Financial time-series forecasting using a semi-supervised learning predictor with attention mechanism, LSTM architecture, and Monte Carlo intervention

Lukas Beckenbauer
TUM School of Management
lukas.beckenbauer@tum.de

31.03.2025

Abstract

Effective and accurate time-series forecasting with financial datasets is a challenging machine learning problem. Existing approaches are often affected from insufficient or incomplete data sets, difficulty for the model to converge and/or generalize, overfitting, and model underperformance. Financial time-series forecasting models therein hold a special relevance, as accurate predictions are directly coupled with increased understanding of market movements, which can translate into achieving market advantage and higher likelihood to achieve stock returns. Given constraints in financial data, such as insufficient data availability for model training, as well as the need to establish accuracy in price forecasting, this study implements a semi-supervised LSTM with Attention Mechanism and Monte Carlo intervention and tests the feasibility of this approach for real-time production environments.

Introduction

Effective and accurate time-series forecasting with financial datasets is a challenging machine learning (ML) problem (Li et al., 2020). Existing approahces are often affected from insufficient or incomplete data (Nguyen & Ha, 2024), difficulty for the model to learn and generalize, overfitting (Yang et al., 2024), and model underperformance (Masoud, 2018). Financial forecasting models therein hold a special relevance, as accurate predictions are directly coupled to increased understanding of market movements (Li et al., 2020), allowing for a higher likelihood to achieve market returns. Further, in real-world market scenarios, as ,e.g., during stock trading, the available data is often incomplete and the target variable is the future price of a given stock upon market close, or on the next day (Kamalov et al., 2021; Lin & Sun, 2021; Nguyen & Ha, 2024). Given these constraints in financial data, such as insufficient data availability for model training, as well as the need to establish accuracy in price forecasting, this paper proposes a semi-supervised long-term short-term memory network (LSTM) with Monte Carlo Attention Mechanism (MCAM-LSTM) and tests the feasibility of this approach for real-time production environments.

To accommodate highly non-linear phenomena, such as market outcomes, and to nudge the model to generalize and improve predictions on unbalanced and previously unseen data, various studies have emphasized the benefits of Attention Mechanism enhanced LSTM architectures (AM-LSMT). AM-LSMTs have been applied to predicting stock closing prices (Zhang & Zhang, 2020), market trends (Li et al., 2020; Masoud, 2018), and price analysis (Li et al., 2023). However, in financial forecasting, standard Attention Mechanisms tend to overemphasize short-term dynamics, - such as daily price volatility-while omitting long-term patterns, such as macroeconomic trends or seasonal effects (Sang & Li, 2024). Consequently, they are often ineffective in detecting trends that contribute to financial performance over the long-run, as they fail to generalize well on new, unseen forms of data (Zhang & Zhang, 2020).

To mitigate these shortcomings, and drawing on the work of Yang, Zhang, and Wang (2024), I propose in this study to integrate the Monte Carlo method as additional input filter within the AM-LSTM. Monte Carlo methods have been shown to mitigate the local overtraining effect of the Attention Mechanism, by introducing perturbations into the training output that help the model minimize loss more effectively and converge faster (Kim & Ko, 2022; Martin et al., 2020; Yang et al., 2024). Subsequently, they improve the ability of the model to generalize on new data and learn from previously unseen patterns (Zhang & Zhang, 2020). However, to date there are no studies that test the performance of the Monte Carlo-infused Attention Mechanism on training models that predict highly non-linear and complex data structures as they describe the dynamics of financial markets. This study will then assess the extent to which a locally-infused Monte Carlo implementation can enhance effectiveness of AM-LSTM predictive networks.

To facilitate the approach, I draw on cryptocurrency market data, representing data on market trends, as well as social media and trading engagement data on 40 distinct currencies in the timeframe from 01-01-2020 to 31-03-2024. I define my target variable "price movement", as a binary outcome (Rise / Fall), by the following formula:

$$\hat{y} = \begin{cases} 1, & \text{if } \hat{\alpha}_{t,30} > 1 \quad \text{(Rise)} \\ 0, & \text{if } \hat{\alpha}_{t,30} < 1 \quad \text{(Fall)} \end{cases}$$

Where daily return $\hat{\alpha}_t$ is defined as

$$\hat{\alpha}_t = \frac{\text{total marketcap organization i on day t}}{total\ marketcap\ organization\ i\ on\ previous\ day\ t-1}$$

and the 30-day risk adjusted daily return is defined as

$$\hat{\alpha}_{t,30} = \hat{\alpha}_t - \frac{1}{30} \sum_{j=t-30}^{t-1} \hat{\alpha}_j$$

Subsequently, I train the model in a semi-supervised setting with 60%, 80% and 100% labeled data over 10 iterations in total. I find that, despite achieving higher prediction accuracy with the non-enhanced AM-LSTM, the MCAM-LSTM moderately outperforms the former in correctly classifying on unseen test data. Further, as suggested by previous research, MC-AM contributes to reductions in overfitting and more balanced class detection. However, in comparison to AM-LSTM, the ability of the MCAM-LSMT to generalize on previously unseen test-data does not improve significantly over time. However, this effect could likely be mitigated by more training data, as well as a more sophisticated Attention Architecture (e.g. Multi-Head Attention (Lin & Sun, 2021; Nguyen & Ha, 2024)), and an extended exposure to training iterations. Future research should investigate how Monte Carlo enhanced Attention improves predictive performance with extended iterative training and how the benefits of Monte Carlo Attention could be utilized to improve model ability to perform on unseen data.

