FinLS4: Deep Latent State Space GAN for Financial Time Series

Prasadh.S
University College London
shyaam.rajan.23@ucl.ac.uk

Kumar.S
University College London
kumar.saurabh.23@ucl.ac.uk

Cango.S
University College London
sreekar.cango.23@ucl.ac.uk

Abstract

We propose 'FinLS4', a novel approach to financial time series forecasting, combining latent state space models with Generative Adversarial Network (GAN) architecture. Addressing limitations in FinGAN literature, our method enhances generative capabilities by incorporating latent variables evolving according to state space ordinary differential equations (ODEs). Additionally, we introduce a novel trading loss function, enabling GANs to be used for classification within supervised learning settings. This architecture streamlines computation by utilizing a convolutional representation of LS4, eliminating the need for explicit evaluation of hidden states. Experimental results demonstrate the robustness of FinLS4, outperforming state-of-the-art methods such as FinGAN, SegRNN, and TLnets in financial time-series forecasting Code and data are available at github/multiagent_ai

1 Introduction

Financial time series are inherently non-stationary and stochastic, characterized by their random walk nature and varying statistical properties across different time periods, which pose significant challenges to accurate prediction. Generative adversarial approaches specifically tailored for financial time series (termed 'FinGAN' (Vuletić et al., 2014)) have not adequately demonstrated robustness in learning sequences with abrupt transitions or in maintaining prediction accuracy over extended sequence lengths.While transformer architectures (Vaswani et al., 2017) offer promising results due to their dense information routing capabilities within context windows, they struggle with modeling data beyond their finite context windows and suffer from quadratic scaling issues. Efforts to mitigate these drawbacks often compromise the transformers' effectiveness. To confront these challenges, we propose a novel GAN architecture, named 'FinLS4', leveraging the capabilities of Structured State Space sequence models (S4 & S6). S4 (Gu et al., 2022a; 2021) and Mamba S6 (Gu & Dao, 2024) models, amalgamating the strengths of Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), provide efficient computation with linear or near-linear scaling concerning sequence length and excel in modeling long-range dependencies. Surpassing alternative models like FinGAN, SEGRNN, and TLnets trained with the BCE loss, our proposed methodology demonstrates superior predictive capabilities for daily stock ETF-excess returns and raw returns, as revealed by extensive numerical analyses. We introduce a set of PNL-based metrics to assess the quality of generated financial time series samples, uncovering LS4's notable outperformance over baselines, particularly on datasets with abrupt transitions, achieving an average of 30% lower MSE scores. Our approach ensures scalability and the ability to capture the collective movements of diverse stocks. Inspired by Sirignano and Cont (2019), we explore the concept of universality across 22 stocks spanning various sectors and 9 sector ETFs, with each sector represented by at least two stocks in the training data. By training the networks on a comprehensive dataset incorporating data from all 31 tickers, we conduct similar experiments within a single sector (XLP). Evaluation on both trained and unseen stocks underscores the potential for significant Sharpe Ratios even with new stock data, underscoring the resilience of our models. Notably, our model trains $\times 20$ faster than baseline methods on sequences with longer context length

2 Related work

State space models (SSMs) are fundamental in dynamical system theory (Chen, 1984) and signal processing (Oppenheim, 1999), also adapted for deep generative modeling, seen in VAE variants by Chung et al. (2015) and Bayer & Osendorfer (2014), later generalized by Franceschi et al. (2020), among others. However, scaling these models to longer sequences is challenging due to unrolling recurrence equations. Our work is inspired by recent advances in deep architectures

like S4 (Gu et al., 2021; 2020), employing deeply stacked linear SSMs initialized with HiPPO to efficiently model long sequences, capturing long-range dependencies across large time scales. Like S4, our generative model uses convolutional representation of SSMs during training and inference, avoiding the need to materialize hidden states of each recurrence, speeding up both processes significantly compared to prior methods. Furthermore, we enhance deep SSMs to model sequential latent variables, increasing their capacity to capture complex temporal dynamics, like stiff transitions, and achieve superior generation results.

