A Reproducibility Report on the CLVSA Model

Charles Ran **Sky Yun**

Abstract

Reproducibility is a critical component of rigorous machine learning research. In this work, we attempt to reproduce the core experimental results of Wang et al. 2019 through an independent implementation. We provide an open-source PyTorch implementation of CLVSA and systematically evaluate its empirical performance. In addition, we conduct ablation studies to assess the contribution of key model components, including the variational component, and examine whether the reported improvements can be reproduced. We further explore small architectural extensions to investigate potential performance improvements.

Our results do not reproduce the performance gains reported in the original work, and ablation experiments suggest that certain components contribute less than expected under our implementation. We identify several potential sources of discrepancy, including underspecified implementation details and sensitivity to training setup. Finally, we discuss the strengths and limitations of the proposed approach.

1 Introduction

Reproducibility is essential for validating empirical claims in machine learning. However, prior work has shown that incomplete reporting of implementation details, hyperparameters, and data preprocessing steps can lead to discrepancies between reported and reproduced results (Semmelrock et al. 2025). Reproducibility studies therefore play a critical role in assessing the reliability and robustness of proposed methods.

Prior work by Wang et al. (2019) proposes a Convolutional LSTM-based Variational Seq2Seq model with Attention (CLVSA), which incorporates a variational latent space to explicitly model uncertainty in sequential financial data. The authors report improved predictive performance and generalization relative to deterministic baselines, attributing these gains to the stochastic representation learned through variational inference.

In this work, we conduct a reproducibility study of Wang et al. (2019), with the goal of independently validating their core claims. Beyond evaluating the proposed model, we perform ablation studies to examine the contribution of the model's main architectural components, including the variational component. We also evaluate small extensions to the original design, such as modified feature transformations and scaled attention, to assess whether these refinements improve empirical performance.

Our contributions are threefold: (1) we provide an independent PyTorch implementation of CLVSA, (2) we systematically evaluate the contribution of its key components, including the variational component, to assess whether the reported performance gains can be reproduced, and (3) we explore small architectural extensions to determine whether further improvements can be achieved.

2 Background and Similar Works

Financial time-series forecasting remains a fundamentally challenging problem due to a high degree of noise and the presence of latent stochastic factors that are difficult to model explicitly. As such,

Originally a course project for UBC CPSC 440/550, April 2026.

several deep-learning approaches have been utilized to predict financial trends and prices despite the inherent complexity of the problem.

Work by Pawar, Jalem, and Tiwari (2019) showed that a vanilla LSTM recurrent neural network can predict stock trends more accurately compared to traditional machine learning algorithms. From this, the authors in Md et al. (2023) proposed a multi-layered sequential LSTM to predict stock prices through context-specific dependencies. Another similar work by Su et al. (2024) used attention-based convolutional networks to predict trends in financial markets.

While the aforementioned approaches have demonstrated empirical success, Wang et al. (2019) asserts that their deterministic nature may limit their ability to capture the inherent uncertainty in financial markets. Thus, Wang et al. (2019) proposes incorporating stochastic latent states.

3 Model Implementation

We implement the proposed CLVSA model and other baseline models as described in the original paper in PyTorch (Wang et al. 2019). Our implementation is available at `https://github.com/rancharles/clvsa`.