Related Work

The application of semi-supervised ML forecasting is still a largely underexplored field (Yang et al., 2024). Research on financial forecasting has utilized fully and semi-supervised deep learning architectures, such as RNN and LSTM models. Especially, LSTM models (an RNN variant) have therein been suggested as an appropriate fit, as their ability to learn long-term dependences in data via recursive feedback gates (Li et al., 2020), mitigates vanishing gradient effects as they are more prone in general RNN models (Li et al., 2020, 2023). However, training neural networks on a large data set with a high magnitude of parameters can cause the model to overfit by learning global patterns in the training data, omitting nuances or local differences, as they tend to be more prevalent and relevant in day-to-day price movement data (Martin et al., 2020; Sang & Li, 2024). To mitigate this effect and enhance financial forecasting accuracy, the implementation of an attention mechanism as part of the deep learning architecture has been proposed (Zhang & Zhang, 2020).

Attention mechanisms are a class of prediction models that generate class weights which selectively attend to relevant information, while reducing the impact of less irrelevant data and noise (Sang & Li, 2024; Yang et al., 2024). They are inspired by how the human brain allocates selective attention, as, e.g., via eye movement. However, with time-series data, the more subtle, contextual influence of long-term dynamics is easily discarded at the expense of more imminent, local time-series dynamics that privilege performance of short-term predictions. Subsequently, attention mechanisms are prone to short-term overfitting, introducing localized, (or short-term) bias into the model when analyzing vectors that contain long-term effects (Sang & Li, 2024). Especially for model training on large financial datasets, this tendency presents a problem: past occurrences in financial patterns often have low explainability fit in predicting the future outcomes of price movement, due to the highly exogenous nature of financial markets (e.g. Multi-Head Attention (Lin & Sun, 2021; Nguyen & Ha, 2024)). Instead, seemingly random occurrences in real-world events, such as changes in political climate, expressions of opinion leaders, news headlines, or even symbolisms of nation's financial departments, play a larger role in market reactions and should be accommodated when trying to achieve more accurate financial prediction models (Nguyen & Ha, 2024).

To mitigate local overfitting effects of the Attention Mechanism, I further implement a Monte Carlo mechanism as part of the attention-enhanced LSTM network, as it has been proposed by Yang, Zhang, and Wang (2024). The Monte Carlo method is a statistical procedure to calculate the distribution of function outputs, assess their empirical distribution and approximate the likeliest high-confidence variables by computing the arithmetic mean of all samples available (Murphy, 2014). In combination with the AM-LSTM, its stochastic sampling technique acts as a regularizer, and is expected to disrupt overfitting by nudging the model to explore alternative weightings of feature influence at time step t.

In this study Monte Carlo approximation has been combined with an AM-LSTM to overcome AMlimitations and train the model to better detect long term trends and improve overall performance on previously unseen market data. The full architecture of this approach will be outlined in the following section.

Architecture, Data, & Modelling Approach

I implement an LSTM network architecture with a softmax (soft)-attention mechanism and Monte Carlo refinement. Drawing on Yang et al., 2024, the soft-attention mechanism is computed as:

$$\text{att}(X, q) = \sum_{n=1}^{N} a_n x_n,$$

where $X = \{x_1, x_2, \ldots, x_N\}$ represents the input sequence, and a_n are the attention weights. These weights are derived from the hidden states of the LSTM using a softmax function:

$$A = \text{softmax}(W_a \tanh(V_a H)),$$

$$C = AH,$$

where $H \in \mathbb{R}^{N \times d}$ is the matrix of hidden states to which attention weights will be assigned. The attention weights A reflect the internal correlations among the elements of H. And $W_a \in \mathbb{R}^{1 \times k}$ and $V_a \in \mathbb{R}^{k \times d}$ are the weight matrices, while (tanh) activation introduces non-linearity to the attention scoring. The softmax function normalizes the scores to produce a probability distribution over timesteps. C represents the weighted matrix.

The LSTM network holds a bi-directional design with 2 hidden layers and 128 nodes per layer. Further, I implement 0.5 dropout at layer 1 and 0.3 dropout at layer 2 with subsequent ReLU activation. The output layer consists of a sigmoid function and provides binary output that is converted into a “Rise” or “Fall” classification.

Forward propagation in layer 1 is implemented as

Input gate: $i_t = \sigma(W_{\text{xi}} X_t + W_{\text{hi}} h_{t-1} + b_i)$

Forget gate: $f_t = \sigma(W_{\text{xf}} X_t + W_{\text{hf}} h_{t-1} + b_f)$

Output gate: $o_t = \sigma(W_{\text{xo}} X_t + W_{\text{ho}} h_{t-1} + b_o)$

Activation: $g_t = tanh(W_{\text{xg}} X_t + W_{\text{hg}} h_{t-1} + b_g)$

Cell state: $c_t = f_t \bigodot c_{t-1} + i_t \bigodot g_t)$

Hidden state: $h_t o_t = \tanh \bigodot (c_t)$

Output: $h_t^{(1)'} = ReLU(Dropout\left(h_t^{(1)}\right))$

Whereas the second LSTM layer follows the structure of Layer 1, but its output

$$h_t^{(2)'} = \text{ReLU}\big(\text{Dropout}(h_t^{(2)},\, p = 0.3)\big)$$

undergoes Monte Carlo-optimized Attention, where:

