sizes.
Time Window: In pursuit of fortifying the model's resilience, in this study we undertook an examination of three discrete time windows. These windows serve as the basis for training the model to forecast values one day ahead. Specifically, the experimentation encompassed time windows of 60, 80, and 120 days, respectively. Notably, the empirical findings underscore the superior performance achieved within the 120-day time window. Consider a scenario where each general stock undergoes trading for 252 days within a calendar year. For this illustration, let us designate the time window as comprising 120 days. In this context, if we utilize data spanning from the 1st day to the 120th day as input, the model will generate a prediction for the closing price on the 121st day. Subsequently, in the next iteration, with data spanning from the 2nd day to the 121st day, the model will furnish a prediction for the closing price on the 122nd day.

Grid search serves as a method for fine-tuning crucial parameters, particularly in the context of hyperparameter optimization. It operates by systematically traversing a predefined grid of hyperparameter combinations in pursuit of the most advantageous set. This method entails exhaustively assessing each combination, rendering it computationally demanding, particularly when dealing with expansive hyperparameter spaces. To ensure thorough evaluation, grid search commonly integrates k-fold cross-validation, dividing the dataset into subsets and scrutinizing each combination across multiple folds. The hyperparameter configuration yielding the highest cross-validation score is deemed the optimal set for the model, furnishing a robust basis for model performance assessment. However, a notable limitation of the grid search technique is its protracted execution duration.

In this context, grid search is specifically employed to identify optimal activation functions and hyperparameters pertaining to various optimizers, batch sizes, epochs, and numbers of units.

3.3. Activation Functions

This study examined three primary activation functions suitable for integration into the proposed hybridized models: (1) ReLU, (2) ELU, and (3) tanh. These three activation functions are mostly used with DL architectures. The choice of activation function depends on the dataset and model architecture. An improper choice might result in sub-optimal performance.

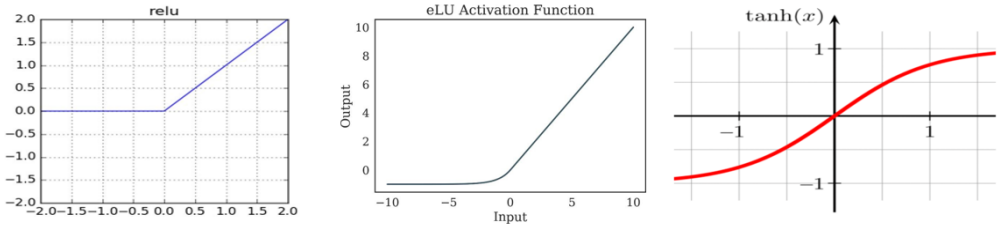
3.3.1. ReLU

The rectified linear unit (ReLU) activation function is extensively utilized in neural networks, providing non-linearity by outputting the input directly for positive values and zero for negative ones. Its popularity stems from its simplicity, computational efficiency, and effectiveness in dealing with the vanishing gradient problem during DNN training. ReLU addresses the issue of vanishing gradients that arise due to the saturation of neurons when sigmoid functions are used. Improper weight initialization can lead to the "dying ReLU" problem causing neurons to die, especially due to negative saturation.

The formula for ReLU is

$$f_{Relu}(h_{i,k}) = max(0, h_{i,k}).$$

Figure 3a presents a graphical representation of ReLU.



(a) ReLU (b) ELU (c) tanh

Figure 3. Activation functions.

3.3.2. ELU

The exponential linear unit (ELU) is an activation function commonly used in artificial neural networks. It brings non-linearity to the network by permitting the inclusion of negative values, unlike traditional activation functions such as ReLU. ELU has a smooth curve for negative inputs, avoiding the issue of dead neurons and enabling better convergence during training. It exhibits the ability to capture information from both positive and negative input ranges, contributing to improved learning in DNN. ELU addresses the issue of the dying ReLU problem that occurs due to negative saturation but bad initialization can still cause underperformance. The ELU activation function is defined as

$$f(k) = \begin{cases} k & \text{if } k > 0 \\ \alpha \cdot (e^k - 1) & \text{if } k \leq 0 \end{cases}.$$

Here, α is a positive hyperparameter controlling the slope of the function for negative inputs. Figure 3b presents a graphical representation of ELU.

3.3.3. Tanh

The hyperbolic tangent function, often denoted as $\tanh(x)$, is a common activation function in neural networks. It squashes input values to the range of $(-1, 1)$, making it zero-centered and aiding in mitigating vanishing gradient problems during training. The tanh activation is particularly useful in scenarios where zero-centered outputs are desired and has applications in various layers of neural network architectures. The tanh activation function delivers output centered around zero, but it can lead to the vanishing gradient problem due to neuron saturation caused by improper weight initialization. The formula for the tanh activation function is

$$\tanh(k) = \frac{e^k - e^{-k}}{e^k + e^{-k}}.$$

Figure 3c presents a graphical representation of tanh.

3.4. Evaluation Metrics

In this research, we have tested the effectiveness of the models from five distinct viewpoints, using five different evaluation metrics. Each of these metrics provides a unique way to test the models.

3.4.1. Mean Squared Error (MSE)

This serves as a commonly employed metric for regression analysis, evaluating the average squared disparity between predicted and actual values. This metric effectively gauges the variability or dispersion of these differences, imposing greater penalties for larger errors. MSE can be mathematically expressed as

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (a_i - \hat{a}_i)^2.$$

Here, a_i is actual and $\hat{a}_i$ is the forecast. N is the dataset size.

3.4.2. Root Mean Squared Error (RMSE)

This serves as a performance metric in regression analysis, measuring the mean measure of discrepancies between forecast and true values. More precisely, it is derived from computing the square root of the MSE. RMSE is given by

$$\text{RMSE} = \sqrt{\text{MSE}}.$$

3.4.3. Mean Absolute Error (MAE)

This serves as a regression measure, quantifying the mean absolute discrepancy between forecast and true values. This evaluation metric offers a straightforward evaluation of the model's effectivity, assigning equal significance to each prediction error. The definition of MAE is

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|.$$

Here, y_i is actual and $\hat{y}_i$ is the forecast. N is the dataset size.

3.4.4. R-Squared (R2) Score

This measures the percentage of variance in the target variable captured by the prediction variables in a regression-based algorithm. The calculation involves comparing the sum of squared differences between forecast and true values to the sum of squared differences between true values and their mean. A higher R-squared score, approaching 1, signifies a superior model fit, suggesting that a larger portion of the target variable's variance is captured by the model. The expression of R-squared is

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(a_i - \hat{a}_i)^2}{\sum_{i=1}^{N}(a_i - \bar{a})^2}.$$

Here, a_i is actual and $\hat{a}_i$ is the forecast. N is the dataset size. $\bar{a}$ is the mean of the real values.