In the realm of univariate analysis, machine learning models like deep learning and tree-based methods such as GBRT adeptly capture input feature-target interactions, even in nonlinear scenarios (Goodfellow et al., 2016; Wu et al., 2020). However, there's growing interest in modeling nonlinear cross-asset interactions, where linear models fall short (Wu et al., 2023, 2022). GANs, pioneered by Goodfellow et al. in 2014, lead the way in various tasks, notably image and video generation, with widespread applications across domains like science, video games, and photo editing (Mustafa et al., 2019; Zhang et al., 2020). Their proficiency in generating samples from unknown data distributions holds promise for enhancing forecasting by capturing cross-asset interactions. ForGAN, introduced by Koochali et al. (2019), utilizes a conditional GAN framework tailored for probabilistic time series forecasting. It leverages previous observations, denoted as x_{-L}, to forecast x_{t+1}, enabling richer information utilization and uncertainty estimation. Employing recurrent neural networks (RNNs), specifically gated recurrent units (GRUs) or long short-term memory (LSTM) cells, ForGAN structures both generator and discriminator to exploit the time-series structure effectively. In our implementation, we adopt LS4 cells due to their prevalence in time series analysis. Diagram illustrating ForGAN's architecture with LS4 is depicted in Figure 1.

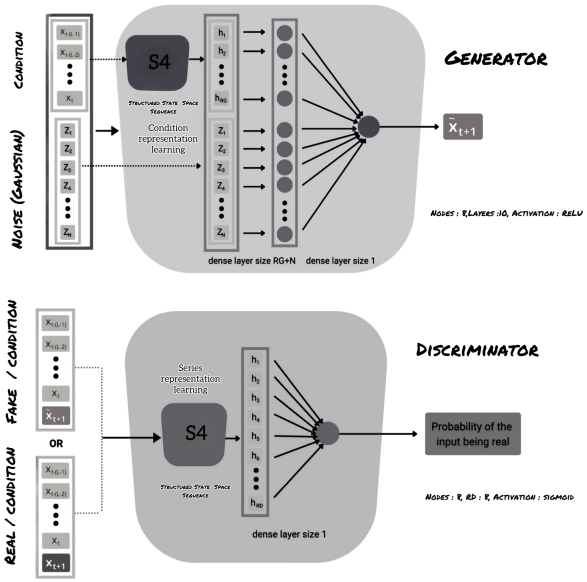

Fig.1 Research Framework - FinLS4

3 Methodology

State Space Models (SSM)

RNNs are renowned for processing sequential data without length constraints, offering versatility in context-driven applications. However, they face challenges such as vanishing and exploding gradients, limiting their learning efficiency. In contrast, CNNs, while having a narrower context scope, excel in parallel processing, handling multiple data points simultaneously. The Transformer model, with its self-attention mechanism, enables concurrent computations during training via dot product

operations. However, its inference process is intricate, with each token processed independently, influenced by the input sequence and attention mechanism-defined context.

Transformers have limitations with long sequences, where state space models (SSMs) excel. However, traditional SSM approaches face challenges in computation and memory. The Structured State Space sequence model (S4) creatively enhances SSM parameterization, notably improving computational efficiency through low-rank updates for stable state matrix diagonalization. The research framework Fig.1 GAN architecture integrates Latent S4 (LS4), employing Structured State Space Models (SSMs) to parameterize essential distributions. These include the generative distribution $p_\theta(x \leq T|z \leq T)p_\lambda(z \leq T)$, the prior distribution $p_\lambda(z \leq T)$, and the inference distribution $q_\phi(z \leq T|x \leq T)$. The structured state space model is described by coupled differential equations:

$$\frac{d}{dt}h_t = \Lambda h_t + \Omega x_t + \Psi z_t \tag{1}$$

$$y_t = \Sigma h_t + \Delta x_t + \Upsilon z_t \tag{2}$$

Here, x_t, y_t, and z_t represent continuous real signals over the time interval $[0, T]$. The function $H(\cdot)$ maps input signals x and z to hidden states h_t at time t, leveraging trainable parameters Λ, Ω, and Ψ. Discretizing the input signals enables convolutional operations, facilitating efficient evaluation where $\Sigma_{tk} = \bar{A}_\sigma \bar{B}_\sigma$, $\Sigma^\dagger_{tk} = \bar{A}_\sigma \bar{E}_\sigma$, which can be evaluated efficiently using FFT. A is HiPPO-initialized for all such SSM blocks.