Monte Carlo Attention: $H^{(2)}_{\text{perturbed}} = H + \ \epsilon, \ \epsilon \sim N(0, 0.01)$

Attention weights: $\alpha_t = softmax(W_\alpha \tanh\left(V_a H_{perturbed,t}\right))$

Context vector: $C = \sum_t \alpha_t H_{perturbed,t}$

And the final prediction is: $\hat{y} \ = \ \sigma(FC\left(C*\right))$

Lastly, Monte Carlo Attention is computed as:

Peturbation: $H^{(m)}_{\text{perturbed}} = H + \ \epsilon^{(m)}, \ \epsilon^{(m)} \sim N(0, 0.01)$

Attention score: $e_t^{(m)} = \ W_\alpha tanh(V_\alpha H^{(m)}_{perturbed,t})$

Attention weights normalization: $\alpha_t^{(m)} = \ \frac{exp(e_t^{(m)})}{\sum_j exp(e_j^{(m)})}$

Context vector: $C = \sum_t \alpha_t^{(m)} H^{(m)}_{perturbed,t}$

Prediction per Monte Carlo sample: $\hat{y}^{(m)} = \ \sigma(FC\left(C^{(m)}\right))$

$$C^* = \text{argmin}_{C(m)} L(\hat{y}^{(m)}, y)$$

I run multiple test runs of network predictions per time t and use Monte Carlo to calculate the arithmetic mean of these network outputs. Consequently, the attention mechanism is fed with more informed data inputs, wherein the expected effect is that class imbalances will be addressed with greater accuracy.

The integration of the attention mechanism into the LSTM setup is then shown in Figure 1.

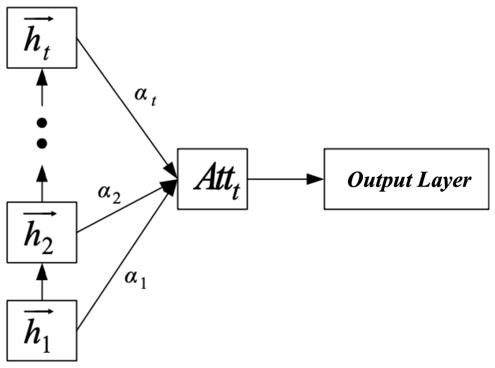


Fig. 1. Integration of the Attention Mechanism with the LSTM. Adapted from (Yang et al., 2024)

Whereas a simplified setup of the bi-directional MCAM-LSTM architecture is depicted in Figure 2.

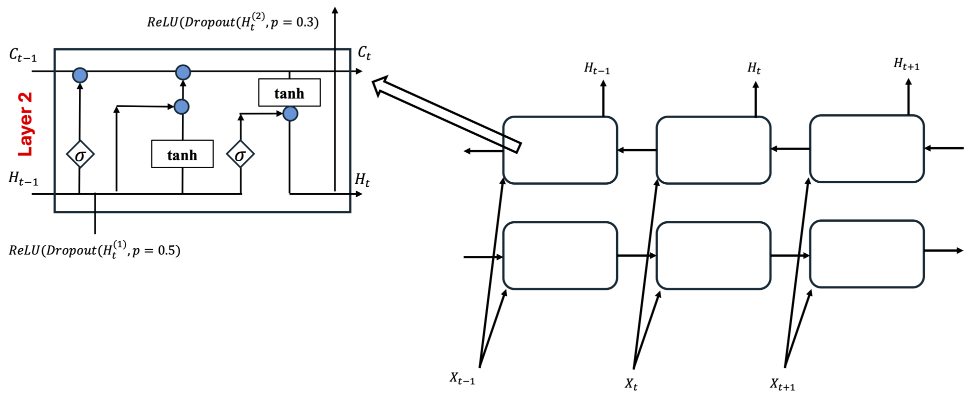


Fig. 2. Simplified overview of bi-directional MCAM-LSTM workflow

A general overview of an LSTM cell is shown in Figure 3.

