Memory-Augmented Variational RNN

Uladzimir Charniauski
Department of Computer Science and Engineering
University of Connecticut
Storrs, CT 06269
uladizmir.charniauski@uconn.edu

Abstract

Financial time series, particularly stock volatility, present a dual modeling challenge: they exhibit long-range temporal dependence where past events influence future behavior over extended horizons, and they contain significant latent stochastic variability that cannot be captured by deterministic models alone. Memory-augmented recurrent neural networks (MRNNs) have significantly enhanced the capacity of recurrent architectures to model long-range dependence through a novel long memory filter component with polynomially decaying weights. Separately, variational RNN (VRNN) enabled recurrent neural networks to capture latent variability across data through integrated variational autoencoders (VAE). However, no existing framework jointly addresses both properties within a unified architecture. We introduce Memory-augmented variational RNN (MVRNN), a novel class of memory-augmented recurrent neural networks that can model long-range dependence and simultaneously capture latent variability. We demonstrate the effectiveness of our approach on several stock volatility datasets, showing that MVRNNs substantially outperform standard MRNNs in capturing the complex dynamics of financial time series.

1 Introduction

Sequential data modeling is an important topic in machine learning. Many empirical observations from the real world for various phenomena, such as stock market volatility, temperature variations, earthquake magnitude sequences, and traffic data [1, 2, 3, 4] often depend on the long histories of past data. It has been widely noted that standard machine learning algorithms cannot fully represent the long memory effect across inputs, creating additional learning difficulties for such methods [5, 6]. Moreover, separate studies have shown that this kind of data possesses high varibility, which is proven to further compromise the learning abilities of existing methods [7, 8].

Autoregressive Fractionally Integrated Moving-Average (ARFIMA) [9, 10, 11] is widely used to model the long memory in the real-world phenomena. The ARFIMA process has been well studied in the classical time series literature. More recent works [12] empirically demonstrated the effectiveness of leveraging ARFIMA model into deep neural networks to capture long-range dependence in sequential data. However, their memory-augmented solutions were developed using standard RNN frameworks that lack mechanisms to model the inherent stochastic variability present in many real-world time series. For instance, stock return data are often heavily influenced by latent factors such as macroeconomic conditions, investor sentiment, monetary policy, and geopolitical events [13, 14]. These drivers are unobserved directly in the price series, but shape variability in returns, highlighting the importance of incorporating mechanisms for modeling latent factors across subsequent timesteps.

An earlier study [15] introduced variational RNN (VRNN) that can model the variability of structured temporal data, arguing that "the internal structure of RNN is entirely deterministic" and, therefore, insufficient for modeling latent dynamics. Their approach integrates a variational autoencoder (VAE) framework within the recurrent architecture, using Kullback-Leibler (KL) divergence to regularize the learned latent representations toward a prior distribution. This variational formulation enables the model to capture complex, potentially multimodal conditional distributions in the latent space, making it particularly effective when the underlying data generating process involves multiple distinct behavioral regimes or stochastic modes. Although this is an effective method for modeling variational complexities of input data, this recurrent network does not incorporate the long memory components like MRNN. This limitation

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

becomes particularly problematic for financial time series, where long memory effects coexist with regime-switching behavior and latent factor dynamics. The complementary limitations of MRNN and VRNN methods motivate the central contribution of this study: a unified framework that integrates both capabilities.

1.1 Original Contribution

In this research, we introduce the **Memory-augmented variational RNN (MVRNN)** that is capable of simultaneously handling the variability and long-range dependence across data. We will present the cross-modified cell structure design of existing MRNN [12] and VRNN [15]. To comprehensively explore probabilistic modeling approaches, we develop two variants memory-augmented variational recurrent networks containing different probabilistic models in their structure. The MVRNN which incorporates the variational autoencoder (VAE), and MVRNN-WAE incorporating the Wasserstein autoencoder (WAE) framework [16] with optimal transport distances. We perform numerical studies on several datasets to illustrate the advantages of our proposed models under both frameworks. We demonstrate that the proposed neural networks benefit from extracting the long memory effect from data and modeling variability and outperform existing deterministic solutions which we will utilize as performance baselines.

2 Long Memory Variational Recurrent Networks

The MVRNN introduces a selected autoencoder with a probabilistic component, VAE or WAE, to the MRNN cell architecture to model sequential data with complex temporal dependence. Our network adopts a *latent-variable probabilistic approach* [17, 15] that models the joint distribution over sequences as follows:

$$p_\theta(x_{1:T}, z_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t|z_t, h_{t-1})\, p_\theta(z_t|h_{t-1}).$$

This factorization captures both *temporal structure* through the dependence on the hidden state $h_{t-1} \in \mathbb{R}^q$ and *uncertainty* through latent variables $z_t \in \mathbb{R}^{p_z}$. The approximate posterior to be defined in future sections is used to make inference tractable, turning the intractable log-likelihood into a variational objective. The result is a probabilistic generative model that learns both the dynamics and variability of sequential data.

