# Modelling Generalized Stochastic Process with Deep Generative Neural Network*

Rishav Bhagat, Erik Scarlatescu

**Abstract.** Modeling stochastic processes often relies on parametric methods (Wiener processes, Gaussian processes, ARCH/GARCH), where distributions are assumed known. Mixture models extend flexibility, but remain limited. Deep learning (RNNs, LSTMs, Transformers) has excelled in time series prediction, yet typically produces deterministic outputs. To capture uncertainty, deep generative models (GANs, VAEs, diffusion models) prove powerful. Non-parametric approaches (KDE, Time Series GANs) are essential when the distribution shape is unknown. Our proposed approach leverages deep learning to quantify uncertainty in stochastic process modeling, making no assumptions about the underlying distribution's form. We do this by modularizing the method into three parts: the network architecture, network stochasticity, and training scheme. We ran ablations and combined methods to generate better models for stochastic processes. We tested this approach on data from financial, physical, and biological domains, as well as on synthetic data.

**Key words.** stochastic processes, generative modeling

**MSC codes.** 68Q25, 68R10, 68U05

## 1. Related Work.

Modeling stochastic processes has a lot of prior work, with parametric approaches like Wiener processes, Gaussian processes, and ARCH/GARCH models forming the foundation for time series analysis with their reliance on maximum likelihood estimation. These approaches provide efficient frameworks when the underlying distribution shape is known. Mixture of processes extends this concept by combining simpler experts with adaptable weighting, offering a more nuanced modeling capability [5]. However, while such parametric methods are able to represent a large process space than any of the individual experts, they are inherently limited in their ability to capture entirely arbitrary distributions.

In recent years, deep learning has revolutionized time series prediction. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks were some of the first deep learning models to tackle sequential data, demonstrating their ability to capture long-range dependencies within time series. Transformer-based models, with their powerful attention mechanisms, have also gained significant traction, offering efficient learning of these dependencies and potentially surpassing RNN/LSTM performance in specific tasks. However, these models (used in their regular form), do not generate stochastic samples; they deterministically act on an input and generate a single output. Structured State Spaces for Sequence Modeling (S4) models are new models that use convolutions instead of autoregressively predictions to speed up inference and training. They also instill an inductive bias via the HIPPO matrix that is good for sequence data. [4] [7]

Beyond deterministic prediction, deep generative models provide a compelling approach for modeling uncertainty in stochastic processes. Generative Adversarial Networks (GANs) utilize an adversarial training scheme where a generator learns to produce samples resembling the target distribution, while a discriminator attempts to distinguish real data from generated

*Submitted to the editors DATE.

**Funding:** This work was funded by the Fog Research Institute under contract no. FRI-454.

samples. Variational Autoencoders (VAEs) employ an encoder-decoder architecture with an interesting twist: noise is introduced into the latent representation during encoding, allowing the decoder to reconstruct the original data and, more importantly, generate novel samples that reflect the inherent uncertainty of the data distribution. Diffusion models take a unique approach, learning the process of progressively adding noise to a data point until it resembles pure noise. By reversing this learned process, they can effectively generate high-quality samples from the target distribution. [1]

Non-parametric approaches to modelling stochastic processes become necessary when the distribution shape is unknown. Kernel Density Estimation (KDE) offers another avenue for non-parametric modeling by directly estimating the distribution of future data points based on historical observations. However, KDE can encounter scalability challenges with high-dimensional data. Time series Generative Adversarial Networks (GANs) have emerged as a powerful tool, leveraging a generator-discriminator architecture in latent space to model complex stochastic processes. Here, a Wiener process injects stochasticity into the generator, allowing it to transform the input into a sample from the target process. There are many approaches applying GANs (with and without transformers) to generate time series samples. [12] [8]

This established body of prior work across various methodologies lays the groundwork for our proposed approach, which utilizes deep learning to quantify uncertainty within the context of modeling stochastic processes without any assumptions of its form.

## 2. Methods.

Our method will consist of three modules, each with interchangeable options. We plan to run an ablation on the different combinations.