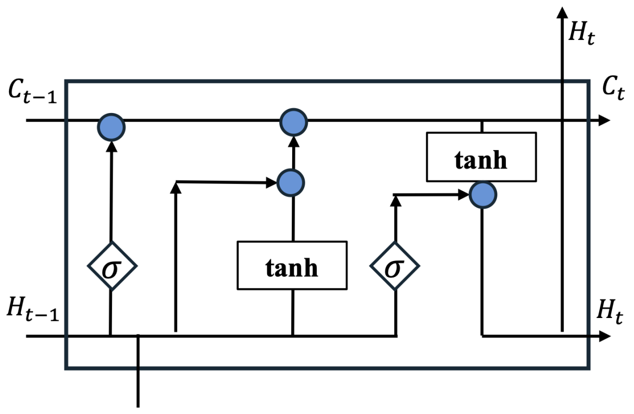
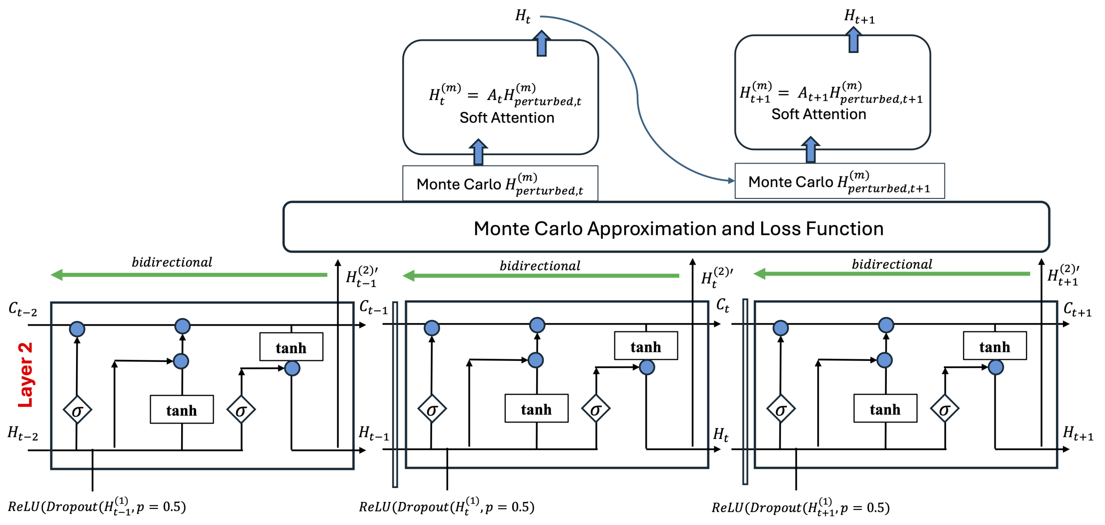


Fig. 3. Simplified overview of regular LSTM cell

And the full depiction of the MCAM-LSTM workflow is shown in Figure 4.



H_t
$H_t^{(m)} = A_t H^{(m)}_{perturbed,t}$
Soft Attention
Monte Carlo $H^{(m)}_{perturbed,t}$
H_{t+1}
$H_{t+1}^{(m)} = A_{t+1} H^{(m)}_{perturbed,t+1}$
Soft Attention
Monte Carlo $H^{(m)}_{perturbed,t+1}$
Monte Carlo Approximation and Loss Function
bidirectional
bidirectional
bidirectional
$H_{t-1}^{(2)'}$
$H_t^{(2)'}$
$H_{t+1}^{(2)'}$
Layer 2
C_{t-2}
C_{t-1}
C_t
C_{t+1}
H_{t-2}
H_{t-1}
H_t
H_{t+1}
tanh
σ
$ReLU(Dropout(H_{t-1}^{(1)}, p = 0.5)$
$ReLU(Dropout(H_t^{(1)}, p = 0.5)$
$ReLU(Dropout(H_{t+1}^{(1)}, p = 0.5)$

Fig. 4. Full overview of the implemented MCAM-LSTM architecture

Model Training

I train my model in a semi-supervised setting over 10 iterations for 60%, 80%, and 100% of labeled training data on a daily time-series dataset containing price and social engagement data on 40 different crypto market assets in a timeframe between 01-01-2020 to 01-03-2024 and holding more than 600.000 observations in total. I assess model performance by computing accuracy, area under the curve (AUC), and log loss for each iteration. Subsequently, I compute the final model evaluation on a separate dataset with 1200 observations of unseen test data, containing 30 days of consecutive timeseries data from 01-03-2024 to 01-04-2024 per organization, once training over all iterations has been completed. All results of model training are listed in Table 1 and Table 2 in the Appendix.

Hardware and Schedule

I train the above model on an Amazon Web Services Instance with 1 core NVIDIA T4 GPU and an 8GB RAM Quad Core CPU.

Optimizer

I employ the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.98$ and an adaptive learning rate according to the formula:

$$lrate = \ d_{\text{model}}^{-0.5} * \min\left(\text{step}_{\text{num}}{}^{-0.5}, step_{\text{num}} * \quad \text{warmup}_{\text{steps}}{}^{-1.5}\right) \ ,$$

Where $warmup\_steps = \frac{t}{step\_size\_increase}$, with step_size_increase $= 10$, the base_lr $= 0.0001$ and max_lr $= 0.001$.

Regularization

I employ three different methods for regularization:

Dropout

Dropout layers are implemented after each LSTM layer, with 0.5 dropout following layer 1 and 0.3 dropout following layer 2. Several tests have found these dropout rates to be the most productive for achieving convergence.

Early Stopping

The model uses early stopping based on validation loss. Training for a given organization is stopped and moved to the next set of observations, once validation loss does not improve for a consecutive five epochs in a row.

Gradient Clipping

To mitigate exploding gradients, gradients are clipped at a maximum norm of 1.0.