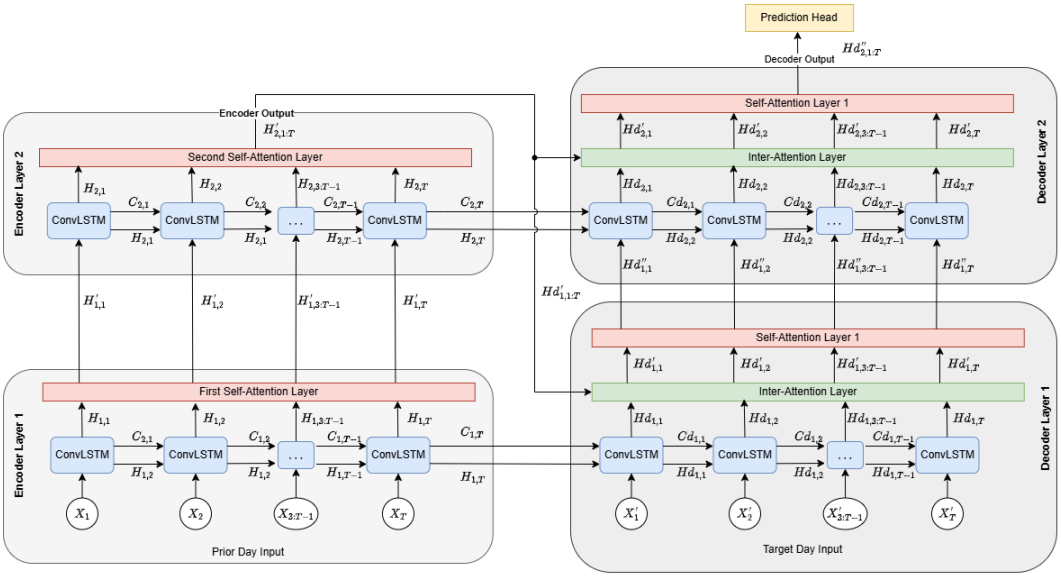

Figure 1: Interpretation of the CLSA architecture with ConvLSTM cells, self-attention, and inter-attention. CLVSA extends this model with a variational backward decoder.

The model is based on a sequence-to-sequence architecture with convolutional LSTM units, self-attention, inter-attention, and a variational backward decoder. The encoder and decoder each receive 2-D financial data frames corresponding to two consecutive trading days. The encoder processes the first day, while the decoder models the second day using the encoder representation. CLVSA further introduces a backward decoder over the reversed second segment to construct an approximate posterior distribution and generate a KL-divergence regularizer (Wang et al. 2019).

3.1 Convolutional LSTM Seq2Seq Design

On a high level, the model uses a sequence-to-sequence encoder-decoder architecture (Sutskever, Vinyals, and Le 2014). The recurrent component is based on convolutional LSTM (convLSTM) units. Unlike a standard LSTM, whose gates are parameterized by fully connected transformations (Hochreiter and Schmidhuber 1997), a ConvLSTM replaces these transformations with convolutional kernels (Shi et al. 2015). Conceptually, this allows the recurrent cell to first extract local temporal patterns from input frames. Each input $X_t \in \mathbb{R}^{5\times 6}$ represents a short window of market data, where

the rows correspond to the five OHLCV features (open, high, low, close, volume), and the columns correspond to six consecutive 5-minute time intervals.

Following Wang et al. (2019), we use 1×3 convolutional kernels that only move horizontally across columns. We use 32 output channels as in the original design, where each channel shares its kernel across all feature rows. In addition, we experiment with a row-specific variant where each channel has its own convolutional kernel. The motivation is that the different OHLCV features may exhibit different pattern dynamics.

3.2 Attention Mechanisms

The model incorporates both inter-attention and self-attention to capture dependencies across and within trading segments (Bahdanau, Cho, and Bengio 2014; Cheng, Dong, and Lapata 2016; Wang et al. 2019). Inter-attention connects the encoder and decoder by allowing each decoder state to attend over the encoder's hidden states, enabling the model to condition predictions on relevant information from the previous segment. Self-attention is applied within each sequence to emphasize informative time steps.

In contrast to the proposed model, which uses unscaled dot-product attention, we adopt scaled dot-product attention. In our experiments, this scaling prevents the attention distribution from being overly sharp and leads to improved performance.

3.3 Variational Backward Decoder

The key extension from CLSA to CLVSA is the variational backward decoder. The motivation is that supervised labels such as *up*, *flat*, and *down* only provide a discretized summary of market movement. They do not fully preserve the variability contained in the original OHLCV sequence. To alleviate this bias, CLVSA introduces stochasticity into the latent states and regularizes the model using a KL-divergence term inspired by variational autoencoders (Kingma and Welling 2013; Rezende, Mohamed, and Wierstra 2014; Wang et al. 2019).

The forward decoder defines a prior distribution over latent variables,

$$p_\theta(z_t \mid x_{1:t}, z_{1:t-1}),$$

while the backward decoder defines an approximate posterior,

$$p_\phi(z_t \mid x_{T:t}, y'_t) = \mathcal{N}(\mu_t, \mathrm{diag}(\sigma_t^2)).$$

The latent variable is sampled using the reparameterization trick:

$$z_t = \mu_t + \sigma_t \circ \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, I).$$

z_t is sampled from the approximate posterior during training, and from the learned prior at evaluation and test time. (Wang et al. 2019).