1. A flexible network architecture that supports stochastic outputs and is flexible enough to capture the complexity of arbitrary time series data
2. A method to introduce stochasticity into the network outputs to allow for sampling of stochastic processes
3. A training scheme to match the converged model's output distribution of time series roll-outs with the data distribution.

### 2.1. Network Architecture.

Neural networks are known to be universal function approximators, making them a suitable choice for this situation. However, this flexibility is not always an advantage, as it can mean that considerably more data is needed for neural networks to learn the patterns in the data. There exist several structures, such as RNNs, CNNs, and transformers, which build off of neural networks, while adding additional structure that serves to work as an inductive bias.

The kernels used by CNNs leverage the fact that order matters in time series data. Additionally, RNNs and transformers are well suited to sequential data. While RNNs have been largely replaced by transformers in many domains, it is still worth experimenting the former to see how well it performs in this domain.

### 2.2. Network Stochasticity.

There are multiple ways to introduce stochasticity into network outputs. A few that we are considering right now are:

1. Concatenate a random vector $\vec{x}_r \in \mathbb{R}^k$ of length $k$ sampled from a chosen source distribution.

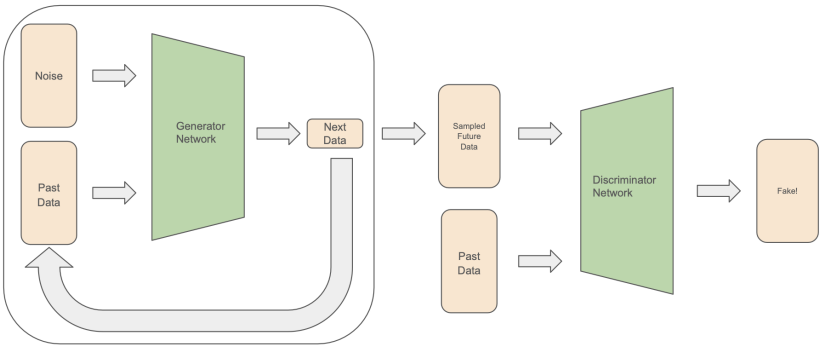


**Figure 1.** *Architecture for conditional GAN*

2. Introduce additive noise sampled from a chosen distribution in an intermediate latent layer.

These also allow for flexibility in the chosen noise distribution. The main things we would like to try is Gaussian noise, uniform noise, and Wiener process noise (noise conditioned on the noise from the last time step).

**2.3. Training Scheme.** The training schemes that could be used to learn the target distribution are variants of the GAN training scheme, variants of the VAE embedding learning, and variants of the diffusion model training scheme. We plan to modify these methods to make them fit our problem of stochastic time series sampling.

To adapt GANs to time series prediction we plan to look at methods from prior work (see section 1) for applying GANs to time series data. A simplistic method to adapt these networks to time series data is to just stack the data from the last $T$ steps to sample the next $N$ steps.

**2.3.1. Generative Adversarial Networks (GAN).** As shown in figure 1, the conditional GAN training scheme consists of a generator, which samples a future trajectory conditioned on the trajectory so far, and a discriminator, which determines whether or not a past trajectory and a future trajectory is real or generated. The sampled trajectories are generated autoregressively, with the generator generating outputs one time step at a time and using its previous generated output to generate the next time step from the sequence.

Once the generator has created a trajectory, this whole trajectory is fed into the discriminator at once and it predicts if it is real or fake. This is because with only one predicted time step, the sample is too noisy for the discriminator to make an accurate prediction. With a longer predicted trajectory, the discriminator has more information with which to make its decision.

**2.3.2. Conditional Variational Autoencoders (VAE).** The conditional VAE training scheme is very similar to a standard VAE, except that the past data is fed into both the encoder and the decoder. This allows the latent space to sample over the future trajectories conditioned on the past trajectory. Once the VAE is trained, points can be generated autoregressively by