Semi-Supervised Model Training

The semi-supervised learning framework in this study iteratively enhances model performance by integrating labeled and unlabeled data. In each iteration $k \in \{1, \ldots, K\}$, the model, initially trained on a labeled subset $\mathcal{D}_L = \{(x_i, y_i)\}_{i=1}^{N_L}$, generates pseudo-labels $\hat{y}_j$ for the unlabeled data $\mathcal{D}_U = \{x_j\}_{j=1}^{N_U}$. These pseudo-labels are filtered using a confidence threshold $\tau = 0.8$ and variance threshold $\sigma^2 = 0.05$, such that $\hat{y}_j = 1$ if $\mathbb{E}[\hat{p}(x_j)] > \tau$ and $\mathrm{Var}[\hat{p}(x_j)] < \sigma^2$, or $\hat{y}_j = 0$ if $\mathbb{E}[\hat{p}(x_j)] < 1 - \tau$ and $\mathrm{Var}[\hat{p}(x_j)] < \sigma^2$; otherwise, $\hat{y}_j = -1$. High-confidence samples $\mathcal{D}_P = \{(x_j, \hat{y}_j) \mid \hat{y}_j \neq -1\}$ are then concatenated with $\mathcal{D}_L$, forming an augmented training set $\mathcal{D}_A = \mathcal{D}_L \cup \mathcal{D}_P$. The model is retrained on $\mathcal{D}_A$, minimizing the binary cross-entropy loss $\mathcal{L} = -\frac{1}{|\mathcal{D}_A|} \sum_{(x,y)\in\mathcal{D}_A} [y \log(\hat{p}(x)) + (1-y) \log(1-\hat{p}(x))]$, thereby improving its generalization by iteratively incorporating reliable pseudo-labeled data over $K = 10$ iterations. As previously stated, semi-supervised model training is executed for 60%, 80% of labeled training data. While training on 100% of labeled data is used to assess the model's baseline performance.

Results

The modelling results show that the MCAM-LSMT moderately outperforms the regular model with achieving 0.76 accuracy and 0.754 AUC after 10 iterations, when compared to 0.725 accuracy and 0.746 AUC of the AM-LSMT for 60% of labeled data and 0.60 accuracy and 0.649 AUC for the MCAM-LSMT with 80% of labeled data after 10 iterations, compared to 0.615 accuracy and 0.654 AUC for the AM-LSMT. Further, MC-enhanced training moderately improves log-loss reduction (from 0.997 to 0.632 after 10 iterations for the AM-LSTM vs. from 1.022 to 0.567 for MC-models), as well as AUC growth with increasing iterations. As the models trained with 60% of labeled data perform with the highest accuracy, I will refer to these in the following, only.

The MC-model appears to be slightly better in class detection during early iteration stages. But, the AM-LSTM appears to perform better in improving class-detection with growing number iterations: While the MCAM-LSTM model does not improve significantly over time, the AM-LSTM improves Rise-Class detection from 0.17 after iteration 1 to 0.27 after the 10th iteration of learning with training data. However, upon validating the models after 10 iterations on unseen data, the AM-LSTM tends to outperform MCAM-LSMT sampling after 10 iterations: while the most performant

MC-enhanced model generalizes poorly at roughly 0.54 accuracy, the most performant non-MC model generalizes at about 0.59 percent accuracy.

Further, the improved effect in class detection for AM-LSTM during training reverses when the models are exposed to previously unseen data. Here, the 0.6 and 0.8 semi-supervised AM-LSTM models fail to detect the Rise class at all, while the MCAM-LSTM performs moderately in Rise-classification precision. Still, the latter also fails to detect the Rise class during Recall. This hints at the fact, that AM-LSTM overtrains on Fall-class detection, whereas MC approximation holds the potential of weakening this effect. And further, that Recall detection could be improved with advanced MC sampling as shown, e.g., in (Yang et al., 2024). To fully test this hypothesis, the underlying models would need to be trained on a larger number of iterations (e.g. n=100 and beyond), which has been beyond the scope of this study.

Lastly and surprisingly, the above results suggest that, performance contributions of either Attention Mechanism (with and without MC) become more pronounced when more unlabeled data samples are available for training. If the model is trained over iterations with only 60% of pre-labeled data, convergence as well as growth in accuracy and AUC is highest, whereas the effect of iterations based on 80% of pre-labeled training data is measurably lower. This dynamic might hint at the fact that Attention Mechanisms generally perform better if input data allows for certain 'degrees of freedom' when computing the most likely output variables that improve convergence (Li et al., 2024). All results and findings are listed in Table 2 in the Appendix. A comparative overview of model performances is provided in Figures 3A, 3B, and 3C.

Discussion & Conclusion

While MC integration to the AM-LSTM does not significantly contribute to improved prediction accuracy on previously unseen data, my results show that the integration of Monte Carlo approximation as part of the Attention Mechanism in LSMT holds promising potential for steadily improving overall model accuracy during training, to contribute to improved detection of the minority class, and subsequently to enhance the utility of LSTM prediction models for complex time series data, as, e.g., it is utilized for financial predictions.

Future improvements of the MC-AM implementation should focus on advancing additional accuracy gains of MC approximation, and on extending MC sampling towards a more balanced approach to minority class detection within AM architectures. More specifically, integrating MC approximation across multiple, connected layers of the LSMT network, instead of the localized output layer as implemented in this study, might further contribute to improved classification dynamics by further diversifying the pool of data samples to learn from and therefore expanding the scope of local minima.

