The object of this research is a deep learning model based on recurrent neural network (RNN), long short-term memory (LSTM), and transformer, applied to predict financial asset prices using historical time series data. The main problem addressed is the absence of a systematic study evaluating the combined effect of weight initialization methods and activation functions in time-series prediction models, particularly regarding convergence speed, prediction accuracy, and the model's ability to capture price variability. The results show that RNN and LSTM have better training stability, are able to converge in one epoch, and provide high prediction performance (RMSE < 3.7, MAPE < 0.015, R^2 close to 0.9999). In contrast, Transformer showed lower prediction performance (RMSE around 37, MAPE around 0.58, R^2 between 0.9884-0.9885) and tended to overfitting on various strategy combinations. In RNN and LSTM models, the All-Zeros (AZ) and ReLU combination specifically degrades stability and leads to overfitting. The superiority of RNN and LSTM is attributed to their sequential architectures, which are more effective at learning short-term temporal patterns and more robust to suboptimal weight initialization. Therefore, selecting an appropriate combination of weight initialization and activation function plays a key role in enhancing model performance. These findings contribute empirical evidence to the importance of configuration choices in deep learning for time-series forecasting. The results can be applied in the development of deep learning-based financial asset prediction and recommendation systems, particularly for assets with long historical records and volatile market conditions

***Keywords:** deep learning, asset financial prediction, weight initialization, financial time series*

UDC 004.8
DOI: 10.15587/1729-4061.2025.326913

EVALUATING THE IMPACT OF WEIGHT INITIALIZATION ON RECURRENT AND TRANSFORMER-BASED MODELS IN FINANCIAL ASSET PRICE PREDICTION

Andri
Doctoral Student of Computer Science*,
Lecturer of Computer Science
Department of Computer Science
Universitas Mikroskil
Thamrin str., 112, Medan, Indonesia, 20212

Tengku Henny Febriana Harumy
Corresponding Author
Doctor of Computer Science*
E-mail: hennyharumy@usu.ac.id

Syahril Efendi
Doctor of Mathematics, Professor*
*Department of Computer Science
Universitas Sumatera Utara
Dr. T. Mansur str., 9, Medan, Indonesia, 20155

Received 05.03.2025
Received in revised form 02.05.2025
Accepted 24.05.2025
Published 30.08.2025

***How to Cite:** Andri, A., Harumy, T. H. F., Efendi, S. (2025). Evaluating the impact of weight initialization on recurrent and transformer-based models in financial asset price prediction. Eastern-European Journal of Enterprise Technologies, 4 (4 (136)), 19–31. https://doi.org/10.15587/1729-4061.2025.326913*

1. Introduction

Asset price forecasting is an important part of investment decision-making, as it helps identify the optimal time to buy or sell a financial asset (e.g. stocks) [1]. Along with the importance of this role, the examination of asset pricing has emerged as one of the most captivating subjects in the field of finance [2]. As artificial intelligence technologies advance, especially in the fields of machine learning and deep learning, AI-driven methods are increasingly significant in tackling the intricacies of evolving financial data. This is evident from the increasing application of AI models in asset pricing studies, which have demonstrated their ability to handle interactions between market factors and improve the accuracy of asset price predictions [3]. Other studies have also shown that machine learning models have been effectively used to detect complex patterns in various domains, such as health risk classification systems based on environmental and historical data [4].

Recurrent neural network (RNN) models and their derivatives, such as long short-term memory (LSTM), have been widely used in financial time-series analysis due to their ability to capture temporal patterns [5]. Recently, large language models (LLMs) based on the transformer architecture have demonstrated the capability to enhance the ability to identify patterns in financial sequences [6–8]. The success of deep learning models is not only determined by the model architecture, but also greatly influenced by the weight initialization strategy. An appropriate strategy can speed up convergence, mitigate gradient vanishing/exploding problems, and improve the accuracy and ability of the model to effectively represent data patterns. Conversely, inappropriate initialization can slow down training and degrade the predictive quality of the model [5, 9]. In another context, the use of neural networks for predictive analysis based on historical data has also been applied to the study of mapping dengue risk areas, although it has not utilized sequential architectures such as RNN or LSTM [10]. In addition, metaheuristic-based deep learning architecture has also been developed to improve training stability and overall model performance [11].

In line with this direction, asset price forecasting studies have applied various machine learning and deep learning approaches, which generally emphasize feature selection and model architecture development. For example, the study by [12] used ESG indices and SHAP values to improve the interpretability of clean energy price predictions. The study by [13] developed a multiscale deep learning architecture (ESTA-Net) to extract complex behavioral patterns of stock prices using attention and directional regularization. The study by [14] focused on feature engineering through the formation of derived indices to improve prediction accuracy using multi-layer perceptron (MLP). Meanwhile, study [15] applied LSTM model to represent the nonlinear structure in stock pricing based on five fundamental factors. Other recent studies, such as [16], proposed a GA-Attention-Fuzzy-Stock-Net hybrid architecture that integrates fuzzy logic, attention mechanism, and genetic algorithm to optimize the architecture configuration and temporal feature selection. In addition, a study by [17] showed the application of LSTM and RNN models in the context of short-term load forecasting (STLF), which evaluated a combination of tanh, ReLU, and LeakyReLU activation functions. Although the focus was on the energy sector, the study showed that the choice of activation function and feature selection greatly influenced the prediction performance. However, the weight initialization strategy was not explicitly analyzed.

Besides the weight initialization strategy, the choice of activation function is also a crucial factor that determines the performance of deep learning models in processing time-series data. The effectiveness of activation functions in time-series data processing is also reinforced by the study of [18], which showed that tanh produced the lowest prediction error in multilayer perceptron (MLP) experiments for non-linear time series forecasting as well as two real datasets. This finding provides additional justification for the choice of activation function used in this study.

Therefore, studies that are specifically devoted to exploring the influence of weight initialization strategies and activation functions on the performance of deep learning models in financial time-series prediction are of high scientific relevance. This relevance is further underscored by the growing need for predictive models that are stable, accurate, and efficient in financial domains that are highly sensitive to estimation errors. Accordingly, this topic deserves further investigation in the context of developing reliable deep learning-based forecasting models.

2. Literature review and problem statement

Previous studies have examined the effect of weight initialization strategies on the performance of deep learning models in various domains. These studies generally show that an appropriate weight initialization method can accelerate convergence, avoid vanishing or exploding gradients, and improve model accuracy. A study by [19] showed that transferring weights from similar domains significantly improved the performance of plant disease classification. This study tested six initialization methods on eight modern deep learning architectures, including CNN and Transformer, and found that domain transfer-based approaches such as plant_imagenet accelerated training convergence by 33.88–73.16% and improved F1-score by 8.72–42.12% over scratch strategies. However, there are still unresolved issues regarding the interaction between initialization strategies and activation functions or sequential architectures such as LSTM. The reason may be the explicit focus on isolating the initialization variables without involving other complexities such as fine-tuning. This approach is used to avoid experimental complexity, but leaves room for further exploration of the interactions between variables.

Study [20] compared four initialization methods (zero initialization, random initialization, Xavier/Glorot initialization, and He initialization) on the MNIST dataset and showed that He initialization is superior in terms of accuracy and loss. However, there are limitations as this study only used ReLU without evaluating other activation functions, and was not tested on the time series domain. This may be due to the simplification of the experimental design. Thus, further studies are needed that consider the interaction of activation function and data domain. Furthermore, study [21] tested the combination of dropout and weight initialization in the context of GAN for face classification. The results showed improved accuracy on unbalanced data. However, there was no explicit comparison between the current initialization methods. This makes the effectiveness of the methods inconclusive. This may be due to the focus on the regularization aspect of GANs, rather than the systematization of initialization. Future research could vary the activation function and explore how changes in network parameters impact the overall behavior of the model.

Research [22] investigated the generalization of deep learning models to X-ray image classification using an ensemble approach and six initialization methods namely Cold-start, Warm-start (ImageNet pretrained), and Shrink & Perturb with a Bayesian optimization approach. However, this study is limited to the VGG-16 architecture and does not explore the performance of initialization methods on architectures such as Transformer or other sequential models. The main reason for this limitation is the study's focus on initialization strategies and model ensembles and limited computational resources. Study [23] proposed a Recursive Weight Initialization (RWI) method in industrial defect diagnosis, which resulted in up to 92% accuracy. Although based on real data, this study did not conduct a systematic comparison with other popular methods. This is likely due to the focus on the integration of RWI and industrial pre-processing. Thus, the generalization validity of this method needs to be strengthened through further studies.