3.4.5. Mean Absolute Percentage Error (MAPE)

This quantifies the mean percent disparity between estimated and true values, offering a metric of the model's effectiveness. A reduced MAPE signifies a closer match between predictions and real outcomes, rendering it an efficient metric for evaluating predictive precision across various industries. The formula for MAPE is

$$MAPE = \frac{1}{n}\sum_{n}\left|\frac{\text{actual} - \text{prediction}}{\text{actual}}\right| \times 100.$$

4. Results and Discussions

4.1. Dataset Consideration

The financial market encompasses a vast array of products derived from various sectors. Each financial product, whether it is a derivative of stocks, cryptocurrencies, etc., reacts differently to distinct market events. Therefore, when developing a predictive model, it is crucial to consider testing under a variety of market conditions with these diverse financial derivatives. In this study, we experimented with three primary basic financial products, from which multiple derivatives are formulated. These include volatile stocks from a range of sectors, the highly volatile cryptocurrency Bitcoin, and the moderately volatile yet stable stock index S&P. These are used for training and testing the proposed models.

To assess the effectiveness of the novel autoencoder-GRU and compare its performance with autoencoder-LSTM, we examined diverse stock datasets spanning various sectors. These include Apple (AAPL) (technology sector), Johnson and Johnson (JNJ) (healthcare

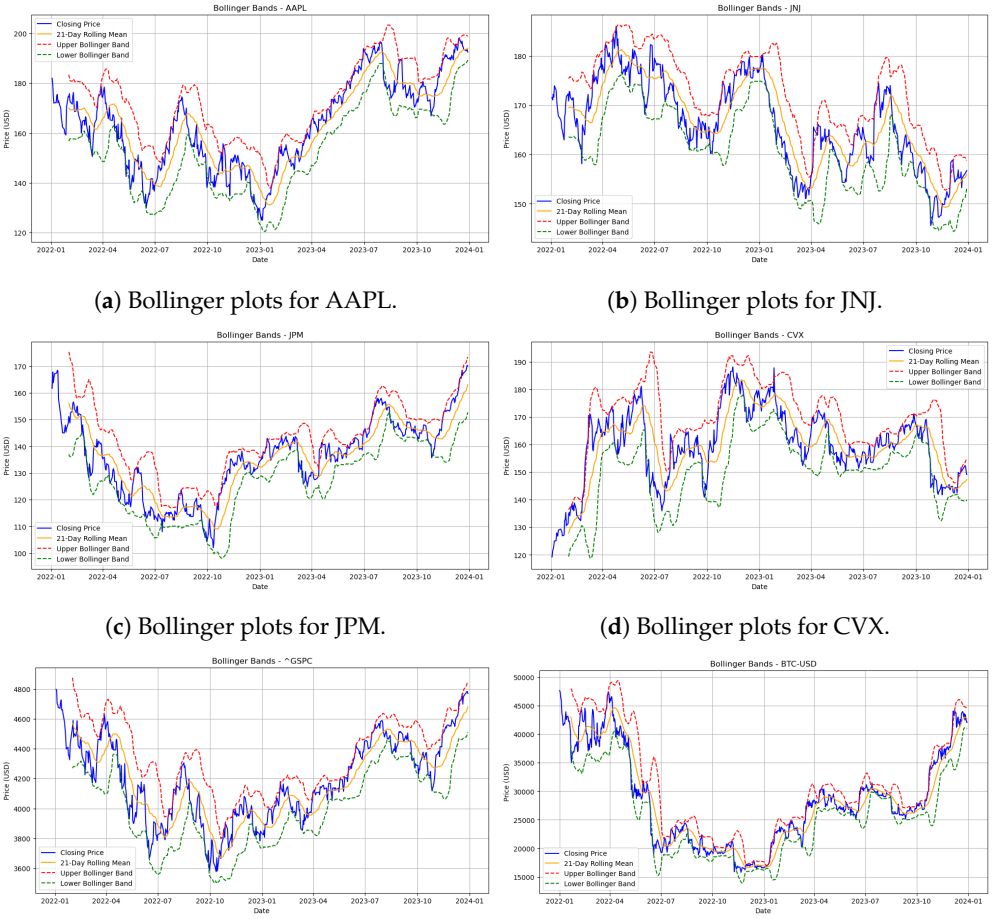
sector), Chevron (CVX) (energy sector), JP Morgan Chase Bank (JPM) (banking sector), Bitcoin (BTC-USD) (cryptocurrency), and the S&P stock index (ĜSPC). The models were trained on data from 1998 to 2022, and their predictive capabilities were evaluated using data from 2023. Additionally, for Bitcoin, the training dataset spans from 2014 to 2022, with the model tested on data from 2023. The dataset analyzed includes significant market occurrences, including the 2001 dot-com bubble burst, the 2008 global financial crisis, and the market crash induced by the COVID-19 pandemic in 2020.

The efficiency of our proposed architecture has been evaluated across multiple machines to analyze its performance under the influence of various machine configurations. These include (a) a Dell Latitude 5540 intel core i7-1355U, 1.70 GHz, with 10 Core(s) and 16 GB RAM; (b) an ASUS VivoBook X515EA intel core i5-1135G7, 2.40 GHz, with 4 Core(s) and 16 GB RAM; (c) DELL Inc. OptiPlex 7040, intel core i5-6600, 3.30 GHz, with 4 core(s) and 16 GB RAM.

All these machines run Windows 11 OS and Jupiter Notebook as the IDE.

4.2. Exploratory Data Analysis

Figure 4 depicts the Bollinger bands for the financial instruments analyzed in this study. Both Figure 4 and Table 2 collectively indicate that AAPL and JPM display elevated volatility, while JNJ demonstrates comparatively lower volatility, and CVX exhibits a moderate level of volatility. S&P registers notably low average volatility, whereas the cryptocurrency Bitcoin experiences the highest volatility between 2022 and 2023 among all the assets considered.



(**a**) Bollinger plots for AAPL.

(**b**) Bollinger plots for JNJ.

(**c**) Bollinger plots for JPM.

(**d**) Bollinger plots for CVX.

(**e**) Bollinger plots for S&P.

(**f**) Bollinger plots for BTC.

Figure 4. Bollinger plots for financial assets from 2022 to 2023.