2.1 Variational Autoencoder

MVRNN will implement variational autoencoder (VAE) in its basis structure that will allow for modeling latent variability in volatility data. This probabilistic approach to encoding will allow our method to generate diverse data samples by modeling latent space distribution.

VAE has a simple generative graphical model: For each time step t, draw a latent variable

$$z_t \sim \mathcal{N}(\mu_{0,t}, \text{diag}(\sigma_{0,t}^2)), \text{with } [\mu_{0,t}, \sigma_{0,t}] = \varphi^{prior}(h_{t-1}), \tag{1}$$

where $\mu_{0,t}$ and $\sigma_{0,t}$ denote the parameters of the conditional prior distribution. Then, draw the observed variable:

$$x_t \mid z_t \sim \mathcal{N}(\mu_{x,t}, \text{diag}(\sigma_{x,t}^2)), \text{with } [\mu_{x,t}, \sigma_{x,t}] = \varphi^{dec}(\varphi^z(z_t), h_{t-1}), \tag{2}$$

where $\mu_{x,t}$ and $\sigma_{x,t}$ denote the parameters of the generating distribution.

The recognition (inference) model, which is our encoder, is an approximate posterior following the equation:

$$z_t \mid x_t \sim \mathcal{N}(\mu_{z,t}, \text{diag}(\sigma_{z,t}^2)), \text{with } [\mu_{z,t}, \sigma_{z,t}] = \varphi^{enc}(\varphi^x(x_t), h_{t-1}), \tag{3}$$

where $\mu_{z,t}$ and $\sigma_{z,t}$ denote the mean and standard deviation of the approximate posterior. It is used only during inference/training and is often drawn as a separate arrow $x \to z$.

The utilized φ^{enc}, φ^{dec}, φ^{prior}, φ^x, φ^z extract features from their respective components and can be any highly flexible function such as neural networks. These feature extractors are found to be crucial for learning complex sequences [15].

Training maximizes the evidence lower bound (ELBO):

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \text{KL}[q_\phi(z|x)\|p(z)], \tag{4}$$

where $\text{KL}[q_\phi(z|x)\|p(z)]$ is the Gaussian KL divergence defined as follows:

$$\text{KL}[q_\phi(x|z)\|p(z)] = \frac{1}{2}\sum_{i=1}^{p_z}\left[\log\frac{\sigma_{prior,i}}{\sigma_{enc,i}} + \frac{\sigma_{enc,i} + (\mu_{enc,i} + \mu_{prior,i})^2}{\sigma_{prior,i}} - 1\right]. \tag{5}$$

Because we work with real-valued sequential data, we pick parametric Gaussian forms $p_\theta(x \mid z)$ and use the reparameterization trick $z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$ ($\epsilon \sim \mathcal{N}(0, I)$) to get low-variance gradients for joint SGD on θ and ϕ. Conceptually, the generative model is $p(z)p_\theta(x|z)$, and the VAE learns a decoder p_θ and an encoder q_ϕ, so the sampling is done by $z \sim p(z),\ x \sim p_\theta(x|z)$.

In practice, maximizing for ELBO is equivalent to minimizing for the selected objective function we define as follows:

$$\mathcal{L}_{VAE} = \mathcal{L}_{MSE} + \beta \cdot \mathcal{L}_{KL}, \tag{6}$$

where $\mathcal{L}_{MSE}$ and $\mathcal{L}_{KL}$ are MSE measured between predicted and actual labels and KL distance losses defined as in Equation 5, and $\beta = \min\{1, n_{epoch}/K\}$ is the annealing parameter with K typically being a large integer factor.

2.2 Wasserstein Autoencoder

Wasserstein autoencoder (WAE) is an alternative probabilistic model for neural networks [16]. We propose a separate variational recurrent MRNN with WAE, which we denote as MVRNN-WAE. This autoencoder follows similar to VAE generative and graphical models described across Equations 1-3. However, instead of KL divergence, WAE minimizes the Wasserstein distance between the aggregated posterior $\mathbb{Q}_Z$ and prior $\mathbb{P}_Z$:

$$W(\mathbb{P}_Z, \mathbb{Q}_Z) = \inf_{\gamma \in (\mathbb{P}_Z, \mathbb{Q}_Z)} \mathbb{E}_{(z,z') \sim \gamma}[c(z, z')]. \tag{7}$$

We approximate $W(\mathbb{P}_Z, \mathbb{Q}_Z)$ using Maximum Mean Discrepancy (MMD) with RBF kernel:

$$\mathrm{MMD}^2(P, Q) = \mathbb{E}_{z,z' \sim P}[k(z, z')] + \mathbb{E}_{z,z' \sim Q}[k(z, z')] - 2\mathbb{E}_{z \sim P, z' \sim Q}[k(z, z')], \tag{8}$$

where

$$k(z, z') = \exp\left(-\frac{\|z - z'\|^2}{2\sigma^2}\right) \tag{9}$$

is the RBF kernel with bandwidth σ.