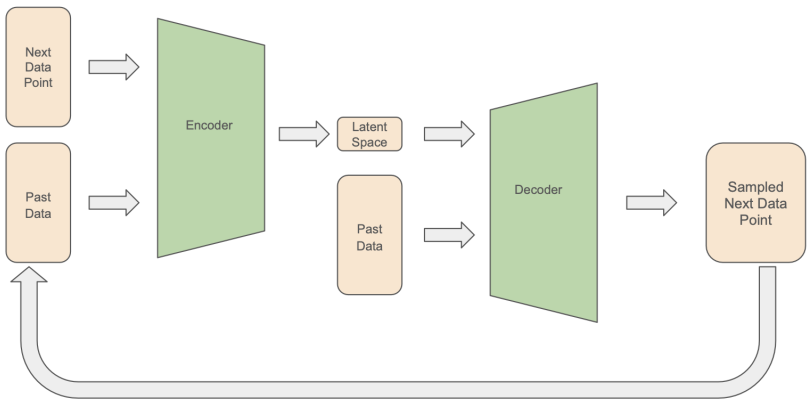


**Figure 2.** *Architecture for conditional VAE*

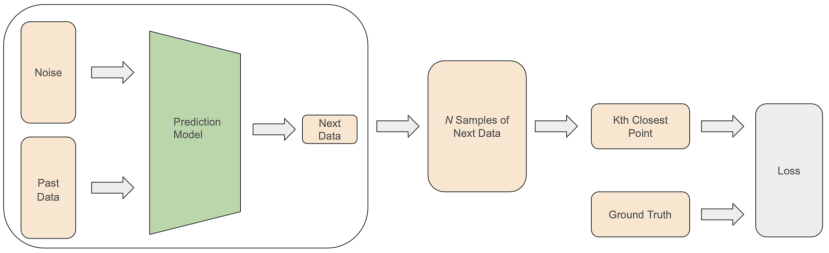


**Figure 3.** *Architecture for SAD EMILIE*

conditioning on the sequence generated so far and sampling from the latent space to predict another point.

**2.3.3. SAmple Density Estimate MaxImum LIkliehood Estimate (SadEmilie).** This is a method that we came up with to directly optimize for the objective of creating a distribution that maximizes the likelihood of our data. Further, since we are using a model that generates samples and then a non parametric density estimate via those samples, our model can represent arbitrary distributions. However, this means that our networks must be regularized so that it doesn't just memorize the training data and reproduce only exact samples from the dataset.

The algorithm proceeds as follows:

For each batch of training data

1. Use the prediction model to generate $n$ samples conditioned on the past data (up to $t$).
2. Find the $k$th closest sample $\hat{X}^k_{t+1}$ to the ground truth $t+1$ data $X_{t+1}$ (for each example in the batch).

3. Set $L := \log \text{MSE}(\hat{X}^k_{t+1}, X_{t+1})$
4. Update the prediction model to minimize $L$

The reason we believe this method to work (besides empirical results) is the following derivation:

Given $n$ predicted samples conditioned on the past data of the next ($t+1$th data point), we use KNN-density estimation to estimate a density function around these samples:

$$f(x) = \frac{k}{n} \frac{1}{V_d R^d_k(x)}$$

where $k$ is a hyperparameter, $V_d$ is the volume of a $d$-dimensional hypersphere, and $R^d_k(x)$ is the distance between $x$ and the $kth$ closest datapoint.

Now, we can use this density function and optimize it to be the $MLE$ of our training dataset samples $X_{t+1}$ by minimizing the negative log likelihood as follows:

$$\begin{aligned} L &= -\log f(X_{t+1}) \\ &= -\log \frac{k}{n} \frac{1}{V_d R^d_k(X_{t+1})} \\ &= -\log \frac{k}{n} + \log V_d + d \log R_k(X_{t+1}) \end{aligned}$$

Notice that everything here is a constant besides the $\log R_k(x)$ term, which we set as our loss to minimize.

Some weaknesses of this method is that it will not work well on predicting long term trends along with potentially memorizing the data is the prediction model is too powerful and doesn't use regularization.

Each of these sections are orthogonal from each other allowing us to run ablations with many combinations and evaluate them with metrics designed to see how well the output distribution models the target distribution.