To streamline the training of state space models and avoid the computational load associated with RNNs, we adopt an expanded computation for the states starting with null initial states. The updates for the states and their corresponding outputs at each time step can be succinctly expressed as:

$$\begin{aligned}
h_{t0} &= \overline{\Omega} x_{t0} \\
y_{t0} &= \overline{\Sigma\Omega} x_{t0} \\
h_{t1} &= \overline{\Lambda\Omega} x_{t1} + \overline{\Omega} x_{t0} \\
y_{t1} &= \overline{\Sigma\Lambda\Omega} x_{t1} + \overline{\Sigma\Omega} x_{t0} \\
&\vdots \\
h_{tK} &= \overline{\Lambda}^K \overline{\Omega} x_{tK} + \overline{\Lambda}^{K-1} \overline{\Omega} x_{K-1} + \cdots + \overline{\Omega} x_{t0} \\
y_{tK} &= \overline{\Sigma\Lambda}^K \overline{\Omega} x_{tK} + \overline{\Sigma\Lambda}^{K-1} \overline{\Omega} x_{K-1} + \cdots + \overline{\Sigma\Omega} x_{t0}
\end{aligned}$$

Given the input sequence x, the output y is efficiently calculated using a convolution with the kernel (K) and a skip connection, resulting in:

$$y = K * x + \overline{\Delta} x \tag{3}$$

In this representation, K is the convolutional kernel constructed from $(\overline{\Omega}, \overline{\Lambda\Omega}, \ldots, \overline{\Lambda}^{L-1}\overline{\Omega})$, and $\overline{\Delta}x$ provides a direct path from the input to the output, enhancing the computation for long sequences.In the realm of VAEs for sequences, there's a vast array of modelling possibilities, particularly in choosing the granularity of the latent space, where 'z' can represent either the entire trajectory or a sequence of variables equivalent in length to the trajectories. Our focus is on the latter approach. Given a set of observed sequence variables $x \leq T$ up to time T, discretized into a sequence $(x_{t0}, ..., x_{tL-1})$ of length L, where $t_{L-1} = T$, a sequence VAE model with parameters θ, λ, ϕ learns both a generative and inference distribution. These distributions are represented as:

$$\begin{aligned}
p_{\theta,\lambda}(x \leq t_{L-1}, z \leq t_{L-1}) &= \prod_{i=0}^{L-1} p_\theta(x_{ti}|x < t_i, z \leq t_i) \cdot p_\lambda(z_{ti}|z < t_i) \\
q_\phi(z \leq t_{L-1}|x \leq t_{L-1}) &= \prod_{i=0}^{L-1} q_\phi(z_{ti}|x \leq t_{ti})
\end{aligned} \tag{4}$$

Here, $z \leq t_{L-1} = (z_{t0}, ..., z_{tL-1})$ denotes the corresponding latent variable sequence. For computational efficiency, the approximate posterior q_ϕ is factorized explicitly as a product of marginals. To this end, we employ the variational lower bound, which involves minimizing the objective:

$$\mathcal{L} = \sum_{i=0}^{L-1} \left(\text{DKL}\left(q_\phi(z_{ti}|x \leq t_i) \| p_\lambda(z_i|z < t_i)\right) - \log p_\theta(x_{ti}|x < t_i, z \leq t_i)\right) \tag{5}$$

Once trained using the above objective, the generative model can sample z_t from the prior p_λ autoregressively. Subsequently, given the sampled z_t, each x_t can be sampled autoregressively using $p_\theta(x_{ti}|x < t_i, z \leq t_i)$.