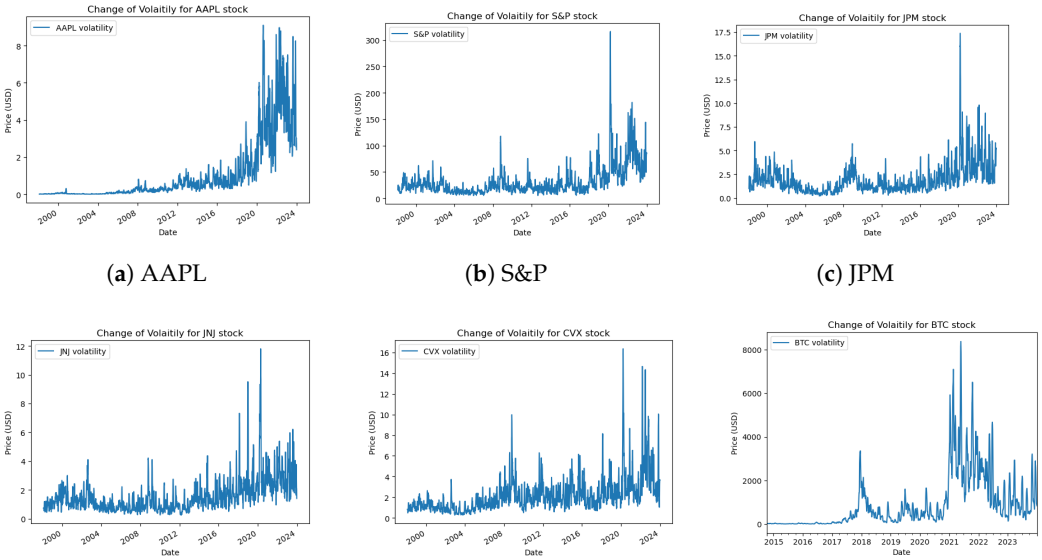
Table 2 illustrates the average volatility metrics derived from a 21-day rolling period for a range of assets. Volatility, within financial contexts, denotes the extent of price fluctuation experienced by an asset over time. Table 2 enumerates various assets, encompassing equities such as Apple, Johnson & Johnson, JP Morgan Chase, and Chevron, alongside the S&P index and the cryptocurrency Bitcoin. Each asset is associated with an average volatility value expressed in percentage terms, reflecting the average daily price oscillation observed over the specified 21-day timeframe.

For instance, Apple exhibits an average volatility of 35.96%, indicating that, on average, its stock price experiences a daily fluctuation of approximately 35.96% during the 21-day period. Similarly, the table furnishes comparable average volatility figures for other assets, facilitating assessments of their respective levels of price variability.

Table 2. Average volatility based on 21 rolling days.

Assets	Average Volatility
Apple	35.96
Johnson & Johnson	17.60
JP Morgan Chase	31.28
Chevron	24.34
S&P	16.71
Bitcoin	52.48

Figure 5 illustrates the change in the volatility trend in the considered timeframe. Notably, Apple stock exhibits a significant rise in volatility. Conversely, Bitcoin's volatility displays abrupt fluctuations. The S&P, reflecting a blend of diverse financial assets, exhibits a high magnitude of volatility but lower average volatility. On the other hand, JPM, JNJ, and CVX consistently experience growth in volatility throughout the years.



(**a**) AAPL (**b**) S&P (**c**) JPM

(**d**) JNJ (**e**) CVX (**f**) BTC-USD

Figure 5. Change in Volatility.

4.3. Predictions by AE-LSTM vs. AE-GRU

This research considered three major activation functions—ReLU, tanh, and ELU. Other hyperparameters include first-moment exponential decay rate β_1, second-moment exponential decay rate β_2, learning rate, and numerical stability. The models take 120 previous time-step data to predict the next step.

4.3.1. Predictions for Individual Stock

The suggested architecture of AE-GRU, employing ReLU as the activation function, outperforms AE-LSTM in the context of stocks AAPL, JNJ, JPM, and CVX. The extensive experimentation encompassed three activation functions and adjustments to hyperparameters, including first-moment exponential decay rate β_1, second-moment exponential decay rate β_2, learning rate, and numerical stability set at 1×10^{-7}. Among various configurations of AE-LSTM, the setup with the first-moment exponential decay rate, β_1 at 0.9, the second-moment exponential decay rate β_2 at 0.999, the learning rate as 0.001, numerical stability at 1×10^{-7}, and ELU as the activation function proves to be the most effective. However, consistently, the proposed AE-GRU architecture with ReLU as the activation function outperforms AE-LSTM in these assessments. Figure 6 visually demonstrates the superiority in providing an accurate prediction of stock trends of AE-GRU compared to AE-LSTM-based predictions. Furthermore, Table 3 provides evidence that the superiority of AE-GRU over AE-LSTM is affirmed by five evaluation metrics.

Table 3. Evaluation of the performance of AE-LSTM and AE-GRU through metrics.

Components	Metric	Algorithms	
		AE-GRU	**AE-LSTM**
AAPL	MSE	0.37	1.22
	RMSE	0.61	1.10
	MAE	0.53	0.97
	R-squared	0.999	0.996
	MAPE	0.3%	0.6%
JNJ	MSE	0.01	0.07
	RMSE	0.12	0.26
	MAE	0.1	0.2
	R-squared	0.9997	0.9987
	MAPE	0.07%	0.12%
JPM	MSE	0.33	4.79
	RMSE	0.58	2.18
	MAE	0.55	2.17
	R-squared	0.996	0.94
	MAPE	0.4%	1.51%

Table 3 presents performance metrics for the proposed novel architectures, AE-GRU and AE-LSTM, across various components such as AAPL, JNJ, and JPM. The metrics evaluated include MSE, RMSE, MAE, R-squared coefficient, and MAPE.

For the component AAPL:

- AE-GRU exhibits lower MSE, RMSE, MAE, and MAPE values compared to AE-LSTM, indicating superior performance in predicting AAPL's closing prices.
- Additionally, AE-GRU achieves higher R-squared values, signifying better goodness-of-fit between predicted and actual values for AAPL.

Similarly, for the components JNJ and JPM:

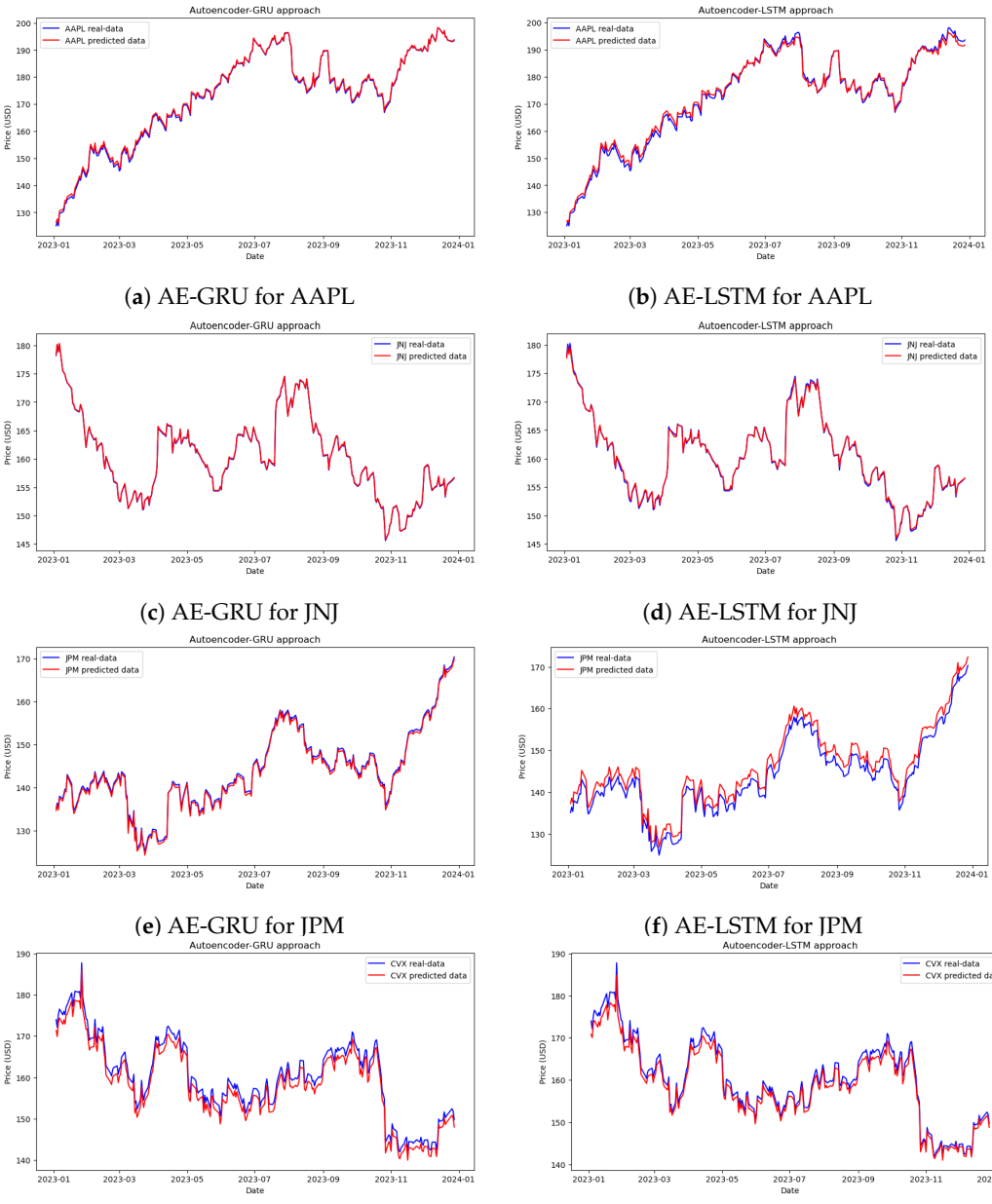
- AE-GRU consistently outperforms AE-LSTM across all metrics, demonstrating its efficacy in predicting closing prices for both JNJ and JPM.
- Notably, AE-GRU achieves higher R-squared values, indicating stronger linear relationships between predicted and actual values for JNJ and JPM.

Despite the similar heightened volatility, as observed in Table 2, and strong correlations between technology and banking stocks, AAPL and JPM exhibited observable variations in the evaluation metrics, as observed in Table 3. This discrepancy arises due to differences in the rates of volatility change between AAPL and JPM, as depicted in Figure 5a,c. While AAPL demonstrates a low magnitude of volatility change, it experiences a rapid increase. Conversely, JPM showcases a low magnitude of change with minimal fluctuations in

volatility. Varying rates of volatility disrupt the performance of the AE-LSTM model to a certain extent but the AE-GRU model shows robustness in such cases.

Thus, for general stocks of different categories, such as technology, banking, healthcare, and energy, it can be observed that the proposed models perform with higher accuracy. In every scenario, AE-GRU outperforms the AE-LSTM.

To our knowledge, such performance efficacy cannot be observed in any baseline models such as vanilla, LSTM, or GRU Pirani et al. (2022) Salimath et al. (2021).



(**a**) AE-GRU for AAPL (**b**) AE-LSTM for AAPL

(**c**) AE-GRU for JNJ (**d**) AE-LSTM for JNJ

(**e**) AE-GRU for JPM (**f**) AE-LSTM for JPM

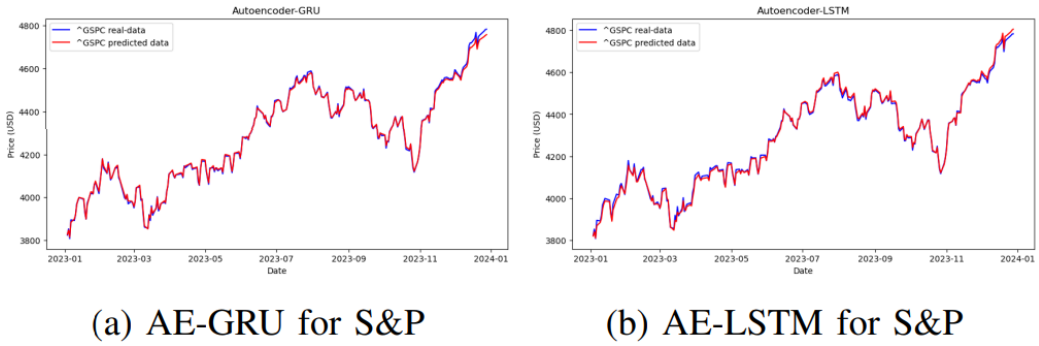
(**g**) AE-GRU for CVX (**h**) AE-LSTM for CVX

Figure 6. Trend prediction by AE-GRU and AE-LSTM for individual stocks.

4.3.2. Predictions for the Stock Index S&P

Hyperparameter tuning for AE-GRU involves adjusting key parameters, such as setting the first-moment and second-moment exponential decay rates β_1 and β_2 to 0.9 and 0.999, respectively, the learning rate to 0.001, and maintaining numerical stability at 1×10^{-7}. Specifically, for stock indices like S&P, AE-GRU performs optimally when using `tanh` as the activation function. In contrast, AE-LSTM with the `ReLU` activation function

exhibits favorable outcomes compared to its other variants. However, in direct comparison, the `tanh`-based AE-GRU outperforms all variations of AE-LSTM in forecasting annual prices of the S&P, as depicted in Figure 7. Despite a slightly higher MSE, attributed to the large scale of values, the proposed AE-GRU architecture significantly outperforms AE-LSTM, with an impressively low MAPE of 0.1%. Consequently, for stock index prediction, the proposed AE-GRU-based architecture demonstrates superiority over AE-LSTM, which is illustrated in Figure 7.