References

[1] Kamalov, F., Gurrib, I., & Rajab, K. (2021). Financial Forecasting with Machine Learning: Price Vs Return. *Journal of Computer Science*, *17*(3), 251–264. https://doi.org/10.3844/jcssp.2021.251.264

[2] Kim, H., & Ko, J. (2022). Fast Monte-Carlo Approximation of the Attention Mechanism (Version 1). *arXiv*. https://doi.org/10.48550/ARXIV.2201.12854

[3] Li, Y., Li, L., Zhao, X., Ma, T., Zou, Y., & Chen, M. (2020). An Attention-Based LSTM Model for Stock Price Trend Prediction Using Limit Order Books. *Journal of Physics: Conference Series*, *1575*(1), 012124. https://doi.org/10.1088/1742-6596/1575/1/012124

[4] Li, Y., Zhang, X., & Zhu, X. (2023). Application of LSTM and Attention Mechanism for Stock Price Prediction and Analysis. In K. Hemachandran, R. S. K. Boddu, & W. Alhasan (Eds.), *Proceedings of the 2023 2nd International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID 2023)* (Vol. 9, pp. 553–561). Atlantis Press International BV. https://doi.org/10.2991/978-94-6463-222-4_60

[5] Lin, H., & Sun, Q. (2021). Financial Volatility Forecasting: A Sparse Multi-Head Attention Neural Network. *Information*, *12*(10), 419. https://doi.org/10.3390/info12100419

[6] Martin, A., Ollion, C., Strub, F., Le Corff, S., & Pietquin, O. (2020). The Monte Carlo Transformer: A stochastic self-attention model for sequence prediction (Version 2). *arXiv*. https://doi.org/10.48550/ARXIV.2007.08620

[7] Masoud, M. (2018). Attention-based Stock Price Movement Prediction Using 8-K Filings. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15751328.pdf

[8] Murphy, K. P. (2014). *Machine Learning—A Probabilistic Perspective*. MIT Press.

[9] Nguyen, A., & Ha, S. (2024). A Lightweight Multi-Head Attention Transformer for Stock Price Forecasting. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4729648

[10] Sang, S., & Li, L. (2024). A Novel Variant of LSTM Stock Prediction Method Incorporating Attention Mechanism. *Mathematics*, *12*(7), 945. https://doi.org/10.3390/math12070945

[11] Yang, Y., Zhang, J., & Wang, L. (2024). Semi-supervised prediction method for time series based on Monte Carlo and time fusion feature attention. *Applied Soft Computing*, *167*, 112283. https://doi.org/10.1016/j.asoc.2024.112283

[12] Zhang, S., & Zhang, H. (2020). Prediction of Stock Closing Prices Based on Attention Mechanism. *2020 16th Dahe Fortune China Forum and Chinese High-Educational Management Annual Academic Conference (DFHMC)*, 244–248. https://doi.org/10.1109/DFHMC52214.2020.00053