## 3. Applications.

**3.1. Domains.** The ability to model stochastic processes using neural networks has far-reaching implications in finance, physics, and biology. In finance, these models fuel more accurate stock price predictions, facilitate advanced option pricing techniques, and provide in-depth risk assessments. Within physics, neural network-based stochastic models are used to simulate complex diffusion processes, track particle movement in unpredictable environments, and explore the probabilistic realm of quantum mechanics. Biologists utilize these techniques to model population growth dynamics, analyze the intricate spread of diseases, and capture the inherent randomness within genetic processes.

For this project, several different datasets were used. First, we tested our methods on several synthetic datasets, including the Wiener process and a modified Wiener process that is still relatively simple, but cannot be easily parameterized by a Gaussian kernel. This was to show that our method can learn relatively simple, but nonparametric processes.

We also tested our method on a couple of real-world datasets. One measures the water level of a lake and the amount of rainfall it receives over time and the other is an aggregation

of stock data for several companies downloaded from Yahoo Finance. Both of these datasets have daily data spanning over several years.

**3.2. Model Applications.** The model can be applied using Monte Carlo sampling methods to generate estimates of the certain statistics such as future mean and volatility/standard deviation. Additionally, traditional models that take a time series roll-out as input (such as a trading model on a financial dataset) can use samples from our time series generator to analyze properties about its distribution. Thus, using these samples, we can quantify uncertainty on these estimates by calculating variances on the metrics.

An example of such a use case is calculating returns on a trading model and then calculating uncertainty on such returns to manage risk. This can bring better estimates of Sharpe ratio and generate safer returns on average.

[1] [2] [12] [3] [5] [7] [8] [9] [13] [6] [10] [11] [14]

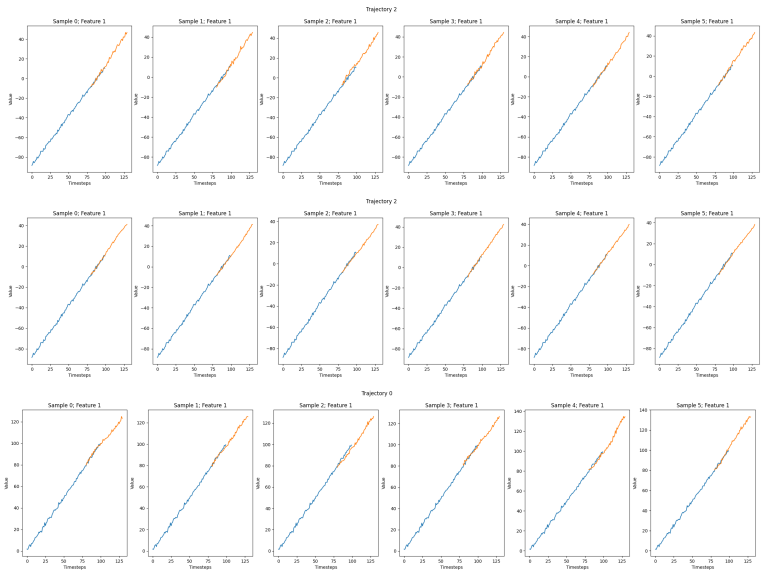

**Figure 4.** *From top to bottom: SAD EMILIE, VAE, and GAN trained on a toy experiment with a small MLP model.*

**4. Results.** (The results got scattered across the pages, see the figures).

We saw samples for future trajectories on all of our datasets and gotta variations between samples that almost always seems reasonable. All the methods were able to learn our toy problems pretty well (at least qualitatively, we should use future statistics to verify this in the future for the toy problems as we have access to these future statistics).

Further, we see qualitatively that when the model has smaller uncertainty bounds it predicts the true data better (the model is more certain on apple and it predicts apple better on average), and that the uncertainty bounds increase over time in the future (which makes sense since further in the future is less predictable).