3.4 Objective Function

The CLVSA objective combines three terms: the forward decoder prediction loss, the backward decoder prediction loss, and a KL-divergence penalty between the approximate posterior and the learned prior. We write the minimization objective as

$$\mathcal{J} = \mathcal{L}^{\mathrm{forward}} + \alpha \mathcal{L}^{\mathrm{backward}} + \beta D_{\mathrm{KL}} \left(p_\phi(z \mid x, y) \parallel p_\theta(z \mid x) \right)$$

Here, $\mathcal{L}^{\mathrm{forward}}$ is the main forward decoder classification loss,

$$\mathcal{L}^{\mathrm{forward}} = -\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{p_\phi(z_{1:t} \mid x_{T:t}, y'_t)} \left[\log p_\theta(y_t \mid x_{1:t}, z_{1:t}) \right],$$

which in implementation is approximated by sampling z and then computing the usual cross-entropy over the decoder outputs. Similarly, $\mathcal{L}^{\mathrm{backward}}$ is an auxiliary cross-entropy loss from the backward decoder over the reversed target sequence, controlled by a constant $\alpha = 2.5 \times 10^{-4}$ as in Wang et al. (2019). The KL term regularizes the approximate posterior $p_\phi(z \mid x, y)$ toward the learned prior $p_\theta(z \mid x)$ at each decoder time step, with β used as a KL annealing weight (Bowman et al. 2015; Wang et al. 2019). The original CLVSA objective also includes an explicit L_2 regularization term. In our implementation, we omit this term and instead use weight decay through AdamW, so weight regularization is handled by the optimizer rather than directly added to the loss.

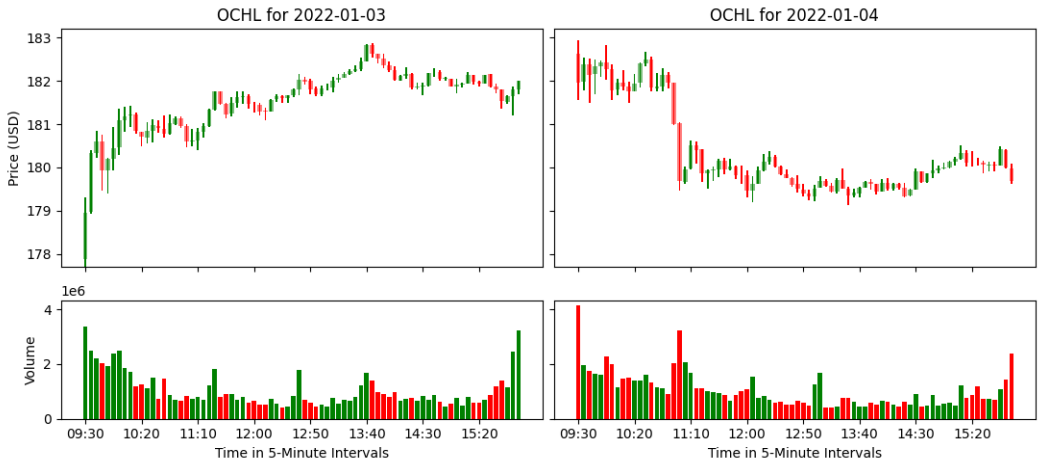

Figure 2: The OCHLV of two-consecutive days of AAPL stock starting from January 3rd, 2022

4 Methodology

4.1 Experimental Objectives

The primary objective of this study is to reproduce the key findings of Wang et al. (2019), specifically the reported performance improvements of CLVSA over common baseline models such as LSTM for financial time-series prediction, as well as the contribution of the variational component. Our secondary objective is to evaluate small extensions of CLVSA to assess whether further performance improvements can be achieved.

To this end, we train, evaluate, and compare CLVSA and its variants on two financial datasets similar to those used in the original work.

4.2 Datasets

Because the original datasets are not publicly released, we use publicly accessible Hugging Face OHLCV datasets. For futures data we use BTCUSDT and for equities we use AAPL stock, both from 2021–2024. Figure 3 shows one particular example of AAPL OHLCV information.

Following Wang et al. (2019), we first aggregate the data into 30-minute frames. We then deviate from the original setup by transforming the raw OHLCV values into stationary features before constructing model inputs. For each frame t, the open, high, low, and close prices are expressed relative to the previous close price:

$$\tilde{o}_t = \log\left(\frac{o_t}{c_{t-1}}\right), \quad \tilde{h}_t = \log\left(\frac{h_t}{c_{t-1}}\right), \quad \tilde{l}_t = \log\left(\frac{l_t}{c_{t-1}}\right), \quad \tilde{c}_t = \log\left(\frac{c_t}{c_{t-1}}\right).$$

For volume, we use a log transform,

$$\tilde{v}_t = \log(1 + v_t).$$

These transformed values form the five input features used by the model. We found that these stationary transformations are essential for reducing regime drift, as they remove dependence on the absolute price levels.