Research [24] introduced a new weight initialization method for satellite image classification, which is superior to Xavier and He's method without increasing the computational burden. However, the research focuses only on CNNs, and has not tested its effectiveness in time-series architectures such as LSTM and Transformer. This creates an important research gap, especially in the financial domain where temporal generalization capabilities are required. Furthermore, the study [25] analyzed 15 combinations between three initialization methods (Nguyen-Widrow, Xavier, Random) and five activation functions (Hyperbolic Tangent Sigmoid, Log-Sigmoid, Rectified Linear Unit, Gaussian Error Linear Unit, and Linear). This study highlights that the combination of Xavier-Linear and Nguyen-Widrow-Tanh gives the best results. However, this study is limited to feedforward neural networks (FNNs) for multi-dataset classification tasks and does not test sequential or time series models, which have different structural characteristics.

Based on a critical analysis of studies related to weight initialization in neural networks, it appears that although this approach has been tested in various domains such as medical image classification, face recognition, and remote sensing-based classification, there is no systematic study that evaluates the combined effect of initialization method and activation function in time-series models, especially regarding convergence speed, prediction accuracy, and the ability of the model to capture price variability. Moreover, the limitations on the variety of architectures and application domains (such as RNN, LSTM, and transformer for financial asset price prediction) suggest that the contribution of weight initialization strategies is still not fully explored in this context. This suggests that further studies are urgently needed to address this important gap. Therefore, this study focuses on systematically investigating this aspect:

1) how the weight initialization technique affects the convergence speed of RNN, LSTM, and Transformer models;

2) how the weight initialization technique impacts the performance of financial asset price prediction, both in terms of accuracy and the ability of the model to capture asset price variability.

To answer this question, this study analyzes various weight initialization techniques, including random uniform (RU), random Gaussian (RG), He normal (HN), Glorot uniform (GT), orthogonal (OG), and all-zeros (AZ), in RNN, LSTM, and transformer models. The six methods are common weight initialization strategies applied in deep learning modeling for time-based data prediction, and have different characteristics in maintaining the stability of gradient propagation and the efficiency of the training process as described in [5]. This study [5] shows that RU and RG initialize the weights randomly using uniform and Gaussian distributions, which are often used as classical baselines in various models. GT is designed to reduce the risk of vanishing and exploding gradients by adjusting the variance of the weights. HN has a similar principle to GT but is optimized for non-differentiable activation functions such as ReLU. OG initializes the weights in the form of an orthogonal matrix, which is known to maintain the stability of the signal representation. Meanwhile, AZ is used as an extreme case, where all weights are initialized with zero or a constant value.

In addition, each combination of model and initialization technique is further tested using three commonly used activation functions in time series processing, namely tanh, ReLU, and LeakyReLU [26–28], to evaluate the interaction between weight initialization and activation function non-linearity on convergence and accuracy of asset price prediction. Study [26] showed that these three functions are often used in CNNs adapted for time series due to their ability to capture spatial and temporal patterns. Study [27] explained that tanh is effective for sequential data because it is zero-centered, ReLU reduces the risk of vanishing gradient, and Leaky ReLU prevents dying neurons. This finding was reinforced by [28], who used all three in a CNN-LSTM model to improve short- and long-term prediction accuracy. Therefore, the selection of these three functions is relevant to examine the interaction with the initialization strategy in the time-series architecture.

The various limitations identified across previous studies indicate that no systematic approach has yet been undertaken to evaluate the effectiveness of weight initialization strategies in sequential deep learning models, particularly in the context of financial time-series forecasting. This unresolved issue arises from the narrow scope of architectural diversity, the predominance of non-financial domains in experimentation, and the limited attention given to the relationship between initialization techniques and training stability. Furthermore, the impact of initialization methods on convergence speed, prediction accuracy, and the model's ability to capture price variability has not been comprehensively investigated. All this allows to assert that it is expedient to conduct a study that systematically evaluates the effect of various weight initialization techniques on the performance of deep learning models for financial asset price prediction based on time-series data.

3. The aim and objectives of the study

The aim of the study is to evaluate the effect of weight initialization techniques on the performance of deep learning models in predicting financial asset prices based on time-series data. Proper weight initialization plays an important role in accelerating model convergence, reducing prediction error, and improving generalization ability.

To achieve this aim, the following objectives were accomplished:

– to analyze the characteristics of the FAR-Trans dataset through preliminary data exploration;

– to analyze the convergence behavior of each model, in terms of training time and the number of epochs required to converge;

– to assess the prediction accuracy of the RNN, LSTM, and transformer models using different weight initialization strategies and three different activation functions, as well as the ability to capture asset price variability.

4. Materials and methods

4. 1. Object and hypothesis of the study

The object of this research is a sequential-based deep learning model, while financial asset price prediction serves as the application context to test the model's performance. The main hypothesis in this study is that the combination of a particular weight initialization method and activation function has a significant influence on the convergence speed, prediction accuracy, and the model's ability to capture asset price variability. For experimental testing, this study uses the FAR-Trans dataset, which is the first public dataset for financial asset recommendations obtained from a major European financial institution [29].

Several assumptions are used to keep the scope of the experiments focused and scalable. First, the model uses only one hidden layer and the number of neurons is uniformed (128 units) to avoid the influence of architecture on performance. Second, the length of the input time window is set to 30 days and the prediction target is the closing price on day $(t+1)$, so the model does not handle long-term prediction horizons. Third, regularization techniques such as dropout or early stopping are not applied to isolate the effects of weight initialization strategies and activation functions on convergence and accuracy. Fourth, the model does not utilize external features other than historical prices.

The selection of the three models is based on the effectiveness of RNNs and LSTMs in capturing short-term temporal patterns in financial data [5], as well as the potential of Transformers in modeling complex sequences in time-series data [6–8]. The six weight initialization methods used (RU, RG, HN, GT, OG, AZ) were chosen because they are common techniques in deep learning for time series prediction, and represent a variety of strategies in managing gradient propagation and training stability [5]. Three activation functions (tanh, ReLU, LeakyReLU) were used as they are three commonly used activation functions in time series processing [26–28]. To operationalize the stated hypothesis and assumptions, this study conducts a comparative analysis of these weight initialization methods across the three models using financial asset price time-series data.

4. 2. Research design

Experiments were conducted in several stages as shown in Fig. 1. The first stage after inputting the dataset is exploratory data analysis to check for missing values and distributional data analysis. After that, data preprocessing is carried out, where asset data that does not meet the minimum number limit will be processed, after which the dataset is processed by sorting asset prices by time and forming a time-series representation suitable for deep learning models, normalized using MinMaxScaler (0,1) to improve the stability of model training.

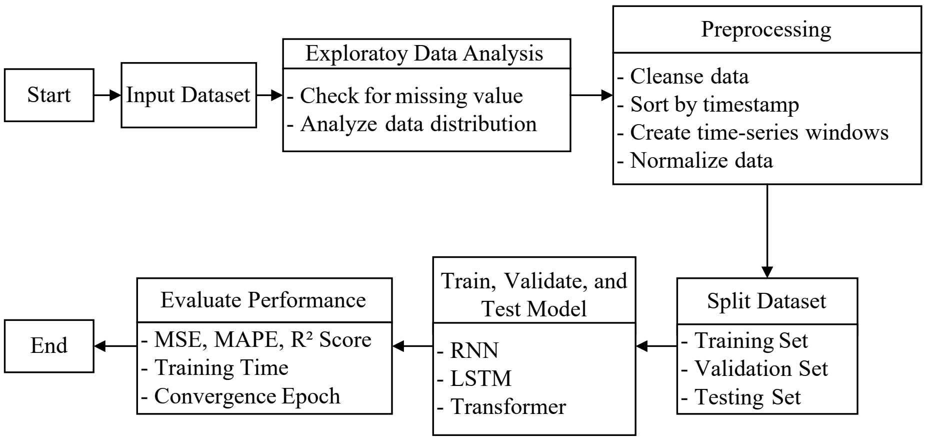

Fig. 1. Research design

The second stage is the division of the dataset into training, validation, and testing sets, which is done specifically to ensure that the historical pattern of the asset is maintained. Next, deep learning models consisting of RNN, LSTM, and transformer are built, with uniform parameters to ensure fair comparison. Models are trained using the training dataset, while model performance is validated using the validation set to monitor learning. After the training process is complete, the best model is tested using the testing dataset. Model performance evaluation is done using MSE, MAPE, and R^2 Score metrics, as well as observing the training time and number of epochs required until convergence.