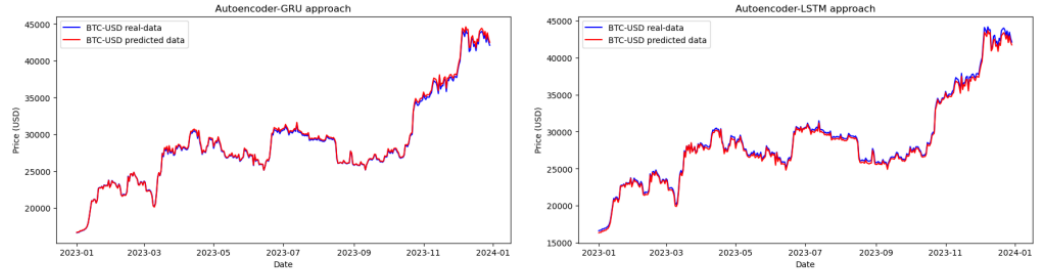


(a) AE-GRU for S&P (b) AE-LSTM for S&P

Figure 7. Trend prediction by AE-GRU and AE-LSTM for stock index.

4.3.3. Predictions for the Cryptocurrency Bitcoin

Figure 8 demonstrates that the proposed AE-GRU design, utilizing ReLU as the activation function, outperforms the AE-LSTM prediction. Although hyperparameter tuning, employing ELU as the activation function, enhanced the predictive accuracy of the AE-LSTM structure to some degree, it still fell short of surpassing the novel AE-GRU design. The proposed AE-GRU architecture exhibited superior performance to the AE-LSTM architecture, achieving an MAPE of 0.5%, which is half the value observed in AE-LSTM.



(a) AE-GRU for Bitcoin (b) AE-LSTM for Bitcoin

Figure 8. Trend prediction by AE-GRU and AE-LSTM for cryptocurrency.

5. Practical Applications

Cakici et al. (2024) revisited evidence on market risk premium predictability by equity anomalies, extending findings internationally and across different anomaly sets, finding that anomalies fail to predict market excess returns due to methodological choices impacting prediction performance. Similarly, our research could be extended in this direction to predict different economic factors such as risk premiums by incorporating features such as market anomalies. The novel encoder–decoder models developed in this study, incorporating LSTM and GRU layers, AE-LSTM and AE-GRU, demonstrate efficacy in predicting various categories of stocks, cryptocurrencies, and stock indices. These models can be an important tool to assist traders and investors in making informed decisions regarding their investment portfolios. By leveraging the model's predictions, traders and investors can make informed decisions regarding their investment portfolios. Furthermore, the insights provided by the model can be utilized to formulate trading strategies based on simple

indicators such as shape ratios and other technical analysis tools. These strategies can help traders identify optimal entry and exit points, manage risk more effectively, and potentially enhance their overall trading performance. Overall, the model's predictions have the potential to contribute to more informed and data-driven decision-making in financial markets, leading to improved trading outcomes. Moreover, considering transaction costs is crucial in assessing the economic gains derived from trading strategies based on predictive models. Integrating transaction costs into the evaluation of trading strategies is essential for ensuring their viability in real-world trading environments. The inclusion of features in the ML models increases learning capacity. Thus, adding transaction costs as a new feature might enhance the model's performance positively in practical scenarios allowing an increase in the efficacy of the model in predicting buy and sell signals more robustly. The addition of such features in the model in such scenarios requires careful experimentation, which will be a nice future extension of this research.

Predicting buy and sell signals based on the prediction of highly volatile market scenarios using the models proposed will be a major direction of this research. This research will also be extended to construct futuristic portfolios based on the prediction to ensure maximized profit under varying market conditions.

6. Conclusions

The proposed architecture, AE-GRU, offers an innovative approach by combining an encoder–decoder architecture and GRU, demonstrating superior performance compared to the proposed AE-LSTM across various market scenarios. We have conducted experiments using stocks from different sectors and also included S&P as the stock index and Bitcoin as a cryptocurrency. Across all sectors, the proposed novel encoder–decoder architecture, AE-GRU, consistently outperformed the AE-LSTM. A novel AE-LSTM architecture is also designed in this study, which improved the existing idea of the LSTM and autoencoder combination through rigorous hyperparameter tuning. Despite experimenting with hyperparameter tuning for AE-LSTM and improving its performance from the existing version, it still could not match the prediction accuracy of the proposed novel AE-GRU architecture. Experiments were organized on three different computing devices, yielding similar results with slight variations in the values of evaluation metrics due to randomized weight initialization. We have considered training data from significant market events of the 2001 high-tech bubble burst, the 2008 global financial crisis, and the 2020 COVID-19 stock market crisis. Our findings advocate for the adoption of the novel AE-GRU architecture, showcasing enhanced predictive accuracy for financial assets across various sectors and market conditions. The proposed encoder–decoder-based GRU architecture retains the long-term non-linear contexts from the past temporal observations, resulting in superior efficacy in predicting future values for stocks, indexes, and highly volatile cryptocurrencies. After significant enhancements were made to the AE-LSTM architecture, the newly proposed AE-GRU architecture demonstrated superior performance. This research proposes two novel encoder–decoder architectures, AE-LSTM and AE-GRU, that provide higher predictive accuracy than the existing models.

It is imperative to recognize certain limitations inherent in our research endeavor. While we have successfully introduced novel architectures aimed at augmenting the predictive accuracy of diverse financial assets, our focus has been primarily on the model development. Moving forward, it is essential to explore the integration of various learnable filters to fortify the robustness of our models. Additionally, investigating the feasibility of generating trade signals from our predictions, particularly in volatile market conditions, holds promise for enhancing the practical utility of our models in real-world economic contexts. These avenues for future research underscore the dynamic nature of financial forecasting and the ongoing quest for ever-improving predictive models.

7. Future Work

This research observes substantial efficacy after using an encoder–decoder-based architecture with RNN algorithms. Moreover, the proposed model takes much less time to train than the TN architectures with a similar amount of data. TN architectures also utilize encoder–decoder architectures but with multi-head attention. They do not learn from the compressed latent representation from the input vectors. Further research can conduct a comparative analysis between the TN and the proposed architectures by extensive investigation. Moreover, the proposed model could be made more robust by adding different types of learnable convolutional filters to conduct more specific feature extraction. Other than that, context retention in the model through a model-agnostic vectorized representation of the data before feeding it into the model could be an interesting addition to the presently proposed research. Also, ML algorithms are generally sensitive to a lot of parameters. Our purpose in this study was to hyperparameter-tune the model in such a way that the proposed novel AE-GRU shows less variation under various circumstances. In our further research endeavor, we aim to extend our investigation to conduct detailed sensitivity analysis of different parameters under various market conditions. Sensitivity analysis could certainly be a new focus of research. Moreover, new features such as transaction costs and others can be included to expand the applicability of the models in practical trading scenarios. To address machine dependencies, thorough hyperparameter tuning is recommended, controlling features like weight optimization, necessitating experimentation with high-performance computing, which we plan to consider in the future.