**5. Limitations and Future Work.** The models struggle to perform well on data that jumps around from zero to high values, like the rainfall data from the lake dataset. Perhaps this

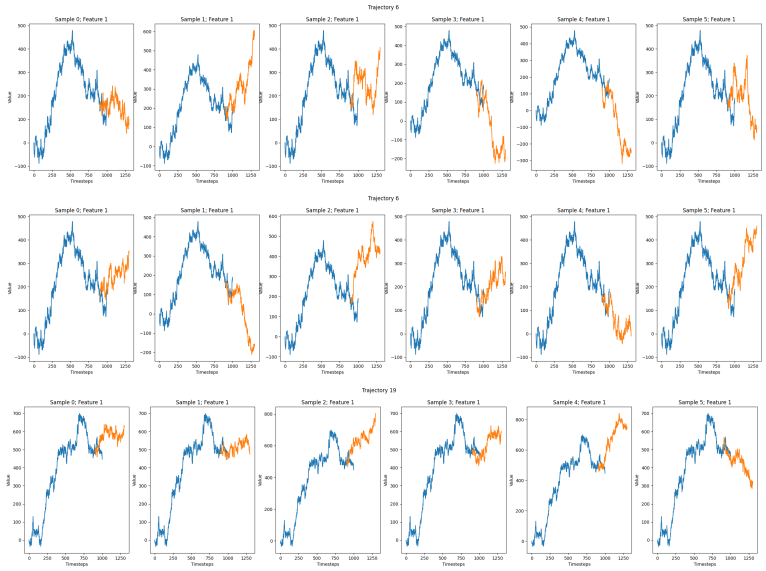

**Figure 5.** *From top to bottom: SAD EMILIE, VAE, and GAN trained on a Wiener process with a small MLP model.*

could be mitigated by preprocessing the data in a smarter, for example, by doing a cumulative sum on the trajectory so the model has an input trajectory that doesn't quite jump around so much. Additionally, in the case of stocks, the model could predict percentage increments rather than absolute increments so it can be applied to more data.

Another issue is that we do not have any quantitative comparisons between methods. For the future, perhaps higher order moments could be computed to serve as an objective measure of the quality of generated trajectories.

Additionally, there are several other model architectures that could be well suited to this problem including diffusion models, transformers, normalizing flows, and SpaceTime, which is a variant of S4 specifically suited for time series data. In the case of SAD EMILIE, different kernel density estimates can be experimented with other than KNN.

Also, more experiments and hyperparemters sweeps can be run to realize the full potential of these methods.

**6. Conclusions..** A model that works well for generic stochastic processes is something that has huge potential for calculating tons of future statistics. This has applications in both industry and research areas. However, this model still has limitations and drawbacks (see next section) and there are still low hanging fruit for model improvement.

From our perspective, we learned a lot about variational inference, generative modelling, and the research process of developing new objectives. Further, we learned about stochastic processes and found out about the many applications of them.