To empirically estimate MMD, we use samples $\{z_t^{\mathrm{enc}}\}_{t=1}^T$ from encoder and $\{z_t^{\mathrm{prior}}\}_{t=1}^T$:

$$\widehat{\mathrm{MMD}}^2 = \frac{1}{T^2} \sum_{t,t'=1}^{T} k(z_t^{\mathrm{enc}}, z_{t'}^{\mathrm{enc}} + \frac{1}{T^2} \sum_{t,t'=1}^{T} k(z_t^{\mathrm{prior}}, z_{t'}^{\mathrm{prior}}) - \frac{2}{T^2} \sum_{t,t'=1}^{T} k(z_t^{\mathrm{enc}}, z_{t'}^{\mathrm{prior}}) \tag{10}$$

This estimate allows us to define the objective function in a fashion similar to VAE case:

$$\mathcal{L}_{\mathrm{WAE}} = \mathcal{L}_{\mathrm{MSE}} + \lambda_{\mathrm{MMD}} \cdot \widehat{\mathrm{MMD}}^2, \tag{11}$$

where $\lambda_{\mathrm{MMD}} := \min\{0.01, n_{epoch}/K\}$ is the MMD annealing parameter with large K. This objective function enforces global distributional alignment rather than per-data-point regularization.

2.3 Long Memory Filter

Long memory filter

MVRNN incorporates the MRNN hidden memory unit formulated as follows:

$$m_t = \tanh(W_{mm} m_{t-1} + W_{mf} F(x_t; d) + b_m). \tag{12}$$

The unit m_t works in parallel with the traditional hidden unit h_t and takes responsibility for modeling the long memory pattern in time series.

For the memory parameter $d = (d_1, \ldots, d_{p_x})'$, we restrict each $0 < d_i < 0.5$ such that the fractional integration can provide long memory. Moreover, to make the design more general, we also let the memory parameter d depend on other variables, and hence the notation d_t.

The memory parameter update occurs in the following fashion:

$$d_t = \frac{1}{2}\sigma(W_d[d_{t-1}, h_{t-1}, m_{t-1}, x_t] + b_d). \tag{13}$$

We denote MVRNNF and MVRNNF-WAE as the network with memory parameters constant over time points t, and it can be implemented by fixing $W_d = 0$ in the update equation 13.

2.4 Recurrence

The recurrence procedure of MVRNN and MVRNN-WAE is given below:

$$\begin{cases} r_t = g(W_{zh}h_t + W_{zm}m_t + b_z) \\ h_t = \tanh(W_{hh}h_{t-1} + W_{hx}[\varphi^x(x_t), \varphi^z(z_t)] + b_h) \\ F(x_t; d)_i = \sum_{j=1}^{K} w_j(d_{t,i}) x_{i,(t-j+1)} \\ d_t = \frac{1}{2}\sigma(W_d[d_{t-1}, h_{t-1}, m_{t-1}, x_t] + b_d) \\ m_t = \tanh(W_{mm}m_{t-1} + W_{mf}F(x_t; d) + b_m) \end{cases}$$

This procedure is similar to the one of MRNN. The core difference lays in the hidden state update, which depends on the concatenated feature extractors $\varphi^x(x_t)$ and $\varphi^z(z_t)$.

The underlying network can then be designed as

$$r_t = g(W_{zh}h_t + W_{zm}m_t + b_z),$$

which encapsulates the memory unit m_t and a hidden unit h_t influenced by latent factors. Such a design allows our network to simultaneously model long-range dependence and latent variabilities arising across data. Appendix A.2 features a graphical representation of the MVRNN and MVRNN-WAE cell structure.

3 Experimental Design

This section reports several numerical experiments. We compare the models using time series forecasting tasks on four financial time series datasets. We also present the Shapiro-Wilk results to demonstrate the convergence of MVRNN(F)-WAE optimal mean distances toward multivariate normal distributions in Appendix B. The code to reproduce the experiments can be found at https://github.com/uladcharn/MVRNN.

All the networks are implemented in `PyTorch`. The selected hidden and latent dimensions consist of 64 and 16 parameters. For modeling long memory, we select $d = 0.4$ and $K = 100$ globally, where the latter parameter is also used in β- and λ_{MMD}-annealing for convenience. We use the AdamW algorithm with learning rate 0.01 for optimization. The optimization is stopped when the loss function drops by less than 10^{-4} or has been increasing for 100 steps or has reached 500 steps in total. The learned model is chosen to be the one with the smallest loss on the validation set. At each training step, we collect latent samples z_t^{enc} and z_t^{prior} through reparametrization, whereas we set those as mean encoder and mean prior signals during validation and test phases.

Due to the non-convex nature of the optimization problem, we initialize with 10 different random seeds and arrive at 10 different trained models for each network. The generated results will be presented in two separate tables featuring achieved RMSE and MAE metrics of considered methods. The first table will feature the distribution of 10 results for RMSE, referred to as *overall performance* in terms of RMSE, and the second for MAE, referred to as *overall performance* in terms of MAE.