To clarify the experimental procedure, here is the pseudocode of this study:

```
1. Load dataset.
2. Exploratory data analysis (EDA):
a) check for missing values;
b) analyze data distribution.
3. Preprocess data:
a) cleanse data;
b) sort by timestamp;
c) normalize values using MinMaxScaler (0,1);
d) create time-series windows.
4. Split dataset (train: 70%, validation: 15%, test: 15%).
5. For each model in [RNN, LSTM, transformer].
For each weight initialization in [tandom uniform, random Gaussian, He normal, Glorot uniform, orthogonal, all-zeros]:
For each activation function in [tanh, ReLU, LeakyReLU]:
a) build model using current model, weight initialization, and activation function;
b) train model on training set;
c) validate model on validation set;
d) test model on testing set;
e) evaluate using performance metrics.
6. Compare results and discuss findings.
```

4. 3. Experimental setup

The software used in this experiment includes Python as the main programming language, with TensorFlow/Keras libraries for the implementation of deep learning models, weight initialization methods, activation functions, and NumPy and Pandas for data manipulation processes. Normalization and performance evaluation of the model are done using Scikit-learn, while visualization is done through Matplotlib. In addition, openpyxl was used to save the evaluation results into an Excel file. All experiments were run using the Jupyter Notebook environment on a server with the following specifications: 20 physical cores (28 logical cores) with a CPU frequency of 2.04 GHz, 62.63 GB of RAM, and one active GPU unit.

This section describes the experimental parameters used in this study. The deep learning models tested have uniformly configured architectures to ensure a fair comparison between the methods:

a) input features. The model receives input in the form of financial asset closing price data, which is represented as a sequence of numerical values in time-series form;

b) window size. The length of the input window is set at 30 days. This means that to predict the closing price on day $(t+1)$, the model utilizes the previous 30 days of closing prices (from day 29 to day t) as input;

c) hidden layer. Each model consists of only one hidden layer to keep model complexity moderate and focus on the influence of weight initialization techniques and activation functions, rather than deep architecture;

d) hidden units. The number of units in the hidden layer is uniformed at 128 neurons;

e) batch size. The batch size is set to 64, which means the model will update its weights every time it finishes processing 64 training samples in one iteration;

f) target output. The model aims to predict the closing price one day after the input period $(t+1)$;

g) train-validation-test split. The dataset is divided into 70% training data, 15% validation data, and 15% testing data. The split is done separately for each asset, based on its International Securities Identification Number (ISIN), to maintain temporal structure and independence between assets;

h) optimizer. Adam's optimization was used with a learning rate of 0.001;

i) loss function. The loss function used during training is the Mean Squared Error (MSE), as it is suitable for regression tasks;

k) weight initialization method. This study tested six commonly used weight initialization techniques namely RU, RG, HN, GT, OG, and AZ;

l) activation function. Three activation functions were tested for each model namely tanh, ReLU and Leaky ReLU;

m) epochs. Training was conducted for a maximum of 50 epochs. This number was considered sufficient to observe the convergence pattern of the model, with monitoring done using validation loss;

n) model evaluation. The performance of the model was evaluated using several metrics, namely RMSE, MAPE, and R^2 Score. In addition, the speed of convergence is evaluated through the number of epochs required to stabilize, as well as the duration of training time.

5. Results on the impact of weight initialization on recurrent and transformer-based models

5. 1. Data characteristics for ensuring model forecasting reliability

To ensure the feasibility of the model training process, the first objective of this research is to analyze the structure and statistical characteristics of the FAR-Trans dataset. This analysis includes identifying the number of records, asset categories, distribution of historical data length per asset, and price trend patterns over the observation period. This exploratory data analysis (EDA) aims to assess whether the dataset has sufficiently rich, complete, and stable historical coverage to support time series-based modeling with deep learning approaches.

The FAR-Trans dataset represents a snapshot of the market available to Greek investors between January 2018 and November 2022, including the period of global crisis due to the Covid-19 pandemic in March 2020 and geopolitical tensions in early 2022. From this dataset, the research focuses on the closing price data in the close_prices.csv file, which contains closing date and price information of financial assets from various categories, including stocks, bonds, and mutual funds. This dataset is provided in comma-separated values (CSV) format and has a data structure that includes asset identification (ISIN), transaction date, and asset closing price. This dataset has no missing values, has 703303 records with 807 unique assets. However, there are 114 unique asset data with an insufficient number of each. Therefore, data cleansing was performed to maintain the input quality of the time-series model. Only assets with a minimum of 250 days of observation are retained, so that the division of training, validation, and testing data remains meaningful. After data cleansing, 688047 records were obtained with 693 unique assets. The statistical data of the closing price data before and after cleansing can be seen in Table 1. The distribution of the number of records per ISIN can be seen in Fig. 2. The daily closing price trend for one of the individual assets (ISIN: 100974271034) can be seen in Fig. 3.

Table 1

Description of asset price dataset

Property	Before cleansing	After cleansing
Amount of data	703303	688047
Unique assets	807	693
Minimum data per asset (days)	1	250
Maximum data per asset (days)	1279	1279
Average data per asset (days)	871.5	992.9

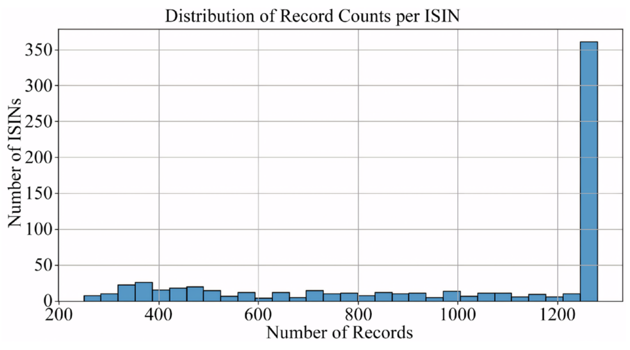

Fig. 2. Distribution of record counts per International Securities Identification Number

The histogram graph in Fig. 2 shows that the majority of assets (ISINs) have a very high amount of historical data. Most assets are close to the maximum of around 1279 records per ISIN, which is reflected by the significant frequency spike on the right side of the graph. In contrast, only a small percentage of ISINs have a number of records below 500.

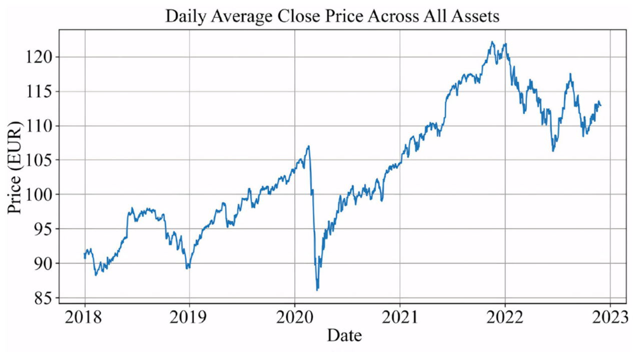

Fig. 3. Daily closing price trend of the asset with International Securities Identification Number 100974271034

Fig. 3 shows a sharp drop in early 2020 and a spike in prices until mid-2021. This visualization provides an overview of the historical patterns and market dynamics of the asset. After performing EDA on the dataset, the next step is to perform data preprocessing to ensure the data is in a ready-to-train condition. Next, the models are built and evaluated based on the scenarios specified in the experimental setup. The training process for each model is carried out up to a maximum of 50 epochs. The overall model testing results are presented in Tables 2–4, which contain a comparison of the evaluation metrics for the three tested architectures, namely RNN, LSTM, and transformer.