Author Contributions: Conceptualization, J.D.D., R.K.T. and A.T.; methodology, J.D.D., R.K.T. and C.H.; software, J.D.D. and R.K.T.; validation, J.D.D., R.K.T. and C.H.; formal analysis, J.D.D., R.K.T., C.H. and A.T.; investigation, J.D.D., R.K.T. and A.T.; resources, R.K.T.; data curation, J.D.D. and R.K.T.; writing—original draft preparation, J.D.D. and R.K.T.; writing—review and editing, J.D.D., R.K.T. and C.H.; visualization, J.D.D. and R.K.T.; supervision, R.K.T.; project administration, R.K.T. and A.T.; funding acquisition, R.K.T. All authors have read and agreed to the published version of the manuscript.

Funding: The first author, Joy Dip Das was funded by the University of Manitoba Graduate Fellowship (UMGF) and Graduate Enhancement of Tri-agency Stipends (GETS) from the Faculty of Graduate Studies, University of Manitoba matching the supervisor Thulasiram's financial support for Joy's MSc thesis program. The principal investigator of this research project, Thulasiram was supported by the Discovery Grant from the Natural Sciences and Engineering Research Council (NSERC) Canada.

Data Availability Statement: The algorithms, the codes designed & developed, and the data used & generated in this study will be made available upon request.

Acknowledgments: The first author acknowledges the University of Manitoba Graduate Fellowship (UMGF) and Graduate Enhancement of Tri-agency Stipends (GETS) from the Faculty of Graduate Studies, University of Manitoba. The second and fourth authors acknowledge the Discovery Grant from the Natural Sciences and Engineering Research Council (NSERC) Canada.

Conflicts of Interest: Authors declare no conflict of interests among themselves and with granting agencies.

Appendix A

Appendix A.1. LSTM

LSTM is an RNN architecture that can handle long-term dependencies in the data. Graves and Graves (2012) address the issue of vanishing gradients faced by the basic RNN. From Figure A1, we can see that it comprises specialized memory units that can maintain and update information over extended sequences[2]. These memory units, called cells, contain three gating mechanisms: input, output, and forget gates. The input gate oversees the selection of data to be stored in the cell, the forget gate governs the determination of

information to be discarded, and the output gate manages the information to be emitted from the cell.

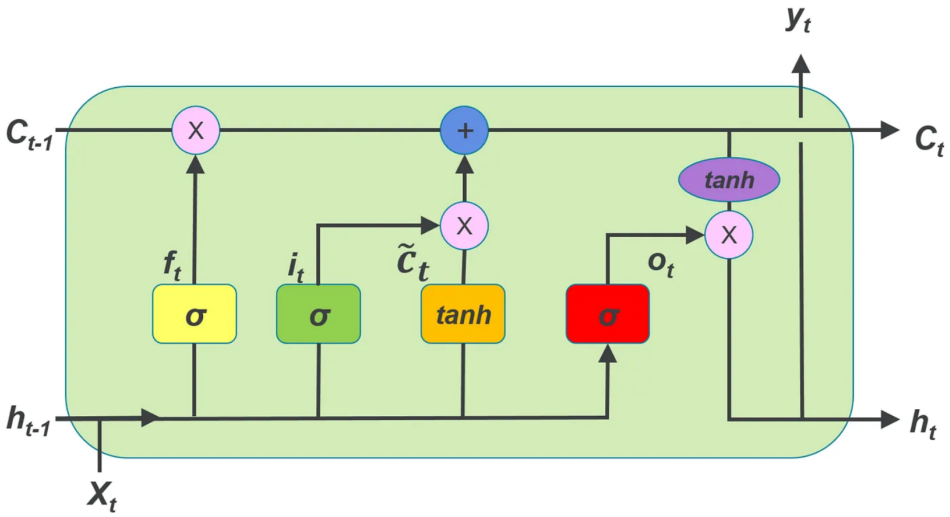


Figure A1. LSTM architecture.

The LSTM equations:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t \odot \tanh(C_t)
\end{aligned}
$$

f_t is the forget gate, governing the information to be omitted from the cell state. i_t denotes the input gate, deciding which values to revise in the cell state. $\tilde{C}_t$ signifies the candidate cell state, presenting new candidate values for the cell state. C_t represents the cell state, storing long-term information. o_t indicates the output gate, managing the information for output from the cell state. h_t corresponds to the hidden state, encompassing the short-term information for output.

Appendix A.2. GRU

The GRU, a prevalent RNN algorithm in stock price prediction, exhibits a unique architecture with gating mechanisms regulating information flow, enhancing long-term dependency capture in input data[3]. Core components, including reset and update gates, enable effective context utilization, as seen in Figure A2. GRU's computational efficiency, attributed to its simpler two-gate structure, contributes to faster training times compared to LSTM (see Yang et al. (2020)).

The following equations explain GRU:

$$
\begin{aligned}
r_t &= \sigma(W_r x_t + U_r h_{t-1}) \\
z_t &= \sigma(W_z x_t + U_z h_{t-1}) \\
\tilde{h}_t &= \tanh(r_t \odot U h_{t-1} + W x_t) \\
h_t &= (1 - z_t) \odot \tilde{h}_t + z_t \odot h_{t-1}
\end{aligned}
$$

In the context of the GRU model, r_t serves as the reset gate, while z_t functions as the update gate. The vector $\tilde{h}_t$ represents a candidate activation vector. The functions σ and tanh

denote the sigmoid and hyperbolic tangent functions, respectively. The weight matrices W_r, U_r, W_z, U_z, W, and U play pivotal roles in the model.

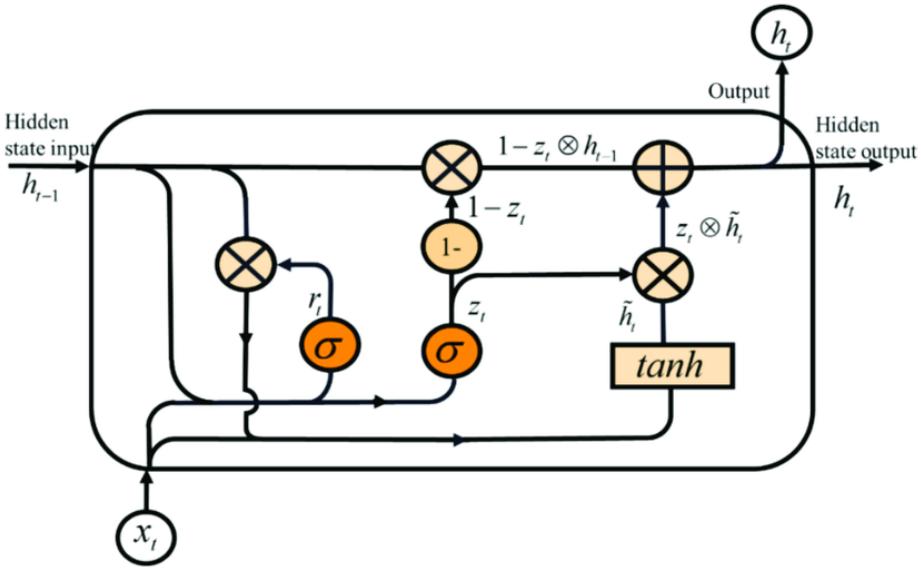


Figure A2. GRU architecture.