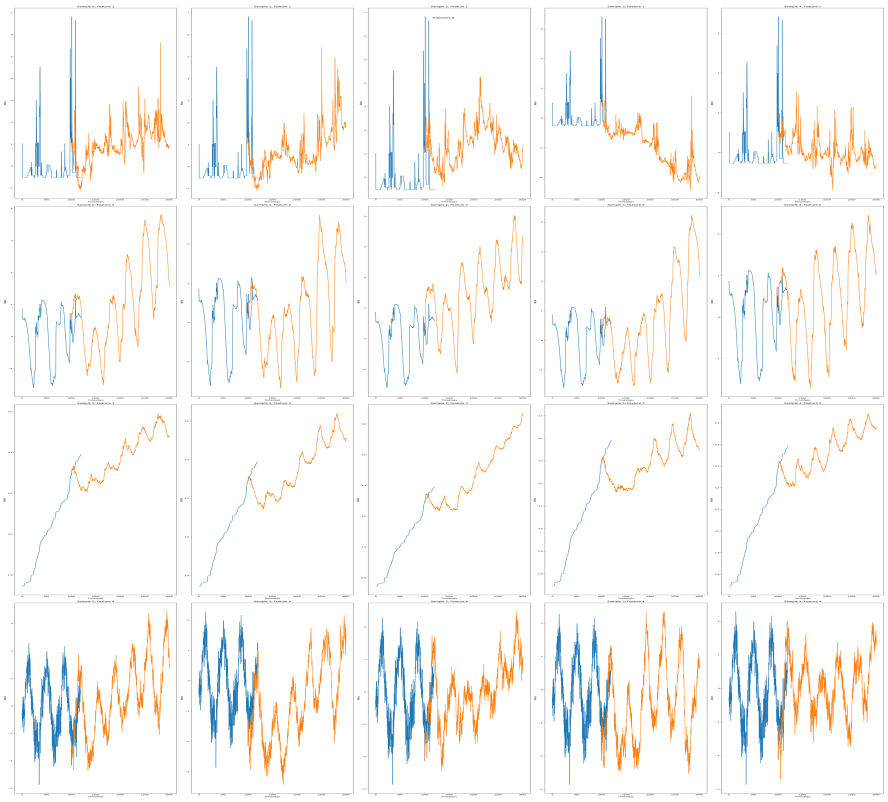

**Figure 6.** *Best model trained so far on lake data. This run is using a GAN with a medium sized MLP.*

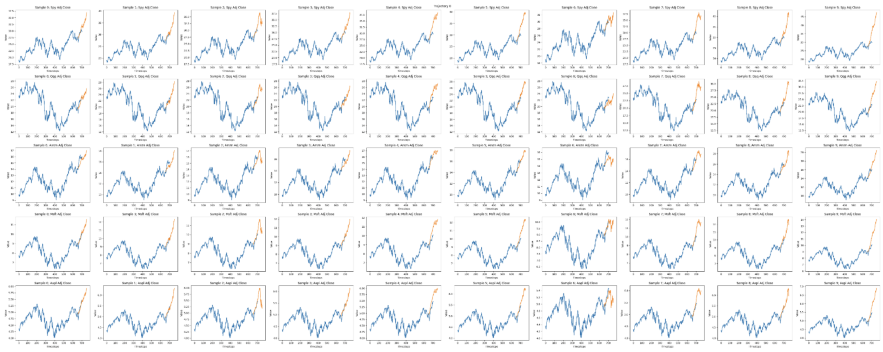

**Figure 7.** *Best model on stock data.*

## REFERENCES

[1] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, *Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44 (2022), p. 7327–7347, https://doi.org/10.1109/tpami.2021.3116668, http://dx.doi.org/10.1109/TPAMI.2021.3116668.

[2] E. Brophy, Z. Wang, Q. She, and T. Ward, *Generative adversarial networks in time series: A survey*