3.1 Datasets

We compare our models with the baselines on four real-world stock volatility datasets: two long, one intermediate, and one short memory. We split the datasets into training, validation and test sets, and report their lengths below using notation $n_{train} + n_{val} + n_{test}$. MVRNN(F) and MVRNN(F)-WAE have target losses defined in Equations 6 and 11. We perform one-step rolling forecasts on the test set and calculate the RMSE and MAE metrics. Table 3.1 summarizes key characteristics of each dataset employed in our experiments. More properties of our data are illustrated in Appendix A.1.

Dataset	**Size**	$n_{train} + n_{val} + n_{test}$	**Memory Behavior**	**Volatility Type**
DJI	5030	2500 + 1000 + 1530	long	log
NSDQ	2528	1500 + 500 + 528	short	log
AAPL	2516	1500 + 500 + 516	intermediate	abs-log
GOOGL	5118	2500 + 1000 + 1618	long	abs-log

Table 1: Description of datasets

3.1.1 Results

Tables 2 and 3 report overall performance of one-step forecasting in terms of an RMSE and MAE metrics, respectively. We can see that MVRNNF and MVRNNF-WAE report the lowest RMSE scores compared to other models on datasets exhibiting long and intermediate memory. On the other hand, RNN has only demonstrated the best performance on NSDQ dataset with explicitly short memory behavior. Although RNN outperformed all other models on this dataset, we suspect that this is due to the insufficiency in number of epochs used for training. This notion could occur due to similar reasons as to that MVRNN resulted in significantly larger RMSE scores on NSDQ and AAPL datasets, where MVRNN's complex architecture would require a more extensive training over larger number of epochs and random seed initiations. Nonetheless, MVRNN(F) and MVRNN(F)-WAE generally remain competitive w.r.t. original methods across the data.

	RNN	VRNN	MRNNF	MRNN	MVRNNF	MVRNN	MVRNNF-WAE	MVRNN-WAE
DJI	0.00328	0.00258	0.00248	0.00263	**0.00243**	0.00244	0.00243	0.00256
NSDQ	**0.01045**	0.02582	0.01064	0.01049	0.01047	0.01996	0.01070	0.02684
AAPL	0.01372	0.01573	0.01328	0.0126	0.01319	0.02138	**0.01259**	0.01505
GOOGL	0.01319	0.01384	0.01356	0.01804	0.01372	0.01336	0.01394	**0.01315**

Table 2: Overall performance in terms of RMSE. The best result is highlighted in **bold**.

	RNN	VRNN	MRNNF	MRNN	MVRNNF	MVRNN	MVRNNF-WAE	MVRNN-WAE
DJI	0.00254	0.00180	0.00174	0.00189	**0.00170**	0.00170	0.00170	0.00181
NSDQ	**0.00754**	0.02370	0.00774	0.00759	0.00762	0.01732	0.00783	0.00783
AAPL	0.01044	0.01253	0.01010	0.00954	0.00994	0.01831	**0.00941**	0.01201
GOOGL	0.00940	0.00967	0.00967	0.01456	0.00931	0.00998	0.01016	**0.00928**

Table 3: Overall performance in terms of MAE. The best result is highlighted in **bold**.

4 Discussion and Conclusion

In this study, we introduced memory-augmented variational recurrent neural networks (MVRNNs) that unify long-range temporal modeling with latent stochastic representation learning. We presented two distinct variants leveraging Variational Autoencoders (MVRNN-VAE) and Wasserstein Autoencoders (MVRNN-WAE), demonstrating their effectiveness on financial time series forecasting tasks where both long-memory dependencies and latent variability are fundamental characteristics. In terms of future work, it is interesting to study whether implementing non-Gaussian priors would help our neural network model more complex latent variabilities along with long memory sequences. Likewise, studying the implementation of other autoencoders beyond VAE and WAE in the MVRNN cell architecture would serve as a potential research venue for expanding the capacity of this network for modeling latent variabilities in data. Because the original work on MRNN concerned the computational cost for models with dynamic d, which was only exarcebated after introducing a variational component, another promising direction for future work would concern model simplification and exploiting faster optimization strategies.

References

[1] S. H. A. Bakar and C. M. Hafner. Forecasting realised volatility using arfima and har models. *Quantitative Finance*, 19(12):1925–1934, 2019.

[2] Naiming Yuan, Zuntao Fu, and Shida Liu. Extracting climate memory using fractional integrated statistical model: A new perspective on climate prediction. *Scientific Reports*, 4:6577, 2014.

[3] Ferry Kondo Lembang, Lexy Janzen Sinay, and Asrul Irfanullah. Arfima modelling for tectonic earthquakes in the maluku region. *Indonesian Journal of Statistics and Its Applications*, 5(1):39–49, 2021.

[4] Mallikarjuna Doodipala. Time series analysis for long memory process of air traffic using arfima. *International Journal of Scientific & Technology Research*, 9(3):6268–6272, 2020.

[5] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[6] Alexander Greaves-Tunnell and Zaid Harchaoui. A statistical investigation of long memory in language and music. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 2394–2403, June 2019.