LSTMs employ a more intricate structure, with separate gates for input, forget, and output, enabling them to preserve long-term dependencies in the input sequence by controlling the flow of information. In contrast, GRUs feature a simplified architecture by merging the functionality of the input and forget gates into a single "update gate", resulting in fewer parameters and computational resources. While LSTMs maintain separate cell states and hidden states for memory management, GRUs combine them into a single vector, which may slightly reduce memory capacity. Both LSTM and GRU units have demonstrated effectiveness in various applications, and the choice between them often depends on factors such as model complexity and computational resources.

Notes

1 Deep learning a subset of the field of machine learning. In this paper, unless stated otherwise, we will use the term machine learning to refer to traditional (such as shallow or non-deep learning alternatives) machine learning methods, and deep learning to refer to more recent models with many layers, large number of parameters, or both.

2 https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af, accessed on: 2 March 2024.

3 https://www.researchgate.net/figure/GRU-structure-diagram_fig2_355705028, accessed on 2 March 2024.

References

Ali, Muhammad, Dost Muhammad Khan, Huda M. Alshanbari, and Abd Al-Aziz Hosni El-Bagoury. 2023. Prediction of complex stock market data using an improved hybrid emd-lstm model. *Applied Sciences* 13: 1429. [CrossRef]

Banerjee, Debadrita. 2014. Forecasting of indian stock market using time-series arima model. Paper presented at the 2014 2nd IEEE International Conference on Business and Information Management (ICBIM), Durgapur, India, January 9–11., pp. 131–35.

Cakici, Nusret, Christian Fieberg, Daniel Metko, and Adam Zaremba. 2024. Do anomalies really predict market returns? New data and new evidence. *Review of Finance* 28: 1–44. [CrossRef]

Cakra, Yahya Eru, and Bayu Distiawan Trisedya. 2015. Stock price prediction using linear regression based on sentiment analysis. Paper presented at the 2015 International Conference on Advanced Computer Science and Information Systems (ICACSIS), Depok, Indonesia, October 10–11. pp. 147–154. [CrossRef]

Chen, Junwei. 2023. Analysis of bitcoin price prediction using machine learning. *Journal of Risk and Financial Management* 16: 51. [CrossRef]

Chen, Luyang, Markus Pelger, and Jason Zhu. 2024. Deep learning in asset pricing. *Management Science* 70: 714–50. [CrossRef]

Fanai, Hosein, and Hossein Abbasimehr. 2023. A novel combined approach based on deep autoencoder and deep classifiers for credit card fraud detection. *Expert Systems with Applications* 217: 119562. [CrossRef]

Fang, Zhen, Xu Ma, Huifeng Pan, Guangbing Yang, and Gonzalo R. Arce. 2023. Movement forecasting of financial time series based on adaptive lstm-bn network. *Expert Systems with Applications* 213: 119207. [CrossRef]
Gradxs, Govind Prasad Buddha, and Nagamalleswara Rao. 2023. Behaviour based credit card fraud detection: Design and analysis by using deep stacked autoencoder based harris grey wolf (hgw) method. *Scandinavian Journal of Information Systems* 35: 1–8.
Graves, Alex, and Alex Graves. 2012. Long short-term memory. In *Supervised Sequence Labelling with Recurrent Neural Networks*; Berlin and Heidelberg: Springer, vol. 385, pp. 37–45.
Gu, Shihao, Bryan Kelly, and Dacheng Xiu. 2021. Autoencoder asset pricing models. *Journal of Econometrics* 222: 429–50. [CrossRef]
Gupta, Punit, Ya Gao, Rong Wang, and Enmin Zhou. 2021. Stock prediction based on optimized lstm and gru models. *Scientific Programming* 2021: 4055281. [CrossRef]
Hindrayani, Kartika Maulida, Tresna Maulana Fahrudin, R. Prismahardi Aji, and Eristya Maya Safitri. 2020. Indonesian stock price prediction including covid19 era using decision tree regression. Paper presented at the 2020 3rd IEEE International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, December 10, pp. 344–47.
Hiransha, Ma, E. Ab Gopalakrishnan, Vijay Krishna Menon, and K. P. Soman. 2018. Nse stock market prediction using deep-learning models. *Procedia Computer Science* 132: 1351–62.
Hossain, Mohammad Asiful, Rezaul Karim, Ruppa Thulasiram, Neil D. B. Bruce, and Yang Wang. 2018. Hybrid deep learning model for stock price prediction. Paper presented at the 2018 IEEE Symposium on Computational Intelligence in Financial Engineering and Economics (CIFEr), Symposium Series on Computational Intelligence, Bangalore, India, November 18–21. pp. 1837–44. [CrossRef]
Hu, Zexin, Yiqi Zhao, and Matloob Khushi. 2021. A survey of forex and stock price prediction using deep learning. *Applied System Innovation* 4: 9. [CrossRef]
Jiang, Qiang, Chenglin Tang, Chen Chen, Xin Wang, and Qing Huang. 2019. Stock price forecast based on LSTM neural network. Paper presented at the Twelfth International Conference on Management Science and Engineering Management, Ontario, ON, Canada, August 5–8. New York: Springer International Publishing, pp. 393–408. [CrossRef]
Jiang, Weiwei. 2021. Applications of deep learning in stock market prediction: Recent progress. *Expert Systems with Applications* 184: 115537. [CrossRef]
Karmiani, Divit, Ruman Kazi, Ameya Nambisan, Aastha Shah, and Vijaya Kamble. 2019. Comparison of predictive algorithms: Backpropagation, SVM, LSTM and Kalman filter for stock market. Paper presented at the 2019 Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates, February 4–6. pp. 228–34. [CrossRef]
Khedr, Ahmed M., Ifra Arif, Magdi El-Bannany, Saadat M. Alhashmi, and Meenu Sreedharan. 2021. Cryptocurrency price prediction using traditional statistical and machine-learning techniques: A survey. *Intelligent Systems in Accounting, Finance and Management* 28: 3–34. [CrossRef]
Kim, Jongyeop, Seongsoo Kim, Hayden Wimmer, and Hong Liu. 2021. A cryptocurrency prediction model using lstm and gru algorithms. Paper presented at the 2021 IEEE/ACIS 6th International Conference on Big Data, Cloud Computing, and Data Science (BCD), Zhuhai, China, September 13–15, pp. 37–44.
Kohli, Pahul Preet Singh, Smriti Srivastava, Yog Raj Sood, and Aamir Ahmad. 2019. Stock prediction using machine learning algorithms. In *Applications of Artificial Intelligence Techniques in Engineering*. Singapore: Springer, pp. 1–38. [CrossRef]
Kumbure, Mahinda Mailagaha, Christoph Lohrmann, Pasi Luukka, and Jari Porras. 2022. Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications* 197: 116659. [CrossRef]
Kwak, Nae Won, and Dong Hoon Lim. 2021. Financial time series forecasting using adaboost-gru ensemble model. *Journal of the Korean Data And Information Science Society* 32: 267–81. [CrossRef]
Lawal, Zaharaddeen Karami, Hayati Yassin, and Rufai Yusuf Zakari. 2020. Stock market prediction using supervised machine learning techniques: An overview. Paper presented at the 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), Gold Coast, Australia, December 16–18, pp. 1–6.
Liou, Cheng-Yuan, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. 2014. Autoencoder for words. *Neurocomputing* 139: 84–96. [CrossRef]
Lu, Xugang, Yu Tsao, Shigeki Matsuda, and Chiori Hori. 2013. Speech enhancement based on deep denoising autoencoder. *Interspeech* 2013: 436–40.
McMillan, David G. 2003. Non-linear predictability of uk stock market returns. *Oxford Bulletin of Economics and Statistics* 65: 557–73. [CrossRef]
Metghalchi, Massoud, Juri Marcucci, and Yung-Ho Chang. 2012. Are moving average trading rules profitable? evidence from the european stock markets. *Applied Economics* 44: 1539–59. [CrossRef]
Nikou, Mahla, Gholamreza Mansourfar, and Jamshid Bagherzadeh. 2019. Stock price prediction using deep learning algorithm and its comparison with machine learning algorithms. *Intelligent Systems in Accounting, Finance and Management* 26: 164–74. [CrossRef]
Petrovic, Aleksandar, Ivana Strumberger, Timea Bezdan, Hothefa Shaker Jassim, and Said Suleiman Nassor. 2021. Cryptocurrency price prediction by using hybrid machine learning and beetle antennae search approach. Paper presented at the 2021 29th IEEE Telecommunications Forum (TELFOR), Belgrade, Serbia, November 23–24, pp. 1–4.
Pierre, Agbessi Akuété, Salami Adekunlé Akim, Agbosse Kodjovi Semenyo, and Birregah Babiga. 2023. Peak electrical energy consumption prediction by arima, lstm, gru, arima-lstm and arima-gru approaches. *Energies* 16: 4739. [CrossRef]