Forward pass through our SSM consists of two phases: initially, we set the hidden states to adhere to $X_{\beta_1}(0, z, 0, t)$ within the span of $[t_0, t_{n-1}]$. Subsequently, within the interval $(t_{n-1}, t_n]$, we exclusively incorporate the most recent provided latent $z_{t_{n-1}}$ as an additional signal for the outputs, succinctly represented as

$$y_{z,n} = GELU(F_{yz} z_{t_{n-1}} + C_{yz} X_{\beta_1}(0, 0, X_{\beta_2}(0, z[t_0, t_{n-1}], h_{t_{n-1}}, 0, t_{n-1}), t_n)) \tag{6}$$

$y_{z,n}$ corresponds in dimensionality to each z_t and is determined by all z preceding t_n, employed to represent the distribution of z_{t_n}.The above block can be visualized as LS4 prior block which plays a crucial role in modeling the conditional distribution of latent variables. It is represented as:

$$LS4prior(z[t_0, t_n - 1], \psi) = \Lambda(G_{yz} y_{z,n} + b_{yz}) + z_{t_n - 1} \tag{7}$$

In this equation, Λ denotes LayerNorm, and ψ represents a combination of parameters β_i, C_{yz}, F_{yz}, G_{yz}, and b_{yz}. Finally, the autoregressive model generates subsequent latent variables, initialized by parameters $\mu_{z,0}$ and $\sigma_{z,0}$:

$$z_{t_0} \sim N(\mu_{z,0}, \sigma^2_{z,0}) \tag{8}$$

The LS4 generative block diverges from the prior block in its approach, as it accepts both observation and latent sequences as input. It generates intermediate outputs $g_{x,n}$ and $g_{z,n}$, subsequently utilized to form an LS4 generative block represented as:

$$\hat{g}_{x,n} = \text{LN}(G_{gx} g_{x,n} + b_{gx}) + x_{t_{n-1}} \tag{9}$$

$$\hat{g}_{z,n} = \text{LN}(G_{gz} g_{z,n} + b_{gz}) + z_{t_n} \tag{10}$$

$$\text{LS4}_{\text{generator}}(x[t_0, t_{n-1}], z[t_0, t_n], \psi) = (g^{\hat{x},n}, g^{\hat{z},n}) \tag{11}$$

In this setting, ψ incorporates all parameters within the block. The generative block generates two streams of outputs, each following the ResNet-like structure as in the previous model. These outputs can then serve as inputs for subsequent stacks. The final mean value $\mu_{x,n}$ is determined as the outcome of a series of LS4 generative blocks.

The inference block computes

$$\hat{y}_{z,n} = \text{GELU}(C_{y\hat{z}} H_{\beta 5}(x[t_0, t_n], 0, 0, t_{n-1}) + D_{y\hat{z}} x_{t_n}) \tag{12}$$

$$\text{LS4}_{\text{inference}}(x[t_0, t_n], \psi) = \Lambda(G_{y\hat{z}} \hat{y}_{z,n} + b_{y\hat{z}}) + x_{t_n} \tag{13}$$

The LS4 inference model, $\text{LS4}_{\text{inference}}(x[t_0, t_n], \psi)$, processes x over the interval $[t_0, t_n]$, unlike the generative model. It utilizes a stack of inference blocks to derive $\mu^{\hat{z},t}$ and $\sigma^{\hat{z},t}$ from x_t.

Generative Adversarial Networks

The generator $G(z; \theta_g)$ transforms information from the latent space to the data space through a differentiable neural network. Typically, this involves introducing input noise z sampled from a distribution $p_z(z)$, often a multivariate normal distribution. On the other hand, the discriminator $N(x; \phi_d)$, another neural network, maps data points to a range between 0 and 1, estimating the probability of x originating from the true data distribution p_{data}. Throughout the training process, N aims to maximize the accuracy in classifying both real and generated samples, while G aims to minimize the log-probability of N correctly classifying its outputs or maximize the log probability of N making an error. Each network is associated

with its own objective function $L(N)(\phi_d, \theta_g)$, with G striving to minimize the log-probability of N correctly classifying its outputs, and N aiming to minimize a loss function similar to cross-entropy, as shown below:

$$L(N)(\phi_d, \theta_g) = -\mathbb{E}_{p_{\text{data}}}[\ln N(x; \phi_d)] - \mathbb{E}_{p_z}[\ln(1 - N(G(z; \theta_g)); \phi_d)] \tag{14}$$