[7] Seyedeh Azadeh Fallah Mortezanejad, Ruochen Wang, et al. Addressing challenges in time series forecasting: A comprehensive comparison of machine learning techniques. *arXiv preprint*, 2025. arXiv:2503.20148.

[8] Cai Chen and Jin Dong. Deep learning approaches for time series prediction in climate resilience applications. *Frontiers in Environmental Science*, 13, 2025.

[9] J. R. M. Hosking. Fractional differencing. *Biometrika*, 68(1):165–176, 1981.

[10] A. I. McLeod and K. Hipel. Fractional time series modelling. *Technometrics*, 28(2):101–111, 1986.

[11] C. W. J. Granger and R. Joyeux. An introduction to long-memory time series models and fractional differencing. *Journal of Time Series Analysis*, 1(1):15–29, 1980.

[12] Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. Do RNN and LSTM have long memory? In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11365–11375. PMLR, 13–18 Jul 2020.

[13] Malcolm P. Baker and Jeffrey Wurgler. Investor sentiment and the cross-section of stock returns. *The Journal of Finance*, 61(4):1645–1680, 2006.

[14] Nai-Fu Chen, Richard Roll, and Stephen A. Ross. Economic forces and the stock market. *The Journal of Business*, 59(3):383–403, 1986.

[15] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015. arXiv:1506.02216 [cs.LG].

[16] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2018. Version: arXiv:1711.01558v4, December 2019.

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

A Data and Model Vizsualizations

A.1 Memory Properties and Latent Dynamics in Data

In Figure 1, we visualize the long memory in the datasets via autocorrelation plots over 400 lags with the first lag $r_0 = 1$ being cropped off. DJI and GOOGL lags exhibit slow, gradual decay, which indicates long memory. AAPL exhibits a more spurious, moderate rate of decay, consistent with intermediate memory. On the other hand, NSDQ lags immediately drop to near-zero values, indicating short memory.

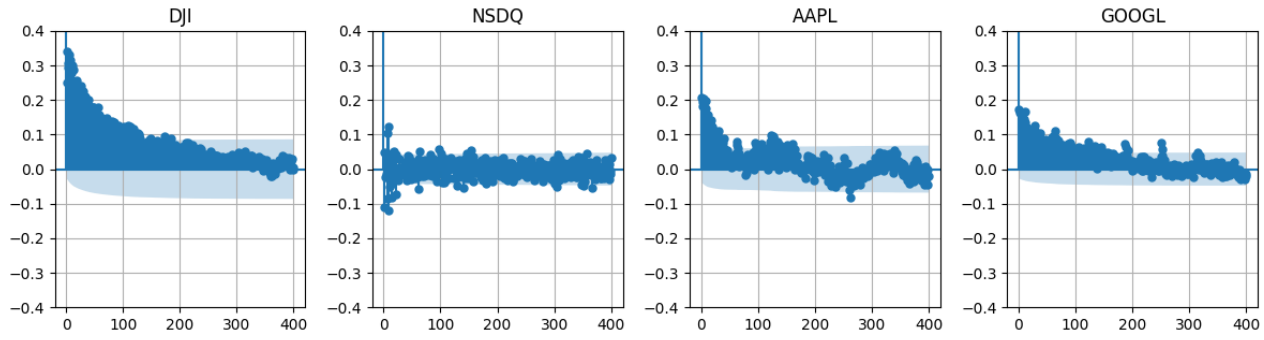

Figure 1: Autocorrelation plots for DJI, NSDQ, AAPL, and GOOGL datasets.

We also provide the rolling statistics visualizations for our datasets to demonstrate the presence of latent variabilities across observations. The gray lines represent the original daily returns or price changes, while the blue and red lines show the rolling mean and rolling standard deviation, respectively, calculated using a moving window approach.

For DJI, the rolling mean fluctuates around zero with the rolling standard deviation exhibiting notable spikes, particularly around timestep 2000, indicating periods of increased market volatility characteristic of market indices. The NSDQ series displays a rolling mean that remains tightly centered near zero with minimal directional trends. The rolling standard deviation shows moderate variability with brief volatility episodes, consistent with the short-memory behavior previously identified for this dataset. Unlike previous two, the AAPL stock exhibits more pronounced fluctuations in both statistics. The rolling mean displays distinct peaks around timesteps 1000 and 1500, suggesting sustained price momentum periods. The rolling standard deviation exhibits considerable variation, reflecting sensitivity to company-specific events and market sentiment. Finally, GOOGL demonstrates relatively moderate behavior with the rolling mean oscillating around zero at smaller amplitudes. The rolling standard deviation remains contained with occasional upticks, suggesting less extreme volatility clustering, which may explain the consistently high normality percentages observed in the latent space analysis.