Table 2

Deep learning model evaluation results based on combination of weight initialization and activation function

Init	Act	E	RMSE	MAPE	R^2 Score	T (s)
1	2	3	4	5	6	7
RNN						
RU	tanh	1	3.6523	0.015	0.9999	867.07
RU	ReLU	1	3.641	0.014	0.9999	872.48
RU	Leaky	1	3.665	0.0107	0.9999	830.47
RG	tanh	1	4.9588	0.0186	0.9998	867.64
RG	ReLU	1	3.6356	0.0123	0.9999	874.68
RG	Leaky	1	3.6755	0.0149	0.9999	828.81

Continuation of Table 2

1	2	3	4	5	6	7
HN	tanh	1	3.6877	0.013	0.9999	867.32
HN	ReLU	1	3.675	0.0193	0.9999	871.75
HN	Leaky	4	3.6445	0.0125	0.9999	829.43
GT	tanh	1	3.7008	0.0169	0.9999	871.02
GT	ReLU	1	3.649	0.0132	0.9999	874.36
GT	Leaky	1	3.6429	0.0129	0.9999	829.16
OG	tanh	1	3.6878	0.0156	0.9999	870.69
OG	ReLU	1	3.6286	0.0158	0.9999	870.64
OG	Leaky	1	3.6402	0.0149	0.9999	832.19
AZ	tanh	1	3.7915	0.0126	0.9999	871.26
AZ	ReLU	2	37.4336	0.586	0.9886	870.05
AZ	Leaky	1	3.6894	0.0213	0.9999	832.76
Average			5.6166	0.0466	0.9993	857.32
LSTM						
RU	tanh	1	3.6516	0.0122	0.9999	631.38
RU	ReLU	1	3.6834	0.0136	0.9999	954.46
RU	Leaky	1	3.6671	0.0105	0.9999	637.45
RG	tanh	1	3.6856	0.0129	0.9999	624.43
RG	ReLU	1	3.6979	0.0135	0.9999	965.42
RG	Leaky	1	3.8304	0.0211	0.9999	641.48
HN	tanh	1	3.6539	0.011	0.9999	628.87
HN	ReLU	2	3.6689	0.0123	0.9999	971.47
HN	Leaky	1	3.7019	0.0128	0.9999	639.36
GT	tanh	1	3.7304	0.0111	0.9999	636.49
GT	ReLU	1	3.7612	0.0159	0.9999	962.76
GT	Leaky	1	3.6705	0.0114	0.9999	641.54
OG	tanh	1	3.6998	0.0118	0.9999	628.69
OG	ReLU	1	3.6567	0.0124	0.9999	952.77
OG	Leaky	1	3.6771	0.0123	0.9999	637.82
AZ	tanh	1	3.6672	0.0108	0.9999	628.69
AZ	ReLU	2	37.3551	0.5867	0.9886	957.43
AZ	Leaky	1	3.6872	0.0107	0.9999	635.89
Average			5.5637	0.0446	0.9993	743.13
Transformer						
RU	tanh	1	37.7091	0.5836	0.9884	364.62
RU	ReLU	1	37.93	0.5816	0.9883	364.73
RU	Leaky	1	37.5683	0.5848	0.9885	363.12
RG	tanh	2	37.7901	0.5828	0.9884	369.79
RG	ReLU	1	37.6441	0.5841	0.9884	367.07
RG	Leaky	2	37.6327	0.5842	0.9884	364.91
HN	tanh	2	37.7588	0.5831	0.9884	366.47
HN	ReLU	2	37.435	0.586	0.9886	366.9
HN	Leaky	2	37.8882	0.582	0.9883	364.1
GT	tanh	2	37.1772	0.5884	0.9888	412.51
GT	ReLU	2	37.6168	0.5844	0.9884	368.66
GT	Leaky	2	37.6409	0.5842	0.9884	361.3
OG	tanh	2	37.2917	0.5873	0.9886	368.84
OG	ReLU	2	37.93	0.5816	0.9883	364.73
OG	Leaky	2	38	0.581	0.9882	365.24
AZ	tanh	2	37.0732	0.5893	0.9888	365.1
AZ	ReLU	2	37.3827	0.5865	0.9886	363.15
AZ	Leaky	2	38.0778	0.5803	0.9882	359.21
Average			37.6415	0.5842	0.9884	367.8

Notes: Init – weight initialization, Act – activation function, E – convergence epoch, T – training time in seconds, RF – random uniform, RG – random Gaussian, HN – He normal, GT – Glorot, OG – orthogonal, AZ – all-zeros, Leaky – leaky ReLU.

Table 3

Average model evaluation results based on weight initialization technique

Init	E	RMSE	MAPE	R^2 Score	T (s)
RNN					
RU	1	3.6528	0.0132	0.9999	856.67
RG	1	4.09	0.0152	0.9999	857.04
HN	2	3.6691	0.0149	0.9999	856.17
GT	1	3.6642	0.0143	0.9999	858.18
OG	1	3.6522	0.0154	0.9999	857.84
AZ	1.33	14.9715	0.2066	0.9961	858.02
LSTM					
RU	1	3.6674	0.0121	0.9999	741.1
RG	1	3.738	0.0158	0.9999	743.78
HN	1.33	3.6749	0.012	0.9999	746.57
GT	1	3.7207	0.0128	0.9999	746.93
OG	1	3.6779	0.0122	0.9999	739.76
AZ	1.33	14.9032	0.2027	0.9961	740.67
Transformer					
RU	1	37.7358	0.5833	0.9884	364.16
RG	1.67	37.689	0.5837	0.9884	367.26
HN	2	37.694	0.5837	0.9884	365.82
GT	2	37.4783	0.5857	0.9885	380.82
OG	2	37.7406	0.5833	0.9884	366.27
AZ	2	37.5112	0.5854	0.9885	362.49

Notes: Init – weight initialization, Act – activation function, E – convergence epoch, T – training time in seconds, RF – random uniform, RG – random Gaussian, HN – He normal, GT – Glorot, OG – orthogonal, AZ – all-zeros, Leaky – leaky ReLU.

Table 4

Best evaluation results per combination of weight initialization and activation function

Init	E	RMSE	MAPE	R^2 Score	T (s)
RNN					
RU	1 (tanh, ReLU, Leaky)	3.641 (ReLU)	0.0107 (Leaky)	0.9999 (tanh, ReLU, Leaky)	830.47 (Leaky)
RG	1 (tanh, ReLU, Leaky)	3.6356 (ReLU)	0.0123 (ReLU)	0.9999 (ReLU, Leaky)	828.81 (Leaky)
HN	1 (tanh, ReLU)	3.6645 (Leaky)	0.0125 (Leaky)	0.9999 (tanh, ReLU, Leaky)	829.43 (Leaky)
GT	1 (tanh, ReLU, Leaky)	3.6429 (Leaky)	0.0129 (Leaky)	0.9999 (tanh, ReLU, Leaky)	829.16 (Leaky)
OG	1 (tanh, ReLU, Leaky)	3.6286 (ReLU)	0.0149 (Leaky)	0.9999 (tanh, ReLU, Leaky)	832.19 (Leaky)
AZ	1 (tanh, Leaky)	3.6894 (Leaky)	0.0126(tanH)	0.9999 (tanh, Leaky)	832.76 (Leaky)
LSTM					
RU	1 (tanh, ReLU, Leaky)	3.6516 (tanh)	0.0105 (Leaky)	0.9999 (tanh, ReLU, Leaky)	631.38 (tanh)
RG	1 (tanh, ReLU, Leaky)	3.6856 (tanh)	0.0129 (tanh)	0.9999 (tanh, ReLU, Leaky)	624.43 (tanh)
HN	1 (tanh, Leaky)	3.6539 (tanh)	0.011 (tanh)	0.9999 (tanh, ReLU, Leaky)	828.87 (tanh)
GT	1 (tanh, ReLU, Leaky)	3.6705 (Leaky)	0.0111 (tanh)	0.9999 (tanh, ReLU, Leaky)	639.49 (tanh)
OG	1 (tanh, ReLU, Leaky)	3.6567 (ReLU)	0.1118 (tanh)	0.9999 (tanh, ReLU, Leaky)	628.69 (tanh)
AZ	1 (tanh, Leaky)	3.6672 (tanh)	0.0107 (Leaky)	0.9999 (tanh, Leaky)	828.69 (tanh)
Transformer					
RU	1 (tanh, ReLU, Leaky)	37.5683 (Leaky)	0.5816 (ReLU	0.9883 (ReLU)	363.12 (Leaky)
RG	1 (ReLU)	37.6327 (Leaky)	0.5828 (tanh)	0.9884 (tanh, ReLU, Leaky)	364.91 (Leaky)
HN	2 (tanh, ReLU, Leaky)	37.435 (ReLU)	0.582 (Leaky)	0.9883 (Leaky)	364.1 (Leaky)
GT	2 (tanh, ReLU, Leaky)	37.1772 (tanh)	0.5842 (Leaky)	0.9884 (ReLU, Leaky)	361.3 (Leaky)
OG	2 (tanh, ReLU, Leaky)	37.2917 (tanh)	0.581 (Leaky)	0.9882 (Leaky)	364.73 (ReLU)
AZ	2 (tanh, ReLU, Leaky)	37.0732 (tanh)	0.5803 (Leaky)	0.9882 (Leaky)	359.21 (Leaky)

Notes: Init – weight initialization, Act – activation function, E – convergence epoch, T – training time in seconds, RF – random uniform, RG – random Gaussian, HN – He normal, GT – Glorot, OG – orthogonal, AZ – all-zeros, Leaky – leaky ReLU.