Loss Function

In the GAN architecture, it is assumed to forecast x_i for $i = 1, \ldots, n_{\text{batch}}$. For each x_i, the generator produces an output $\hat{x}_i$ based on a noise sample z and the condition formed by the previous L values of the time-series x_{-L}, where $\hat{x}_i = G(z, x_{-L})$. The loss function used to train the generator in this GAN architecture comprises three terms: PnL, MSE, and Sharpe Ratio. The Profit and Loss (PNL) is defined as

$$pPNL = \frac{10000}{n} \sum_{i=1}^{n} \text{sign}(\hat{r}e^{A_i}) \hat{r}e^{A_i} \tag{15}$$

The weighted strategy is calculated using:

$$PnL_i = 10000 \left(\text{sign}(\hat{r}e^{A_{2i}}) r_e^{A_{2i}} + \text{sign}(\hat{r}e^{A_{2i+1}}) r_e^{A_{2i+1}} \right), \tag{16}$$

The PnL term, while intuitive for maximizing the generator's training objective, presents challenges due to the non-differentiable sign function, hindering backpropagation. To address this, Following Vuletic et al.(2023),we introduce a smooth approximation, denoted as PnL*, defined as the average of smoothed PnL terms for each forecast pair.

$$PnL^*(x, \hat{x}) = \sum_{i=1}^{n_{\text{batch}}} PnL_a^*(x_i, \hat{x}_i) \tag{17}$$

$$PnL_a^*(x_i, \hat{x}_i) = \tanh(k_{\text{tanh}} \hat{x}_i) \cdot x_i \tag{18}$$

where PnL_a^* is a smooth approximation to PnL for each forecast. The hyperparameter k_{tanh} governs the accuracy of our approximation, balancing between precision and gradient strength. We set $k_{\text{tanh}} = 100$ to handle small excess return values with accuracy and learnability property.We aim to ensure that $\hat{x}_i$ closely matches x_i while also maintaining consistency with the forecast's sign. To achieve this, we integrate a Mean Squared Error (MSE) term into the training objective.

$$\frac{1}{n_{\text{batch}}} \text{MSE}(x, \hat{x}) = \sum_{i=1}^{n_{\text{batch}}} (x_i - \hat{x}_i)^2. \tag{19}$$

The Sharpe ratio term is defined as

$$SR^*(x, \tilde{x}) = \frac{\frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} PnL_a^*(x_i, \tilde{x}_i)}{\sqrt{\frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} \left(PnL_a^*(x_i, \tilde{x}_i) - \frac{1}{n_{batch}} \sum_{j=1}^{n_{batch}} PnL_a^*(x_j, \tilde{x}_j) \right)^2}}, \tag{20}$$

The Standard deviation term is defined as

$$STD(x, \tilde{x}) = \sqrt{\frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} \left(PnL_a^*(x_i, \tilde{x}_i) - \frac{1}{n_{batch}} \sum_{j=1}^{n_{batch}} PnL_a^*(x_j, \tilde{x}_j) \right)^2}, \tag{21}$$

Hence the loss function is defined as :

$$Loss_G(x, \hat{x}) = K(G)(x, \hat{x}) - \alpha PnL^*(x, \hat{x}) + \beta MSE(x, \hat{x}) - \gamma SR^*(x, \hat{x}) + \delta STD(x, \hat{x}), \tag{22}$$

$Loss_G(x, \hat{x})$ serves as the loss function, while $K(G)(x, \hat{x})$ denotes the objective function or cost linked with the generator G. $PnL^*(x, \hat{x})$ denotes the profit and loss, whereas $MSE(x, \hat{x})$ quantifies the mean squared error between predicted and actual values. Additionally, $SR^*(x, \hat{x})$ represents the Sharpe Ratio, utilized for maximizing risk-adjusted returns, and $STD(x, \hat{x})$ signifies the standard deviation of profit and loss. Furthermore, α, β, γ, and δ denote the weighting coefficients assigned to each respective term within the loss function.