Labels are constructed following Wang et al. (2019) by thresholding the next-frame close-price log return $\tilde{c}_{t+1}$. Observations above $b_{\text{up}} = \log((\mu_c + \lambda)/\mu_c)$ are labeled *up*, observations below $b_{\text{down}} = \log((\mu_c - \lambda)/\mu_c)$ are labeled *down*, and the rest are labeled *flat*. We choose λ to approximately balance the three classes.

4.3 Training Procedure

Each model is trained to classify the trend of the next 30-minute interval as *up*, *flat*, or *down*, given all preceding 30-minute frames. Each frame contains normalized OHLCV features as described in the previous subsection.

For each training run, we use a contiguous 3-year window starting from January 1 of the first year to December 31 of the last year. The final 200 days are reserved for validation and testing, split into consecutive 100-day validation and test sets. The remaining data is used for training. For each training run, we select the threshold parameter λ to approximately balance the class labels in the training set. Each training example consists of two consecutive days. Each example is a contiguous two-day window: the first day forms the encoder source sequence, and the second day forms the decoder target sequence.

We employ early stopping based on validation balanced accuracy (BACC), with a minimum of 8 epochs and a patience of 6 epochs. Models are trained using AdamW with a learning rate of 3×10^{-4} and weight decay of 1×10^{-4}. The final model is selected based on the epoch achieving the highest validation BACC.

Following Wang et al. (2019), we report mean average precision (MAP) as the primary evaluation metric. In practice, we find that BACC provides a more stable signal for model selection during training. All experiments are conducted on an RTX 3060 Ti GPU.

4.4 Model Variants

We train a plain LSTM baseline for comparison with CLSA and CLVSA. We also evaluate ablations of CLSA to isolate the effects of its convolutional and attention components:

- **Attention**: We remove inter-attention, self-attention, or both attention mechanisms.
- **Convolutions**: We replace ConvLSTM cells with standard LSTM cells. We also evaluate a row-specific ConvLSTM variant, where each OHLCV feature row has its own convolutional kernels.

5 Experimental Results

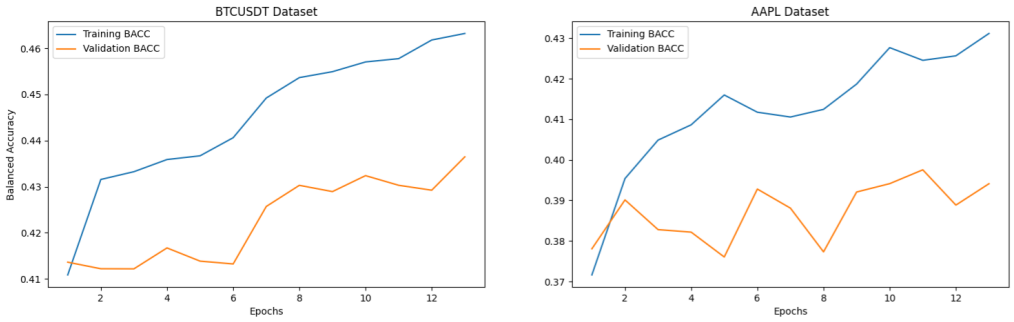

Figure 3: Validation balanced accuracy over training epochs for CLVSA on the BTCUSDT and AAPL datasets from 2022–2024.

Table 1: Experimental results of the testing MAP (%) corresponding to each trained model and dataset. CLS and LSA refer to the CLSA model without attention entirely and the convolutional part of LSTM cell, respectively.

	Model				
Dataset	CLVSA	CLSA	LSTM	CLS	LSA
BTCUSDT (Futures)	43.3	43.2	42.9	43.2	43.0
AAPL (Equity)	41.2	40.6	40.0	40.6	40.2

The results of the ablation studies suggest that both inter-attention and self-attention provide only small gains from 0.1 to 0.2 MAP. The ConvLSTM cell gives a modest improvement over standard LSTM cells of approximately 0.4 MAP. Our row-specific convolutional kernel extension did not improve performance, suggesting that shared kernels across OHLCV rows are sufficient.

Overall, our experimental results do not reproduce the large performance gains reported for CLSA and CLVSA over basic baselines such as LSTM. Table 1 reports the test MAP for each model and dataset. CLSA achieves only modest improvements over the LSTM baseline, and adding the variational component provides limited additional improvement around 0.5 MAP. In particular, we are unable to reproduce the consistent 2.0 MAP gains of CLVSA from CLSA nor reach the 45–50 MAP range reported by Wang et al. (2019).