The tests shown in Table 2 were designed to evaluate a combination of three deep learning models, six weight initialization techniques, and three activation functions. Table 3 presents the average evaluation results of the three activation functions, namely tanh, ReLU, and LeakyReLU. The averages are calculated for each weight initialization method in each model, namely RNN, LSTM, and transformer. Meanwhile, Table 4 summarizes the best results of each combination of weight initialization and activation function.

5. 2. Evaluation of model convergence speed

Table 2 shows that most of the RNN configurations converged on the 1st epoch with an average training time of 857 seconds. Most of the LSTM model configurations also converged within 1 epoch with an average training time of 743 seconds. The transformer model showed different convergence behavior compared to the RNN and LSTM. The transformer requires two epochs in almost all combinations with an average training time of 368 seconds.

The convergence graph of the RNN model is shown in Fig. 4, the LSTM model in Fig. 5, and the transformer model in Fig. 6.

Fig. 4 shows that the RNN model with OG-ReLU combination converges in only 1 epoch, which is characterized by the alignment between training loss and validation loss. Fig. 5 shows a similar convergence pattern for the LSTM model with the RU-Leaky combination. Meanwhile, Fig. 6 shows that the transformer model with GT-ReLU combination requires 2 epochs to reach convergence.

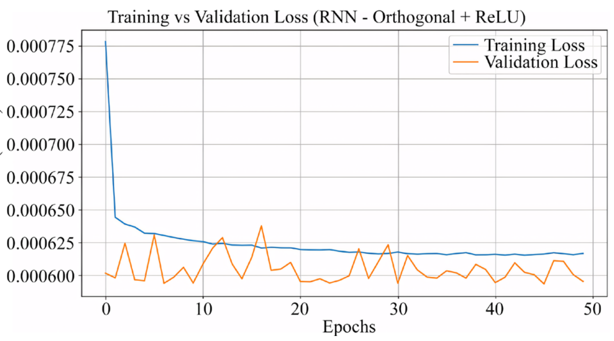

Fig. 4. Training and validation loss graph of the recurrent neural network model using orthogonal weight initialization and rectified linear unit activation function

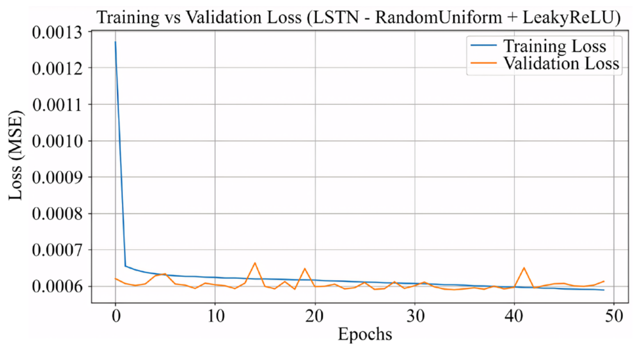

Fig. 5. Training and validation loss graph of the long short-term memory model using random uniform weight initialization and leaky rectified linear unit activation function

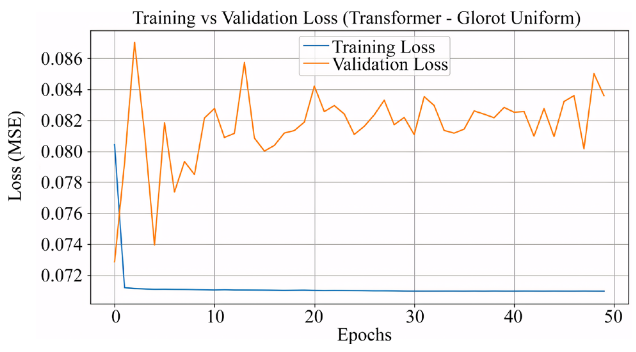

Fig. 6. Transformer model using Glorot uniform weight initialization and rectified linear unit activation function

5. 3. Evaluation of prediction performance

Based on the results in Table 2, the average RMSE value of the RNN model is 5.6166, while the average MAPE value reaches 0.0466. The lowest RMSE value of was recorded at 3.6286, and the lowest MAPE value was 0.0107. The average RMSE value of the LSTM model is 5.5637, while the average MAPE value reaches 0.0466. The lowest RMSE value was recorded at 3.6516, and the lowest MAPE value was 0.0105. The transformer model shows the average RMSE value is 37.6415, while the average MAPE value reaches 0.5842. Visualization of the performance of the RNN, LSTM and transformer models can be seen in Fig. 7.

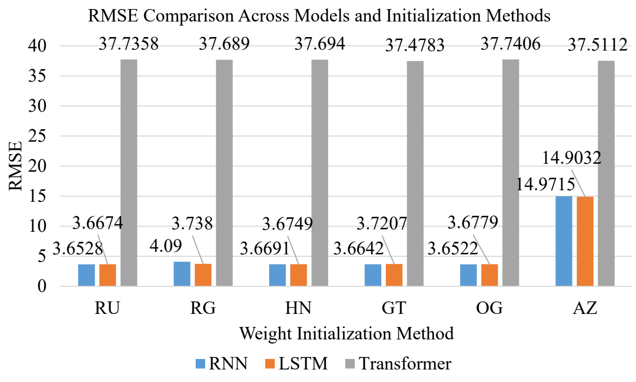

Fig. 7. Root mean squared error comparison chart across models and weight initialization methods

Fig. 7 presents the average RMSE values for each weight initialization method in the RNN model, as shown in Table 3, with values ranging from 3.65 to 14.97. In the LSTM model, the average RMSE value ranges from 3.67 to 14.9, while in the Transformer it is in the range of 37. Fig. 8 shows the comparison between the actual and predicted prices of the RNN model with the OG-ReLU combination. Similar results are also obtained for the LSTM model, which shows a prediction pattern close to the actual price with certain combinations of methods and activation functions. Fig. 9 shows a similar comparison of the Transformer model with the GT-ReLU combination.

Fig. 10 presents a visualization of the average MAPE value for each weight initialization method in the RNN, LSTM, and transformer models. The MAPE values on the RNN model are in the range of 0.01 to 0.21. In the LSTM model, the average MAPE value ranges from 0.01 to 0.2, while in the Transformer it is in the range of 0.58.