Appendix: Model Results and Comparison Tables

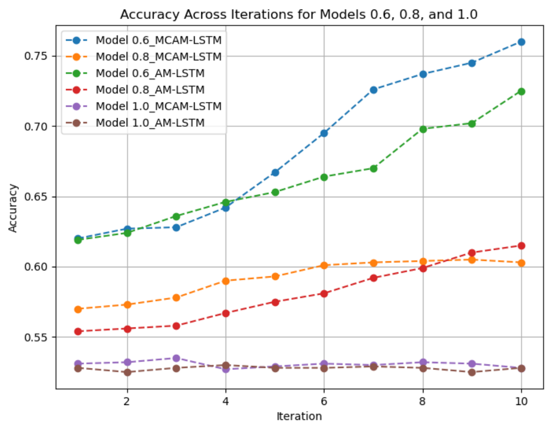


Fig. 5. Comparison of Accuracy per model

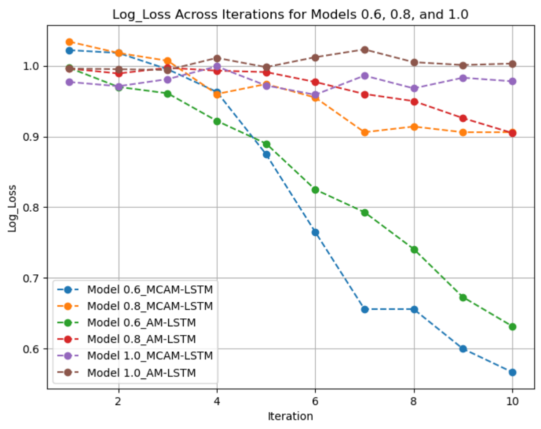


Fig. 6. Comparison of Log Loss per model

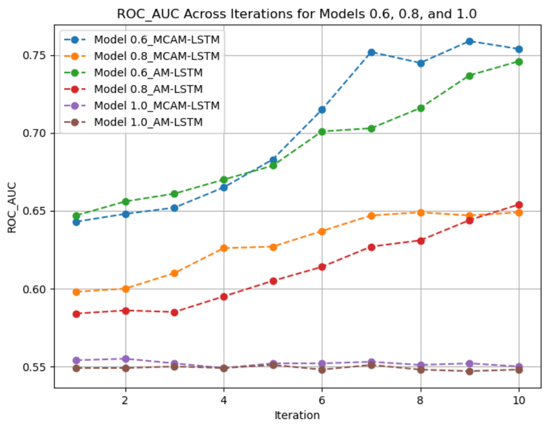


Fig. 7. Comparison of ROC AUC per model

Table 1. Classification Metrics Across Models and Iterations

Labeled Data (%)	Model	Iteration	Accuracy	ROC_AUC	Log_Loss	Precision Fall	Precision Rise	Recall Fall	Recall Rise	F1 Fall	F1 Rise	Support Fall	Support Rise
0.6	0.6_MCAM-LSTM	1	0.62	0.643	1.022	0.64	0.50	0.88	0.19	0.74	0.28	15919	9761
0.6	0.6_MCAM-LSTM	2	0.627	0.648	1.018	0.64	0.51	0.92	0.14	0.75	0.22	15733	9476
0.6	0.6_MCAM-LSTM	3	0.628	0.652	0.995	0.65	0.49	0.89	0.19	0.75	0.27	15340	9002
0.6	0.6_MCAM-LSTM	4	0.642	0.665	0.963	0.66	0.50	0.91	0.17	0.77	0.25	14674	8156
0.6	0.6_MCAM-LSTM	5	0.667	0.683	0.875	0.70	0.49	0.89	0.22	0.78	0.30	13701	6780
0.6	0.6_MCAM-LSTM	6	0.695	0.715	0.765	0.73	0.48	0.90	0.22	0.80	0.30	12275	5275
0.6	0.6_MCAM-LSTM	7	0.726	0.752	0.656	0.76	0.49	0.91	0.24	0.83	0.32	10708	4001
0.6	0.6_MCAM-LSTM	8	0.737	0.745	0.656	0.77	0.49	0.91	0.24	0.84	0.32	9836	3483
0.6	0.6_MCAM-LSTM	9	0.745	0.759	0.6	0.78	0.29	0.92	0.22	0.84	0.31	9516	3213
0.6	0.6_MCAM-LSTM	10	0.76	0.754	0.567	0.79	0.47	0.93	0.21	0.86	0.28	9082	2753
0.8	0.8_MCAM-LSTM	1	0.57	0.598	1.034	0.58	0.51	0.86	0.19	0.69	0.28	17544	13440
0.8	0.8_MCAM-LSTM	2	0.573	0.6	1.018	0.59	0.52	0.84	0.22	0.69	0.31	17513	13393
0.8	0.8_MCAM-LSTM	3	0.578	0.61	1.007	0.59	0.51	0.86	0.19	0.70	0.28	17030	12595
0.8	0.8_MCAM-LSTM	4	0.59	0.626	0.96	0.61	0.52	0.84	0.24	0.71	0.32	16441	11708
0.8	0.8_MCAM-LSTM	5	0.593	0.627	0.974	0.61	0.51	0.86	0.21	0.71	0.29	16072	11165
0.8	0.8_MCAM-LSTM	6	0.601	0.637	0.955	0.62	0.51	0.87	0.20	0.72	0.28	15669	10599
0.8	0.8_MCAM-LSTM	7	0.603	0.647	0.906	0.62	0.51	0.86	0.21	0.72	0.30	15488	10360
0.8	0.8_MCAM-LSTM	8	0.604	0.649	0.914	0.62	0.51	0.87	0.20	0.73	0.29	15374	10213
0.8	0.8_MCAM-LSTM	9	0.605	0.647	0.906	0.63	0.51	0.85	0.24	0.72	0.32	15323	10141
0.8	0.8_MCAM-LSTM	10	0.603	0.649	0.906	0.63	0.50	0.85	0.23	0.72	0.32	15236	10054
0.6	0.6_AM-LSTM	1	0.619	0.647	0.997	0.64	0.48	0.89	0.17	0.74	0.25	16016	9712
0.6	0.6_AM-LSTM	2	0.624	0.656	0.97	0.64	0.50	0.89	0.18	0.75	0.26	15945	9570
0.6	0.6_AM-LSTM	3	0.636	0.661	0.961	0.66	0.49	0.91	0.16	0.76	0.24	15396	8742
0.6	0.6_AM-LSTM	4	0.646	0.67	0.922	0.67	0.50	0.90	0.19	0.77	0.27	15006	8196
0.6	0.6_AM-LSTM	5	0.653	0.679	0.89	0.67	0.50	0.90	0.18	0.77	0.27	14530	7760
0.6	0.6_AM-LSTM	6	0.664	0.701	0.825	0.69	0.50	0.88	0.23	0.78	0.31	13709	6895
0.6	0.6_AM-LSTM	7	0.67	0.703	0.793	0.70	0.48	0.89	0.22	0.78	0.30	12980	6250
0.6	0.6_AM-LSTM	8	0.698	0.716	0.741	0.73	0.49	0.90	0.22	0.81	0.31	12087	5202
0.6	0.6_AM-LSTM	9	0.702	0.737	0.673	0.75	0.49	0.87	0.29	0.81	0.36	11523	4761
0.6	0.6_AM-LSTM	10	0.725	0.746	0.632	0.77	0.48	0.89	0.27	0.83	0.35	10835	3991
0.8	0.8_AM-LSTM	1	0.554	0.584	0.996	0.57	0.50	0.83	0.20	0.68	0.29	19113	15270