Pintelas, Emmanuel, Ioannis E Livieris, Stavros Stavroyiannis, Theodore Kotsilieris, and Panagiotis Pintelas. 2020. Investigating the problem of cryptocurrency price prediction: a deep learning approach. In *Artificial Intelligence Applications and Innovations: Proceedings, Part II 16, 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, 5–7 June 2020*. Berlin: Springer, pp. 99–110.

Pirani, Muskaan, Paurav Thakkar, Pranay Jivrani, Mohammed Husain Bohara, and Dweepna Garg. 2022. A comparative analysis of ARIMA, GRU, LSTM and BiLSTM on financial time series forecasting. Paper presented at the 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), Ballari, India, April 23–24, pp. 1–6. [CrossRef]

Qi, Chenyang, Jiaying Ren, and Jin Su. 2023. Gru neural network based on ceemdan–wavelet for stock price prediction. *Applied Sciences* 13: 7104. [CrossRef]

Rubell, Marion Lincy G., Nevin Selby, Aditya Taparia. 2023. An efficient stock price prediction mechanism using Multivariate Sequential LSTM Autoencoder. Preprint, 21 February 2023. Version 1. Available online. https://www.researchsquare.com/article/rs-2599921/v1 (accessed on 2 March 2024).

Saad, Omar M., and Yangkang Chen. 2020. Deep denoising autoencoder for seismic random noise attenuation. *Geophysics* 85: V367–76. [CrossRef]

Salimath, Shwetha, Triparna Chatterjee, Titty Mathai, Pooja Kamble, and Megha Kolhekar. 2021. Prediction of stock price for indian stock market: A comparative study using lstm and gru. In *Advances in Computing and Data Sciences: Part II 5, 5th International Conference, ICACDS 2021, Nashik, India, 23–24 April 2021, Revised Selected Papers*. Berlin: Springer, pp. 292–302.

Shahi, Tej Bahadur, Ashish Shrestha, Arjun Neupane, and William Guo. 2020. Stock price forecasting with deep learning: A comparative study. *Mathematics* 8: 1441. [CrossRef]

Shen, Guizhu, Qingping Tan, Haoyu Zhang, Ping Zeng, and Jianjun Xu. 2018. Deep learning with gated recurrent unit networks for financial sequence predictions. *Procedia Computer Science* 131: 895–903. [CrossRef]

Sheth, Dhruhi, and Manan Shah. 2023. Predicting stock market using machine learning: Best and accurate way to know future stock prices. *International Journal of Systems Assurance Engineering and Management* 14: 1–18. [CrossRef]

Sirisha, Uppala Meena, Manjula C Belavagi, and Girija Attigeri. 2022. Profit prediction using arima, sarima, and lstm models in time series forecasting: A comparison. *IEEE Access* 10: 124715–27. [CrossRef]

Song, Hyunsun, and Hyunjun Choi. 2023. Forecasting stock market indices using recurrent neural network based hybrid models: Cnn-lstm, gru-cnn, and ensemble models. *Applied Sciences* 13: 4644. [CrossRef]

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. New York: Curran Associates, Inc., vol. 30, pp. 1–11. Available online: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. (accessed on 2 March 2024).

Xia, Kun, Jianguang Huang, and Hanyu Wang. 2020. Lstm-cnn architecture for human activity recognition. *IEEE Access* 8: 56855–66. [CrossRef]

Yang, Shudong, Xueying Yu, and Ying Zhou. 2020. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. Paper presented at the 2020 IEEE International Workshop on Electronic Communication and Artificial Intelligence (IWECAI), Shanghai, China, June 12–14. pp. 98–101.

Zhao, Heng, Xinran Li, Jiakai Xu, Xianghua Fu, and Jiehao Chen. 2023. Financial time series data prediction by combination model Adaboost-KNN-LSTM. Paper presented at the 2023 IEEE International Joint Conference on Neural Networks (IJCNN), Gold Coast, Australia, June 18–23, pp. 1–8. [CrossRef]

Zhong, Xiao, and David Enke. 2017. Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications* 67: 126–39. [CrossRef]

Zhou, Xianzheng, Hui Zhou, and Huaigang Long. 2023. Forecasting the equity premium: Do deep neural network models work? *Modern Finance* 1: 1–11. [CrossRef]