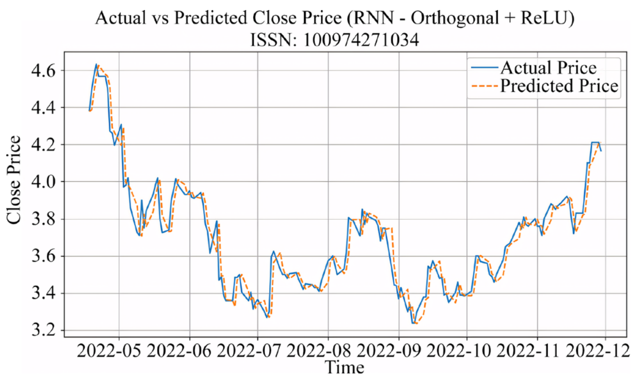

Fig. 8. Graph of actual and predicted price of one asset using the recurrent neural network model with orthogonal weight initialization and rectified linear unit activation function

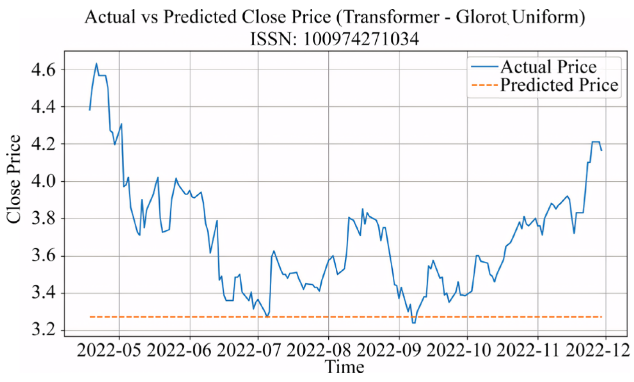

Fig. 9. Graph of actual and predicted price of one asset using the transformer model with Glorot uniform weight initialization and rectified linear unit activation function

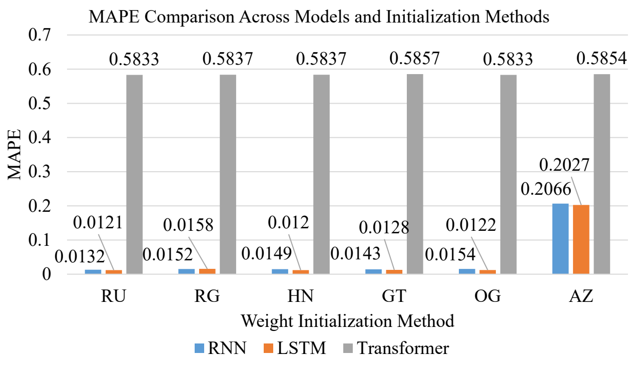

Fig. 10. Mean absolute percentage error comparison chart across models and weight initialization methods

In addition to prediction accuracy, this study also evaluates the model's ability to capture asset price variability, which is measured using the coefficient of determination (R^2).

The test results in Table 2 show that the RNN model obtained a very consistent R^2 value of 0.9999 in almost all combinations of weight initialization and activation function. The LSTM model also showed a similar pattern, with most combinations yielding R^2 values of 0.9999.

In contrast, the transformer model recorded lower R^2 values, although it remained relatively stable with an average of 0.9884.

6. Discussion of evaluation results on the effect of weight initialization techniques on convergence speed, accuracy, and ability to capture price variability in asset prediction models

An exploratory analysis of the FAR-Trans dataset was conducted to address the first objective of this study. The results of the analysis show that the data used has a reasonable historical coverage and stability before being used in time-series model training. The distribution pattern in Fig. 2 shows that the dataset has many assets with complete historical coverage, while a small number of assets have relatively short histories. This reinforces the reason for data cleansing to remove assets with too little data so that the learning model is not disturbed by minimal and unstable data representation. This exploratory data analysis as shown in Table 1, Fig. 3 confirms that the dataset contains assets with rich historical patterns and volatile price behavior. The data structure is free of missing values, and the cleansing process based on observation quantity makes this dataset suitable for deep learning-based prediction experiments.

An analysis of the effect of weight initialization on model convergence speed was conducted to answer the second objective of this study using two main metrics, namely convergence epoch and training time. In Table 3, all weight initialization methods in the RNN model (except HN and AZ) achieved convergence in an average of 1 epoch with an average duration of 856 to 858 seconds. HN requires an average of 2 epochs. While AZ required an average of 1.33 epochs, but with poor RMSE and MAPE results, indicating unstable convergence. This finding is consistent with study [5], which reported that AZ tends to slow down the network training process. In terms of time efficiency, the HN (856.17 seconds) and RU (856.67 seconds) methods were the fastest according to the average training time on the RNN. However, Table 4 indicates that the best results of all initialization methods on the RNN can be achieved in 1 epoch through the activation functions of:

a) tanh, ReLU and LeakyReLU (RU, RG, GT, and OG);
b) tanh and ReLU (HN);
c) tanh and Leaky (AZ).

The fastest training time was shown by RG (828.81 seconds) and GT (829.16 seconds) with LeakyReLU activation function. The finding that the GT method has an advantage in convergence speed is in line with the study of [5]. However, different from [5], this study shows that the RU method can provide fast and stable convergence, less in line with the claim that randomized methods always produce uncontrollable initial weights.

In the LSTM model, Table 3 shows that most methods achieve convergence within 1 epoch (including RU, RG, GT, OG), while HN and AZ require longer epochs. Interestingly, the training time of LSTM is generally faster than that of RNN, with an average duration of 739 to 747 seconds. Notably, the OG and AZ methods recorded the most efficient training times of 739.76 seconds and 740.67 seconds, respectively. However, Table 4 shows that the best results of all initialization methods on LSTM can be achieved within 1 epoch through the activation functions of a) tanh,ReLU,ReLULeaky (RU, RG, GT, OG), and b) tanh,LeakyReLU (HN and AZ). In terms of time efficiency, the initialization methods in LSTM that require the fastest training time are RG with a duration of 624.43 seconds and OG with a duration of 628.69 seconds, both using the tanh activation function. The finding of the superiority of the

OG method in stabilizing the training process is in line with research [5].

Meanwhile, the transformer model shows a different convergence behavior. Table 2 shows that transformer requires two epochs in almost all combinations, indicating training that is not as fast as RNN/LSTM. However, the fastest training time is for the Transformer with an average duration of 362 to 381 seconds, making it runtime efficient, but resulting in lower accuracy. Tables 3, 4 also indicate that while some methods in the transformer achieve the best results within 1 epoch, the high RMSE and MAPE indicate that fast convergence does not necessarily translate to good learning in the context of asset price prediction.

Thus, the results of this analysis close the gap identified in the lack of systematic evaluation, especially of the convergence speed of models based on a combination of initialization and activation. This study shows that the RU method, which was previously considered less stable, is capable of providing fast convergence in the context of the FAR-Trans dataset. Moreover, the finding that Transformer requires more epochs despite its short runtime provides new insights that time efficiency does not always guarantee training efficiency.

An analysis of the effect of weight initialization on the model's predictive performance is conducted to address the third objective of this study, using three main metrics, namely RMSE and MAPE to measure prediction accuracy, and R^2 Score to evaluate the model's ability to capture asset price variability. Based on Table 2, the AZ-ReLU combination shows symptoms of overfitting in the RNN and LSTM models, characterized by high RMSE and MAPE values (around 37 and 0.5 respectively). This is likely due to the initialization of the A-Z weights that resulted in a suboptimal initial weight distribution for the asset price dataset, resulting in the model focusing too much on specific patterns of the training data and failing to generalize to the test data. In addition, the ReLU activation function is prone to the dying ReLU problem (zero gradient on negative inputs), which worsens the model's ability to capture high variability in the data. Fig. 4 shows that the RNN model with the OG-ReLU combination has good predictive ability of asset prices. Meanwhile, Fig. 5 shows that the LSTM model with RU-Leaky combination also shows good predictive performance.

Based on Table 3, which displays the average values of the three activation functions for each weight initialization method, the RNN and LSTM models consistently show much better accuracy performance compared to the transformer model. On the RNN architecture, the lowest RMSE is obtained from the OG initialization method with an average value of 3.6522 and RU with an average value of 3.6528, as shown in Fig. 7, the lowest MAPE is obtained from RU initialization method with an average value of 0.0132 and GT with an average value of 0.0143, as shown in Fig. 10. Fig. 8 shows that there is a high agreement between the actual price and the predicted result in the RNN model. In contrast, the AZ method gave poor results (average RMSE: 14.9715; average MAPE: 0.2027), indicating that initialization with zero weights is not able to provide a good initial representation in the RNN training process. This finding is consistent with the results of [5]. Table 4 shows the best RMSE values for the RNN were obtained from the OG initialization method with a value of 3.6286 and RG with a value of 3.6356, both using the ReLU activation function. The best MAPE for RNN was obtained from the RU initialization method with a value of 0.0107 through the LeakyReLU activation function and RG with a value of 0.0123 through the ReLU activation function. This finding shows that both ReLU and LeakyReLU can provide excellent prediction performance depending on the initialization combination used. The contribution of LeakyReLU in providing excellent accuracy is in line with the results of [17].