4 Data

Using data from CRSP on Wharton Research Data Services, we investigate the daily stock ETF-excess returns and raw ETF returns spanning from January 2000 to December 2021, following the methodology outlined by Vuletić et al. (2023). This analysis encompasses ETF-excess returns for 22 different stocks across nine sectors, along with the raw returns of the nine sector ETFs, all of which are currently included in the S&P500 index. To ensure comprehensive analysis, the dataset is partitioned into training, validation, and testing sets in an 80-10-10 split. The analysis incorporates sliding windows and condition windows, with the former spanning a single time unit (t) and the latter extending over ten time units ($t-10$ to t), offering crucial context for predicting future market movements. Positioned one time unit ahead, the Prediction Window ($t+1$) facilitates the forecasting of future returns. Prior to analysis, the dataset undergoes pre-processing steps, including price adjustments and capping returns at $\pm 15\%$ to address potential outliers. Matrices are then constructed for each of the three data sets, organizing them into columns that represent the condition window ($x_1, \ldots, x_{10}$) and the target variable (x_{11}).Table 1 describes the ETFs and stocks in dataset.

Table 1: Sector names, ETF tickers, and stock tickers

Sector name	ETF ticker	Stock tickers
Consumer Discretionary	XLY	AMZN, HD, NKE
Consumer Staples	XLP	CL, EL, KO, PEP
Energy	XLE	APA, OXY
Financial	XLF	WFC, GS, BLK
Health Care	XLV	PFE, HUM
Industrial	XLI	FDX, GD
Technology	XKL	IBM, TER
Materials	XLB	ECL, IP
Utilities	XLU	DTE, WEC

5 Experiments

This study focuses on accurately classifying the returns or excess returns of ETFs, especially during periods of significant price fluctuations. Rather than striving to generate forecasts that closely match exact values, our emphasis lies in identifying trends. This choice stems from recognizing that while forecasts may closely follow realized values, they might fail to predict the correct trend. In financial time series forecasting, accurately identifying trends outweighs precise value prediction due to its direct impact on Profit and Loss. Our objective is to enhance existing architectures with additional data to better replicate the directional aspects of the data, crucial for our analysis. We utilize three key metrics to gauge the trading forecast of the FinLS4 model. Overnight PnL assesses gains or losses from trades held overnight, revealing how positions react to overnight market movements and risks. Intraday Cumulative PnL measures profits or losses within a single trading day, indicating daily strategy efficacy and intraday price trends. Cumulative PnL encompasses both intraday and overnight trades, offering a holistic perspective on total profit or loss over a specified period for long-term performance assessment.Due to computational constraints, we focused our LS4 architecture analysis on four carefully chosen stocks from different sectors for market-wide insights. AMZN and AZO from the Consumer Discretionary sector are sensitive to economic fluctuations and consumer behavior. EL represents the Consumer Staples sector, often stable even in economic slumps. GS stands for the Finance sector, indicative of banking and financial performance. These stocks were also selected for their frequent news coverage, which significantly influences stock volatility and trends.

We implemented FinLS4 to analyze the PnL dynamics of selected stocks from October 2019 to January 2022, focusing on Cumulative, Intraday, and Overnight PnLs. The analysis involved 500 epochs for SegRNN and TLNets, while FinLS4, which converged more quickly, required only 5 epochs. We compared these results against established benchmark architectures such as the original FinGAN by Vuletić et al. (2023), SegRNN by Lin et al. (2023), and TLFNets by Wang et al. (2023). In addressing the limitations of conventional recurrent neural networks (RNNs) like LSTM and GRU, which struggle with long-range dependencies due to vanishing gradients, we explored the SegRNN approach. SegRNN, by segmenting input sequences into fixed-length segments, reinforces gradient flow and memory retention, thereby potentially enhancing model performance.

Time series often face challenges of diminished performance and explainability, particularly as the prediction horizon lengthens. To tackle this, we juxtaposed our findings with TLFNets, which introduces specialized building blocks including FT-Matrix, FT-SVD, FT-Conv, and Conv-SVD. These blocks are meticulously crafted to capture both global and local information effectively. While FT and SVD blocks shine in capturing overarching trends, Conv blocks zoom in on local nuances. The matrix block, meanwhile, amalgamates these capabilities, enabling the simultaneous assimilation of global and local features, thereby augmenting the model's predictive prowess.