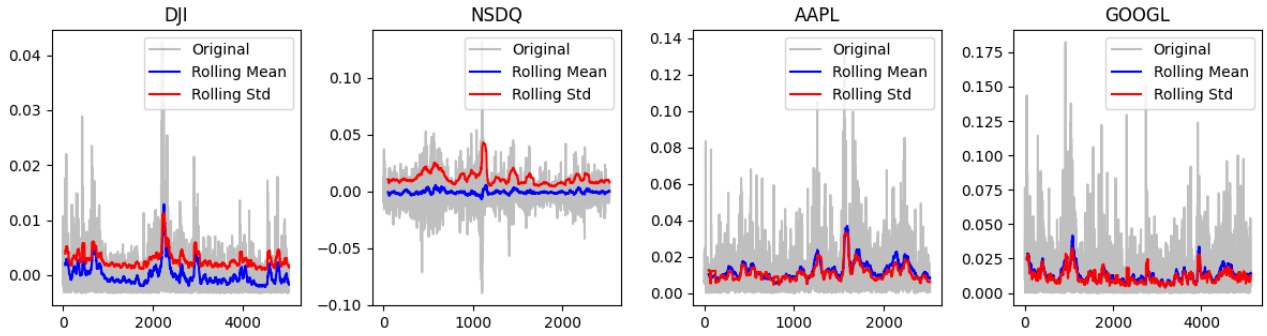

Figure 2: Rolling Statistics plots for DJI, NSDQ, AAPL, and GOOGL datasets.

A.2 MVRNN and MVRNN-WAE Cell Structures

Figures 3 and 4 provide the cell structure design for MVRNN and MVRNN-WAE models. The designs represent combined cells of MRNN and VRNN. The difference between MVRNN and MVRNN-WAE cell is in the presence of z_t^{prior} in the output used for computing MMD loss.