Table 3 shows the RU and HN initializations of the LSTM model recorded the lowest RMSE with average values of 3.6674 and 3.6749, as shown in Fig. 7, HN and RU initialization recorded the lowest MAPE with average values of 0.012 and 0.0121, as shown by Fig. 10. Although the AZ method has individually produced quite low MAPE, the high RMSE values and inconsistency of the results indicate that this method is not feasible to use like the results of [5, 20]. Table 4 shows the best RMSE values for LSTM were obtained from the RU initialization method with a value of 3.6516 and HN with a value of 3.6539, both using the tanh activation function. The best MAPE for LSTM is obtained from the RU initialization method with a value of 0.0105 through the LeakyReLU activation function and HN with a value of 0.011 through the tanh activation function. The findings on the reliable performance of the HN, GT, and RU methods are in line with the results of [20]. Study [18] also supports the contribution of the tanh activation function in producing accurate predictions.

On the other hand, the transformer model shows clear symptoms of overfitting as shown in Tables 2, 3, characterized by the high values of RMSE (Fig. 7) and MAPE (Fig. 10) across all combinations of initialization and activation functions, as well as the striking discrepancy between actual prices and model predictions (Fig. 9). Fig. 6 also shows the overfitting pattern of the transformer model with the GT-ReLU combination, where there is an imbalance between training loss and validation loss throughout the training. Similar findings are also corroborated by the study of [8], which shows that the Transformer's performance is also ineffective for case studies with other time-series data. Theoretically, this weakness can be explained by the technical review in [7], which shows that transformer is very sensitive to the dataset size and requires a specialized training approach.

In addition to the prediction error indicator, the model's ability to capture price variability is also an important aspect to evaluate. Based on Table 3, all weight initialization methods in the RNN and LSTM models show very high average R^2 values of 0.9999 for all initialization methods except AZ. This indicates that the RNN and LSTM models with various initialization techniques are able to capture almost all variations in asset prices. In contrast, the AZ method only achieved an average R^2 of 0.9961, indicating a decline in performance. This decrease is most likely due to the initial weight distribution which tends to produce missing gradients during training. In the Transformer model, the average R^2 value is much lower than that of the RNN and LSTM, ranging from 0.9884 to 0.9885 for all initialization techniques. This indicates that the transformer is less effective in representing asset price variability in the context of the dataset used, although it remains within the range of good performance in general. This finding of transformer's ineffective performance for the time-series data case study is also corroborated by [8].

This finding is further reinforced by the data in Table 4, which shows that the best R^2 values across all models were achieved for almost all initialization methods, with maximum values ranging from 0.9998 to 0.9999 for RNN and LSTM. This strengthens the evidence that initialization methods such as

RU, RG, HN, GT, OG, and AZ combined with any activation function (except the AZ-ReLU combination) are able to optimally capture price variability patterns. Meanwhile, on the transformer, the highest values remain limited to the range of 0.9882 to 0.9885.

Thus, the results of this analysis close the identified gap, which is the lack of a systematic evaluation of the effect of the combination of weight initialization method and activation function on the predictive performance of the model, both in terms of prediction accuracy and the ability to capture asset price variability. The results of this study provide empirical evidence that certain combinations can significantly improve model performance in both aspects. Specifically, this study shows that the combination of All-Zeros (AZ) with ReLU activation function tends to cause overfitting and results in lower R^2 values in RNN and LSTM models. In contrast, such symptoms do not appear when AZ is paired with tanh or LeakyReLU activation functions.

This study has a number of limitations that need to be considered in interpreting the results. Firstly, this study only compares three deep learning model architectures, namely RNN, LSTM, and transformer. Other models that are also commonly used in time series processing were not included, so the results cannot be generalized to all types of models. Secondly, the activation functions tested were limited to tanh, ReLU, and LeakyReLU. Alternative activation functions may yield different results but have not been explored in this study. Third, this study only uses one financial asset price dataset, so the results have not been tested on assets with different characteristics or in highly volatile market conditions.

Several shortcomings of this study have the potential to reduce the reliability of the results, even within the specified scope. First, the study did not include interactions with regularization techniques (e.g., dropout, early stopping), which can significantly improve model generalization. Secondly, indications of overfitting found in transformer models with certain combinations have not been followed up in depth with mitigation strategies, so the potential for performance improvement is still not optimized. Third, there has been no exploration of method combinations such as ensemble or hybrid models, even though these approaches can overcome the shortcomings of each model.

As a practical implication of the findings, this research has relevant applications in deep learning-based asset price prediction systems, particularly in the development of financial asset recommendation models, automated trading systems, portfolio decision support systems, and early warning mechanisms against market risks. Based on the experimental findings, weight initialization methods such as RU, RG, OG, and GT are recommended for RNN architecture, while RU and HN show the best performance for LSTM. In contrast, the use of AZ should be avoided, especially in combination with ReLU, as it significantly increases the risk of overfitting. This approach can be effectively applied in the context of using time series data with sufficient historical length and good preprocessing. In addition, it is suitable for short-term prediction scenarios using inputs of historical prices without involving external variables such as news or market sentiment. Under these conditions, the selection of appropriate weight initialization strategies and activation functions is proven to speed up the convergence process, improve prediction accuracy, and reduce overfitting, especially for sequential models such as RNN and LSTM.

To overcome the limitations of the study, it is suggested that future research consider using additional architectures, or hybrid models. Future research can also extend experiments to other activation functions and weight initialization schemes to gain more insight. In addition, the use of more varied datasets from different asset sectors and temporal coverage will improve the generalizability of the results. The addition of regularization techniques and more extensive hyperparameter tuning can also improve the stability and accuracy of the model, especially in dealing with overfitting. Finally, future research is recommended to integrate interpretability methods to understand the contribution of features to model predictions, thus supporting the transparency and validity of machine learning-based decisions in finance.

However, the development of this study is likely to face a number of challenges. Mathematically, the use of more complex architectures and new activation functions may increase the risk of gradient exploding/vanishing. Methodologically, more extensive hyperparameter tuning requires significantly longer training time and high computational resources. Experimental challenges may also arise from the limited availability of multivariable datasets with sufficient label quality. In addition, the interpretation of complex models such as LSTM and transformer in the context of time-series data and financial domain remains a challenge. Therefore, further development should be accompanied by mitigation strategies for these risks.

7. Conclusions

1. The FAR-Trans dataset is shown to have stable and representative historical coverage, with 688,047 records of 693 unique assets after cleansing. Most assets have near-maximum data length (1279 days), with price fluctuations reflecting real market dynamics. This makes the dataset feasible for use in deep learning-based financial time series prediction studies.

2. The convergence speed evaluation shows that the RNN and LSTM models excel in training stability, requiring only one epoch on average to converge. In contrast, the Transformer requires two epochs but shows efficient training time per epoch. This confirms that while the transformer is often promoted as a superior model in many domains, for the context of relatively stable financial time-series, the traditional sequential architecture remains more optimal in terms of training efficiency and stability.

3. In terms of prediction accuracy and ability to capture price variability, the RNN and LSTM models show the most consistent performance with RMSE < 3.7 and MAPE < 0.015. The R^2 value of almost 0.9999 indicates the high ability of both models to recognize historical price patterns. In contrast, the transformer model shows inferior performance (RMSE around 37, MAPE around 0.58, average R^2 value between 0.9884 to 0.9885), which is thought to be due to the sensitivity of its architecture to data length and input sequence patterns which is weaker than that of the sequential model. This study also shows that the use of AZ should be avoided, especially when combined with ReLU, as it causes overfitting of the RNN and LSTM. However, the combination of AZ-tanh and AZ-LeakyReLU can still be used with caution. These findings provide empirical evidence that proper selection of weight initialization methods and activation functions can significantly improve model performance and prevent instability in training.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this study, whether financial, personal, authorship or otherwise, that could affect the study and its results presented in this paper.

Financing

The study was performed without financial support.

Data availability

Manuscript has associated data in a data repository.

Use of artificial intelligence

The authors have used artificial intelligence technologies within acceptable limits to provide their own verified data, which is described in the research methodology section.