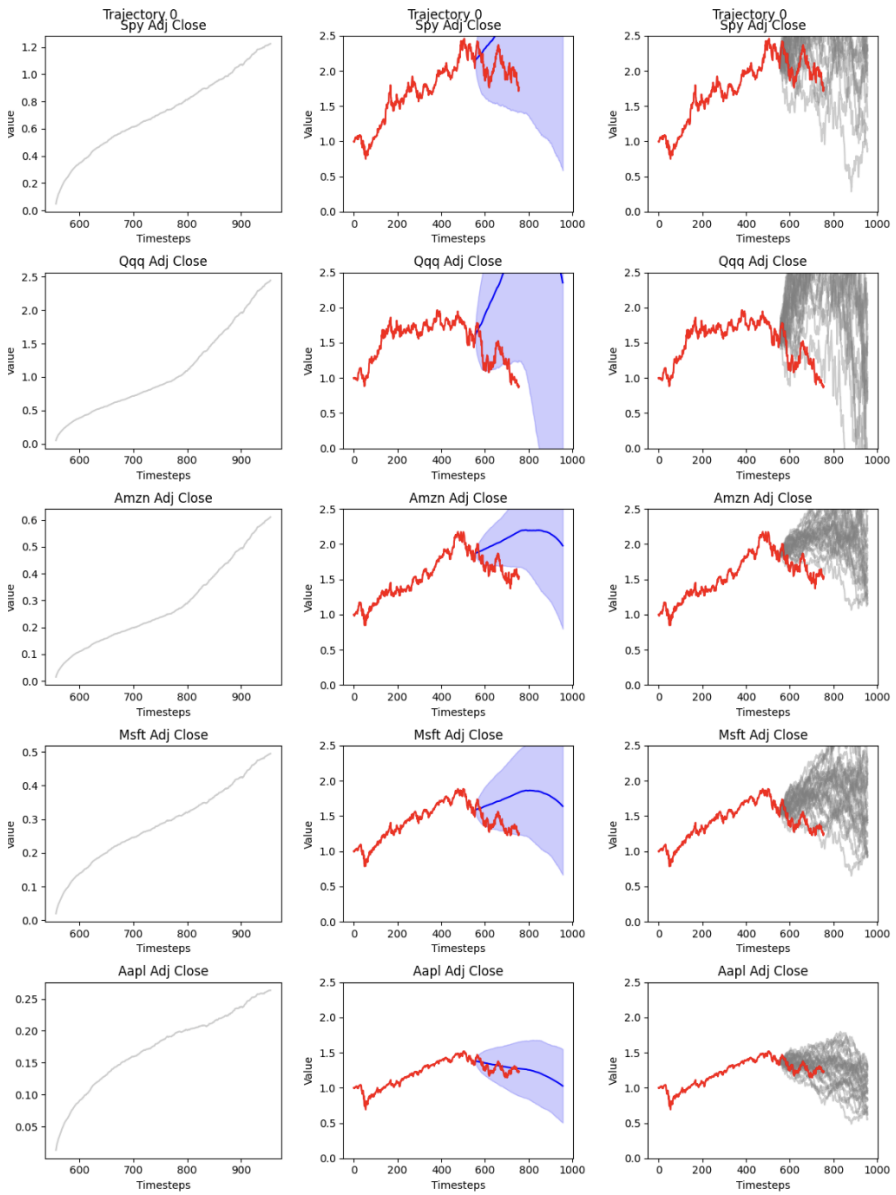


**Figure 8.** *Analysis graphs of the best model on stock data. The left column is the variance of the model prediction over future time, the middle column is mean and 95% confidence intervals of model predictions, and the right column is future trajectory samples from the model (all on stock data).*

*and taxonomy*, 2021, https://arxiv.org/abs/2107.11098.

[3] A. Desai, C. Freeman, Z. Wang, and I. Beaver, *TimeVAE: A variational auto-encoder for multivariate time series generation*, 2022, https://openreview.net/forum?id=VDdDvnwFoyM.

[4] A. Gu, K. Goel, and C. Ré, *Efficiently modeling long sequences with structured state spaces*, 2022, https://arxiv.org/abs/2111.00396.

[5] G. Huerta, W. Jiang, and M. A. Tanner, *Time series modeling via hierarchical mixtures*, Statistica Sinica, 13 (2003), pp. 1097–1118, http://www.jstor.org/stable/24307162 (accessed 2024-03-06).

[6] G. Huerta, W. Jiang, and M. A. Tanner, *Time series modeling via hierarchical mixtures*, Statistica Sinica, 13 (2003), pp. 1097–1118, http://www.jstor.org/stable/24307162 (accessed 2024-03-06).
[7] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, *Modeling long- and short-term temporal patterns with deep neural networks*, 2018, https://arxiv.org/abs/1703.07015.
[8] X. Li, V. Metsis, H. Wang, and A. H. H. Ngu, *Tts-gan: A transformer-based time-series generative adversarial network*, 2022, https://arxiv.org/abs/2202.02691.
[9] L. Lin, Z. Li, R. Li, X. Li, and J. Gao, *Diffusion models for time series applications: A survey*, 2023, https://arxiv.org/abs/2305.00624.
[10] D. Salinas, V. Flunkert, and J. Gasthaus, *Deepar: Probabilistic forecasting with autoregressive recurrent networks*, 2019, https://arxiv.org/abs/1704.04110.
[11] K. E. Smith and A. O. Smith, *Conditional gan for timeseries generation*, 2020, https://arxiv.org/abs/2006.16477.
[12] J. Yoon, D. Jarrett, and M. van der Schaar, *Time-series generative adversarial networks*, in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds., 2019, pp. 5509–5519, https://proceedings.neurips.cc/paper/2019/hash/c9efe5f26cd17ba6216bbe2a7d26d490-Abstract.html.
[13] J. Yoon, D. Jarrett, and M. van der Schaar, *Time-series generative adversarial networks*, in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., vol. 32, Curran Associates, Inc., 2019, https://proceedings.neurips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf.
[14] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, *Informer: Beyond efficient transformer for long sequence time-series forecasting*, 2021, https://arxiv.org/abs/2012.07436.