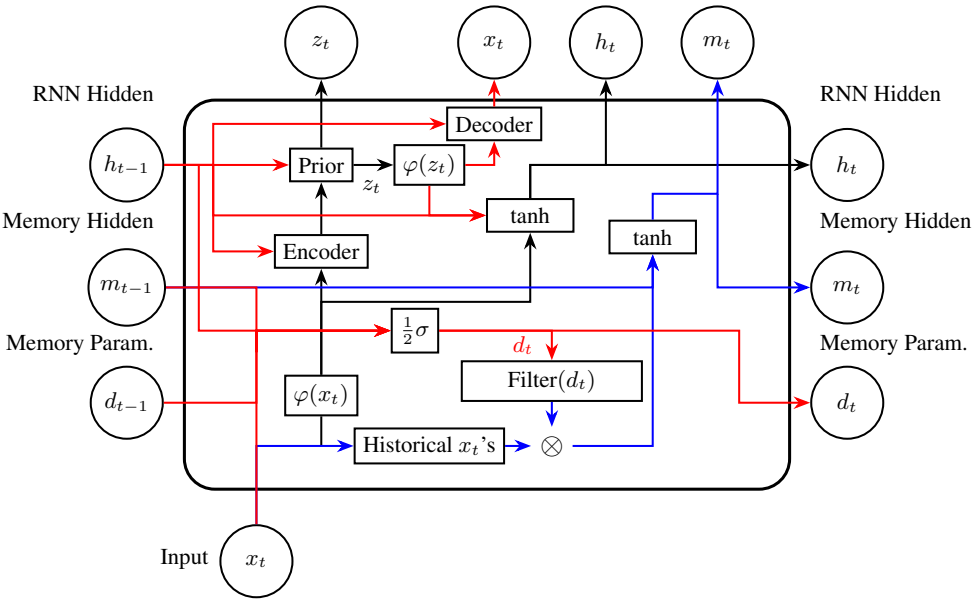

Figure 3: MVRNN cell structure

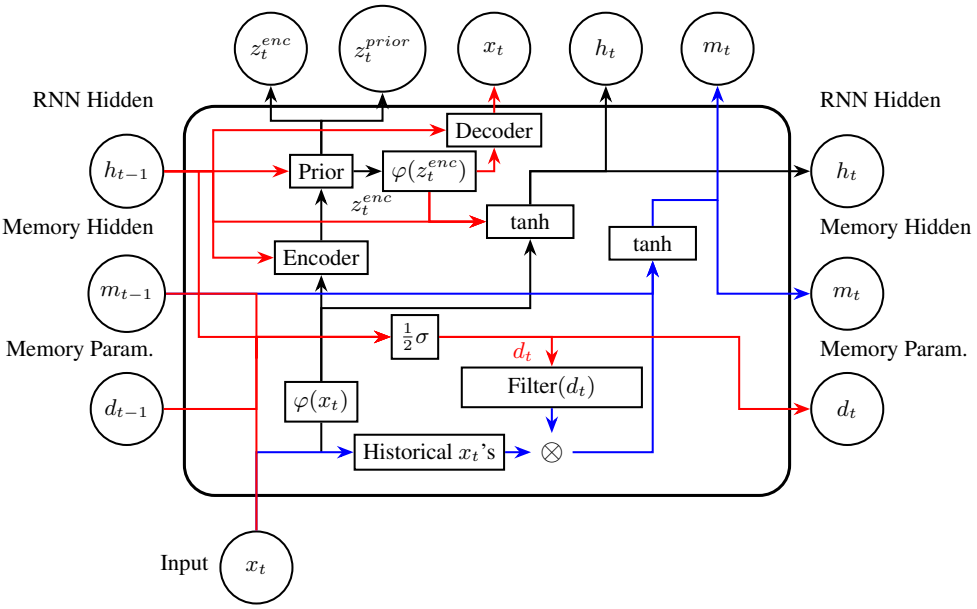

Figure 4: MVRNN-WAE cell structure

B Shapiro-Wilk Test for Normality of Optimal Mean Distances

This section provides a comprehensive analysis of the normality properties of latent dimensions in both MVRNN-WAE and MVRNNF-WAE models. A fundamental objective of Wasserstein Autoencoders is to match the aggregated

posterior distribution to a specified prior distribution—in this case, a standard Gaussian. The degree to which this matching succeeds directly impacts the model's ability to generate realistic samples and learn meaningful latent representations. To rigorously assess this objective, we employ the Shapiro-Wilk test for normality on latent samples extracted from the validation set.

The Shapiro-Wilk test computes a W-statistic that measures the deviation of empirical samples from a normal distribution, with the null hypothesis being that the data follows a Gaussian distribution. We reject normality at significance level $\alpha = 0.05$ when the p-value falls below this threshold. This statistical framework allows us to quantify not just whether the latent distributions are approximately normal, but to what extent the normality assumption holds across different datasets and model configurations.

Our methodology operates as follows: For each model, we collect 5000 latent samples with our models by performing a forward pass through the validation sequences with deterministic sampling. We then generate 5000 random projections sampled from the standard normal distribution. The projections are tested for normality using Shapiro-Wilk test diagnostics. We visualize the test outcomes through the distribution historgram of p-values on every dataset described in Section 3.1.

B.1 Results

Figures 5-8 demonstrate the p-value distributions of encoder and prior projections with MVRNN(F)-WAE over 5000 samples. The percentage score included in a title for each plot indicates the percentage of normal encoder and prior projections. We first observe that samples collected with MVRNNF-WAE on AAPL dataset demonstrated severely skewed distributions for both components. This is likely due to the fact that the memory parameter value $d = 0.4$ used for modeling long-term temporal dynamics was too high for this dataset with intermediate memory property. Likewise, the distributions with MVRNN-WAE samples collected on DJI dataset demonstrate high skewedness toward smaller values, especially compared to those collected with MVRNNF-WAE on the same dataset. The cause is likely that this model was unable to learn strong temporal dependence over the learning horizon. We note that this is not the issue for MVRNNF-WAE, where the memory parameter is fixed at high values.

Surprisingly, the encoder and prior sample projections for both MVRNN-WAE and MVRNNF-WAE demonstrate a visible convergence of their respective p-value distributions on a NSDQ dataset exhibiting short memory behavior. Both models achieve relatively uniform distributions with high percentages of normal projections, suggesting that short memory dynamics are effectively captured by both the fixed and flexible memory parameter approaches. Finally, the GOOGL dataset maintains remarkably consistent performance across all scenarios. This stability indicates that the model architectures are well-suited for capturing the temporal characteristics present in this particular stock's price movements, regardless of whether memory parameters are fixed or learned adaptively.

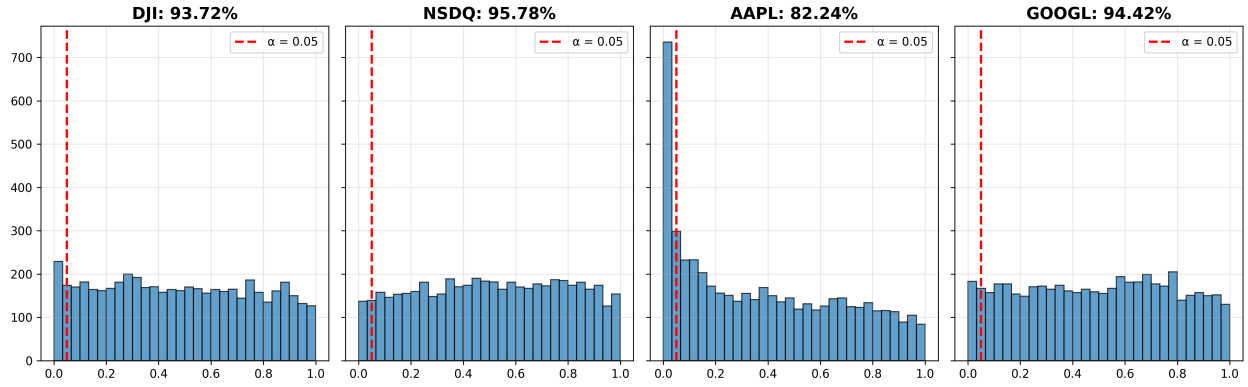

Figure 5: Encoder p-values distributions for MVRNNF-WAE on the tested datasets.