However, we do observe that the variational regularizer improves training consistency, producing more stable validation performance across epochs. We also find that CLSA and CLVSA improve balanced accuracy (BACC) substantially. The LSTM baseline achieves a BACC (%) of 36.3 on the AAPL dataset. CLSA improves BACC by 2.9 over LSTM, and CLVSA provides an additional 0.7 BACC improvement over CLSA. This is consistent with our use of validation BACC for early stopping, which also produced the best test MAP results in our experiments.

6 Conclusion

The aim of this paper is to reproduce the performance improvements reported for the CLSA and CLVSA models in Wang et al. (2019). We evaluate CLSA, CLVSA, and several variants on AAPL and BTCUSDT datasets.

Reproducing the results of Wang et al. (2019) proved challenging, partly because several implementation and training details are underspecified. For example, the paper specifies that both attention mechanisms are used, but does not fully determine whether self-attention is applied before or after inter-attention, which affects the hidden states passed through the decoder.

Our results show that the vanilla LSTM baseline performs competitively in terms of test MAP, with CLSA and CLVSA producing only marginal improvements. This differs from Wang et al. (2019), where CLVSA substantially outperforms the baseline models. The discrepancy may be due to differences in dataset construction, training procedure, or interpretation of architectural details. Access to the original datasets and implementation would make it easier to isolate these factors and more directly reproduce the reported results.

Although we did not reproduce the large reported gains, several small extensions improved performance in our experiments. In particular, using log-relative scaling for stationary OHLCV features greatly improved performance by reducing regime shifts. Replacing unscaled dot-product attention with scaled dot-product attention further led to small improvements in empirical performance.

Overall, our findings highlight the difficulty of reproducing complex architectures from descriptions alone, and emphasize the importance of releasing precise implementation details, datasets, and code alongside proposed model designs.

Acknowledgments

We acknowledge the use of ChatGPT and Claude to search the literature and aid with the implementation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*. arXiv: 1409.0473 [cs.CL].

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio (2015). *Generating Sentences from a Continuous Space*. arXiv: 1511.06349 [cs.LG].

Jianpeng Cheng, Li Dong, and Mirella Lapata (2016). *Long Short-Term Memory-Networks for Machine Reading*. arXiv: 1601.06733 [cs.CL].

Sepp Hochreiter and Jürgen Schmidhuber (1997). "Long Short-Term Memory." *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Diederik P. Kingma and Max Welling (2013). *Auto-Encoding Variational Bayes*. arXiv: 1312.6114 [stat.ML].

Abdul Quadir Md, Sanjit Kapoor, Chris Junni A.V., Arun Kumar Sivaraman, Kong Fah Tee, Sabireen H., and Janakiraman N. (2023). "Novel optimization approach for stock price forecasting using multi-layered sequential LSTM." *Applied Soft Computing* 134, p. 109830. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2022.109830. URL: https://www.sciencedirect.com/science/article/pii/S1568494622008791.

Kriti Pawar, Raj Srujan Jalem, and Vivek Tiwari (2019). "Stock Market Price Prediction Using LSTM RNN." *Emerging Trends in Expert Applications and Security*. Ed. by Vijay Singh Rathore, Marcel Worring, Durgesh Kumar Mishra, Amit Joshi, and Shikha Maheshwari. Singapore: Springer Singapore, pp. 493–503. ISBN: 978-981-13-2285-3.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra (2014). *Stochastic Backpropagation and Approximate Inference in Deep Generative Models*. arXiv: 1401.4082 [stat.ML].

Harald Semmelrock, Tony Ross-Hellauer, Simone Kopeinik, Dieter Theiler, Armin Haberl, Stefan Thalmann, and Dominik Kowald (2025). "Reproducibility in machine-learning-based research: Overview, barriers, and drivers." *AI Magazine*, p. 7000. DOI: https://doi.org/10.1002/aaai.70002.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo (2015). "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting." *Advances in Neural Information Processing Systems*, pp. 802–810.

Hongyang Su, Xiaolong Wang, Yang Qin, and Qingcai Chen (2024). "Attention based adaptive spatial–temporal hypergraph convolutional networks for stock price trend prediction." *Expert Systems with Applications* 238, p. 121899. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2023.121899. URL: https://www.sciencedirect.com/science/article/pii/S0957417423024016.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks." *Advances in Neural Information Processing Systems*, pp. 3104–3112.

Jia Wang, Tong Sun, Benyuan Liu, Yu Cao, and Hongwei Zhu (2019). "CLVSA: A Convolutional LSTM Based Variational Sequence-to-Sequence Model with Attention for Predicting Trends of Financial Markets." *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, pp. 3705–3711. DOI: 10.24963/ijcai.2019/514.