0.8	0.8_AM-LSTM	2	0.556	0.586	0.989	0.57	0.50	0.84	0.20	0.68	0.29	19043	15217
0.8	0.8_AM-LSTM	3	0.558	0.585	0.997	0.57	0.50	0.85	0.20	0.68	0.28	18989	15163
0.8	0.8_AM-LSTM	4	0.567	0.595	0.993	0.58	0.51	0.84	0.21	0.69	0.30	18358	14261
0.8	0.8_AM-LSTM	5	0.575	0.605	0.991	0.59	0.50	0.86	0.20	0.70	0.28	17400	12876
0.8	0.8_AM-LSTM	6	0.581	0.614	0.977	0.60	0.50	0.85	0.21	0.70	0.30	17015	12199
0.8	0.8_AM-LSTM	7	0.592	0.627	0.96	0.61	0.51	0.86	0.21	0.71	0.30	16604	11545
0.8	0.8_AM-LSTM	8	0.599	0.631	0.95	0.62	0.50	0.85	0.23	0.72	0.31	16077	10713
0.8	0.8_AM-LSTM	9	0.61	0.644	0.926	0.63	0.51	0.87	0.21	0.73	0.30	15717	10167
0.8	0.8_AM-LSTM	10	0.615	0.654	0.905	0.64	0.49	0.85	0.23	0.73	0.32	15286	9482
1.0	1.0_MCAM-LSTM	1	0.531	0.554	0.977	0.54	0.50	0.80	0.23	0.64	0.32	22759	20201
1.0	1.0_MCAM-LSTM	2	0.532	0.555	0.971	0.54	0.50	0.78	0.25	0.64	0.33	22759	20201
1.0	1.0_MCAM-LSTM	3	0.535	0.552	0.981	0.54	0.51	0.80	0.24	0.64	0.33	22759	20201
1.0	1.0_MCAM-LSTM	4	0.527	0.549	0.999	0.54	0.49	0.78	0.24	0.64	0.32	22759	20201
1.0	1.0_MCAM-LSTM	5	0.529	0.552	0.972	0.54	0.50	0.78	0.25	0.64	0.33	22759	20201
1.0	1.0_MCAM-LSTM	6	0.531	0.552	0.959	0.54	0.50	0.80	0.23	0.64	0.31	22759	20201
1.0	1.0_MCAM-LSTM	7	0.530	0.553	0.986	0.54	0.50	0.78	0.25	0.64	0.33	22759	20201
1.0	1.0_MCAM-LSTM	8	0.532	0.551	0.968	0.54	0.51	0.76	0.27	0.63	0.36	22759	20201
1.0	1.0_MCAM-LSTM	9	0.531	0.552	0.983	0.54	0.50	0.80	0.23	0.64	0.32	22759	20201
1.0	1.0_MCAM-LSTM	10	0.528	0.550	0.978	0.54	0.50	0.80	0.22	0.64	0.31	22759	20201
1.0	1.0_AM-LSTM	1	0.528	0.549	0.996	0.54	0.50	0.77	0.26	0.63	0.34	22681	20357
1.0	1.0_AM-LSTM	2	0.525	0.549	0.995	0.53	0.50	0.79	0.23	0.64	0.31	22681	20357
1.0	1.0_AM-LSTM	3	0.528	0.550	0.994	0.54	0.50	0.79	0.23	0.64	0.32	22681	20357
1.0	1.0_AM-LSTM	4	0.530	0.549	1.011	0.54	0.51	0.80	0.23	0.64	0.32	22681	20357
1.0	1.0_AM-LSTM	5	0.528	0.551	0.998	0.53	0.50	0.81	0.22	0.64	0.30	22681	20357
1.0	1.0_AM-LSTM	6	0.528	0.548	1.012	0.53	0.50	0.80	0.22	0.64	0.31	22681	20357
1.0	1.0_AM-LSTM	7	0.529	0.551	1.023	0.54	0.50	0.78	0.24	0.64	0.33	22681	20357
1.0	1.0_AM-LSTM	8	0.528	0.548	1.005	0.53	0.50	0.80	0.22	0.64	0.31	22681	20357
1.0	1.0_AM-LSTM	9	0.525	0.547	1.001	0.53	0.50	0.79	0.23	0.64	0.31	22681	20357
1.0	1.0_AM-LSTM	10	0.528	0.548	1.003	0.54	0.50	0.79	0.24	0.64	0.32	22681	20357

Table 2. Final Evaluations per Model on 30 days of Unseen Data after 10 iterations of training

Model	**Class**	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**	**Macro Avg F1**
0.6 MCAM-LSTM	Fall	0.55	0.98	0.70	662	0.54	0.36
	Rise	0.22	0.01	0.01	538		
0.8 MCAM-LSTM	Fall	0.53	1.00	0.69	629	0.53	0.35
	Rise	0.62	0.01	0.02	571		
1.0 MCAM-LSTM	Fall	0.53	0.81	0.64	630	0.53	0.47
	Rise	0.50	0.21	0.30	570		
0.6 AM-LSTM	Fall	0.59	1.00	0.74	706	0.59	0.37
	Rise	0.00	0.00	0.00	494		
0.8 AM-LSTM	Fall	0.55	1.00	0.71	660	0.55	0.35
	Rise	0.00	0.00	0.00	540		
1.0 AM-LSTM	Fall	0.52	0.86	0.65	631	0.51	0.42
	Rise	0.44	0.12	0.19	569		