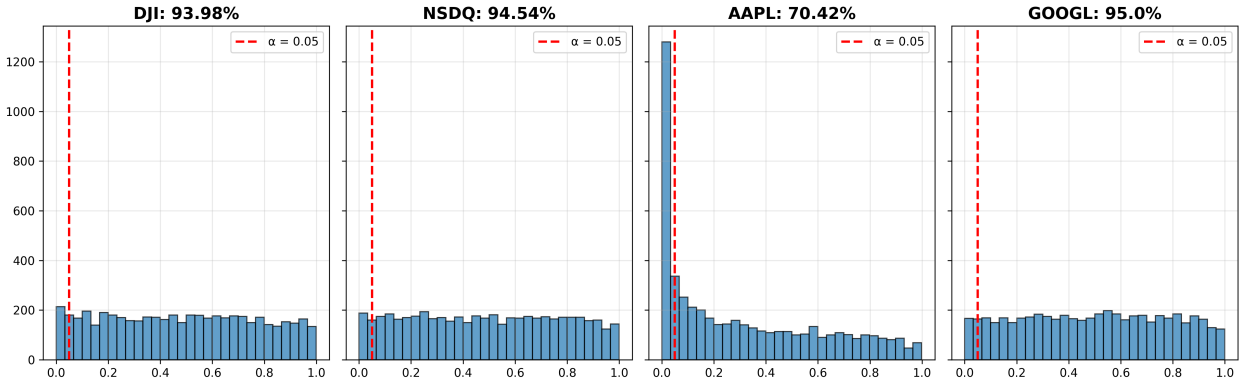

Figure 6: Prior p-values distributions for MVRNNF-WAE on the tested datasets.

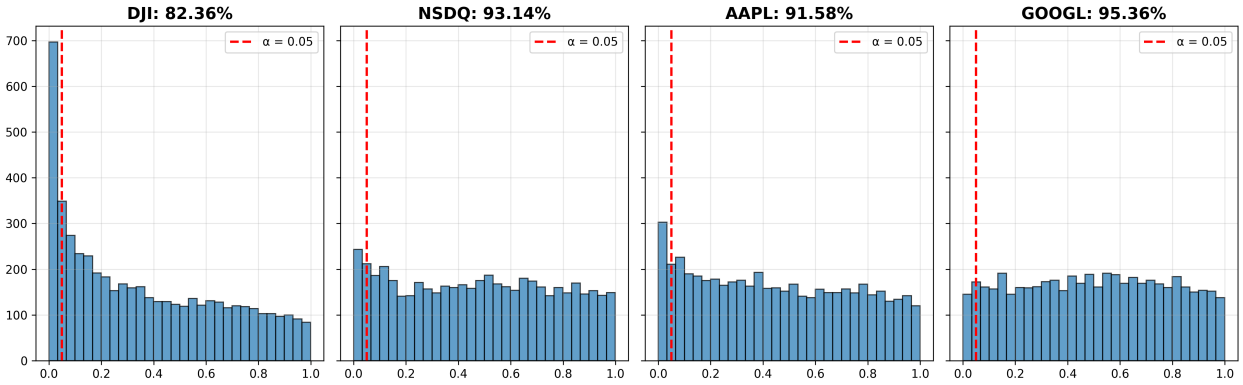

Figure 7: Encoder p-values distributions for MVRNN-WAE on the tested datasets.

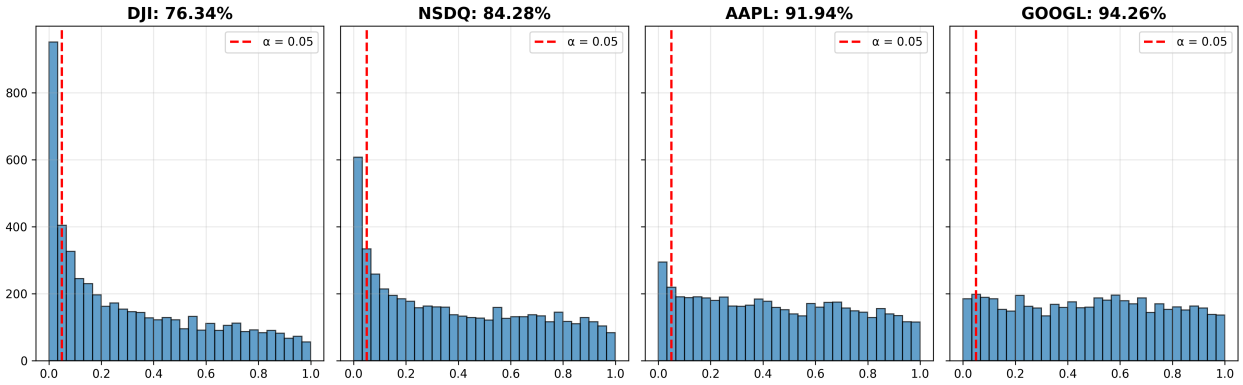

Figure 8: Prior p-values distributions for MVRNN-WAE on the tested datasets.