Acknowledgments

We would like to thank Prof. Dr. Syahril Efendi S. Si., M. I. T. and Dr. Tengku Henny Febriana Harumy S. Kom, M. Kom from the Universitas Sumatera Utara, Medan, Indonesia for the support, guidance, and direction to make this article could be produced and published.

References

1. Achury-Calderón, F., Arredondo, J. A., Sánchez Ascanio, L. C. (2025). A novel predictive analytics model for forecasting short-term trends in equity assets prices. Decision Analytics Journal, 14, 100534. https://doi.org/10.1016/j.dajour.2024.100534
2. Tan, J., Deveci, M., Li, J., Zhong, K. (2024). Asset pricing via fused deep learning with visual clues. Information Fusion, 102, 102049. https://doi.org/10.1016/j.inffus.2023.102049
3. Chen, Y., Zhang, L., Xie, Z., Zhang, W., Li, Q. (2025). Unraveling asset pricing with AI: A systematic literature review. Applied Soft Computing, 175, 112978. https://doi.org/10.1016/j.asoc.2025.112978
4. Harumy, H. F., Hardi, S. M., Al Banna, M. F. (2024). EarlyStage Diabetes Risk Detection Using Comparison of Xgboost, Lightgbm, and Catboost Algorithms. Advanced Information Networking and Applications, 12–24. https://doi.org/10.1007/978-3-031-57931-8_2
5. Al-Selwi, S. M., Hassan, M. F., Abdulkadir, S. J., Muneer, A., Sumiea, E. H., Alqushaibi, A., Ragab, M. G. (2024). RNN-LSTM: From applications to modeling techniques and beyond – Systematic review. Journal of King Saud University - Computer and Information Sciences, 36 (5), 102068. https://doi.org/10.1016/j.jksuci.2024.102068
6. Kim, D.-K., Kim, K. (2022). A Convolutional Transformer Model for Multivariate Time Series Prediction. IEEE Access, 10, 101319–101329. https://doi.org/10.1109/access.2022.3203416
7. Ahmed, S., Nielsen, I. E., Tripathi, A., Siddiqui, S., Ramachandran, R. P., Rasool, G. (2023). Transformers in Time-Series Analysis: A Tutorial. Circuits, Systems, and Signal Processing, 42 (12), 7433–7466. https://doi.org/10.1007/s00034-023-02454-8
8. Ahn, J. Y., Kim, Y., Park, H., Park, S. H., Suh, H. K. (2024). Evaluating Time-Series Prediction of Temperature, Relative Humidity, and CO_2 in the Greenhouse with Transformer-Based and RNN-Based Models. Agronomy, 14 (3), 417. https://doi.org/10.3390/agronomy14030417
9. de Pater, I., Mitici, M. (2023). A mathematical framework for improved weight initialization of neural networks using Lagrange multipliers. Neural Networks, 166, 579–594. https://doi.org/10.1016/j.neunet.2023.07.035
10. Harumy, T., Ginting, D. S. Br. (2021). Neural Network Enhancement Forecast of Dengue Fever Outbreaks in Coastal Region. Journal of Physics: Conference Series, 1898 (1), 012027. https://doi.org/10.1088/1742-6596/1898/1/012027
11. Harumy, T. H. F., Zarlis, M., Lydia, M. S., Efendi, S. (2023). A novel approach to the development of neural network architecture based on metaheuristic protis approach. Eastern-European Journal of Enterprise Technologies, 4 (4 (124)), 46–59. https://doi.org/10.15587/1729-4061.2023.281986
12. Ghallabi, F., Souissi, B., Du, A. M., Ali, S. (2025). ESG stock markets and clean energy prices prediction: Insights from advanced machine learning. International Review of Financial Analysis, 97, 103889. https://doi.org/10.1016/j.irfa.2024.103889
13. Xu, F., Tan, S. (2021). Deep learning with multiple scale attention and direction regularization for asset price prediction. Expert Systems with Applications, 186, 115796. https://doi.org/10.1016/j.eswa.2021.115796
14. Abolmakarem, S., Abdi, F., Khalili-Damghani, K., Didehkhani, H. (2022). A Multi-Stage Machine Learning Approach for Stock Price Prediction: Engineered and Derivative Indices. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4074883
15. Pan, S., Long, S., Wang, Y., Xie, Y. (2023). Nonlinear asset pricing in Chinese stock market: A deep learning approach. International Review of Financial Analysis, 87, 102627. https://doi.org/10.1016/j.irfa.2023.102627
16. Gülmez, B. (2025). GA-Attention-Fuzzy-Stock-Net: An optimized neuro-fuzzy system for stock market price prediction with genetic algorithm and attention mechanism. Heliyon, 11 (3), e42393. https://doi.org/10.1016/j.heliyon.2025.e42393
17. Islam, B. ul, Ahmed, S. F. (2022). Short-Term Electrical Load Demand Forecasting Based on LSTM and RNN Deep Neural Networks. Mathematical Problems in Engineering, 2022, 1–10. https://doi.org/10.1155/2022/2316474
18. Rahman, N. H. A., Yin, C. H., Zulkafli, H. S. (2024). Activation functions performance in multilayer perceptron for time series forecasting. Proceedings of the 38th International Conference of the Polymer Processing Society (PPS-38), 3158, 070001. https://doi.org/10.1063/5.0223864
19. Sinanc Terzi, D. (2024). Effect of different weight initialization strategies on transfer learning for plant disease detection. Plant Pathology, 73 (9), 2325–2343. https://doi.org/10.1111/ppa.13997

20. Srivastava, G., Vashisth, S., Dhall, I., Saraswat, S. (2020). Behavior Analysis of a Deep Feedforward Neural Network by Varying the Weight Initialization Methods. Smart Innovations in Communication and Computational Sciences, 167–175. https://doi.org/10.1007/978-981-15-5345-5_15
21. Alhlffee, M. H. B., Ahmad Abuirbaiha, R. A. (2024). The Effects of Dropout and Weight Initialization on Human Face Classification Accuracy Using Multiple-agent Generative Adversarial Network. 2024 7th International Conference on Information and Computer Technologies (ICICT), 271–276. https://doi.org/10.1109/icict62343.2024.00050
22. Rajaraman, S., Zamzmi, G., Yang, F., Liang, Z., Xue, Z., Antani, S. (2024). Uncovering the effects of model initialization on deep model generalization: A study with adult and pediatric chest X-ray images. PLOS Digital Health, 3 (1), e0000286. https://doi.org/10.1371/journal.pdig.0000286
23. Berghout, T., Bentrcia, T., Lim, W. H., Benbouzid, M. (2023). A Neural Network Weights Initialization Approach for Diagnosing Real Aircraft Engine Inter-Shaft Bearing Faults. Machines, 11 (12), 1089. https://doi.org/10.3390/machines11121089
24. Boulila, W., Alshanqiti, E., Alzahem, A., Koubaa, A., Mlaiki, N. (2024). An effective weight initialization method for deep learning: Application to satellite image classification. Expert Systems with Applications, 254, 124344. https://doi.org/10.1016/j.eswa.2024.124344
25. Wong, K., Dornberger, R., Hanne, T. (2022). An analysis of weight initialization methods in connection with different activation functions for feedforward neural networks. Evolutionary Intelligence, 17 (3), 2081–2089. https://doi.org/10.1007/s12065-022-00795-y
26. Mojtahedi, F. F., Yousefpour, N., Chow, S. H., Cassidy, M. (2025). Deep Learning for Time Series Forecasting: Review and Applications in Geotechnics and Geosciences. Archives of Computational Methods in Engineering. https://doi.org/10.1007/s11831-025-10244-5
27. Mienye, I. D., Swart, T. G., Obaido, G. (2024). Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. Information, 15 (9), 517. https://doi.org/10.3390/info15090517
28. Huang, H., Wang, Z., Liao, Y., Gao, W., Lai, C., Wu, X., Zeng, Z. (2024). Improving the explainability of CNN-LSTM-based flood prediction with integrating SHAP technique. Ecological Informatics, 84, 102904. https://doi.org/10.1016/j.ecoinf.2024.102904
29. Sanz-Cruzado, J., Droukas, N., McCreadie, R. (2024). FAR-Trans: An Investment Dataset for Financial Asset Recommendation. IJCAI-2024 Workshop on Recommender Systems in Finance (Fin-RecSys). https://doi.org/10.48550/arXiv.2407.08692