This evaluation coincided with the COVID-19 pandemic, providing insights into the model's robustness under uncertain market conditions. From Fig.2, the first chart illustrates the overnight P&L, where AMZN notably excels, exhibit-

ing a robust and continuous upward P&L trend. In contrast, GS maintains a fairly stable line with a marginal downtrend. AZO and EL experience fluctuations, particularly EL, which sees a pronounced drop before a partial recovery. The second chart, capturing intraday P&L, presents a picture of higher volatility for AMZN and AZO, both eventually trending positively despite significant ups and downs. GS, on the other hand, shows an early peak followed by a consistent downward trajectory, hinting that intraday strategies might be less beneficial for this stock. EL's flat and slightly negative trend suggests minimal intraday profitability. The third chart consolidates both overnight and intraday P&L, reinforcing AMZN's substantial growth and confirming the effectiveness of the applied strategies over both time frames. AZO shows an overall growth yet with greater unpredictability compared to AMZN. Conversely, GS and EL are on a downward trend, with GS gradually descending and EL marked by considerable volatility and an overall decline in P&L.

In the TLF Nets comparison of cummulative P&L(Fig.3.7-3.10), the ForGAN-FT-Matrix model showcased AMZN's strong performance, mirroring the positive outcomes seen in the LS4 architecture and suggesting that both may have effective strategies for this stock. However, a notable difference was observed with GS in the FT-Matrix model, which experienced a marked decline, deviating from its relatively stable LS4 performance. The ForGAN-Conv-SVD and ForGAN-FFT-Conv models presented a more challenging environment across all stocks, with GS facing particular hardship, a stark contrast to the milder negative impact seen in LS4. On a different note, the ForGAN-F-SVD model revealed a degree of resilience, with AZO and GS recovering over time, which stands in relief against the generally downward trend these stocks experienced in the LS4 framework.

When comparing the LS4 and TLF Net architectures, multiple interpretations emerge, highlighting the strengths of each model in different contexts. If the primary objective is to achieve high returns specifically for AMZN, both LS4 and the ForGAN-FT-Matrix TLF Net demonstrate promising results. However, if the goal shifts towards a stable and less volatile performance across a range of stocks, the LS4 architecture may be the preferred option due to its relative consistency. On the other hand, for strategies that value the ability to rebound from early losses, the ForGAN-F-SVD TLF Net shines, offering resilience and a strong recovery potential in its performance. For brevity, we didn't include the discussion of overnight and intraday P&L.

Fig.2 Cummulative / Overnight/ Intraday PNL - LS4

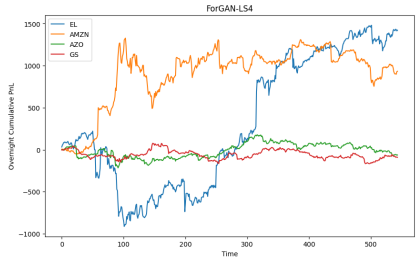

Fig.2.1

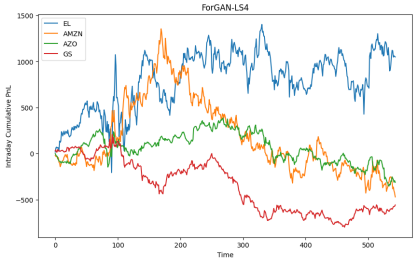

Fig.2.2

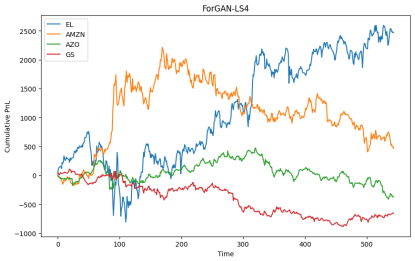

Fig.2.3

Relative to the LS4 model, the ForGAN-SegRNN (Fig.3.4-3.6) exhibits notably improved performance, especially with AMZN during overnight P&L assessments and AZO overall. GS which is on a downtrend, does not experience as severe a decline as seen with certain other TLF Net models. This trend suggests the ForGAN-SegRNN may handle AMZN and AZO more effectively, potentially making it the model of choice due to its robustness across various market conditions, as demonstrated in both intraday and overnight analyses. The enhanced performance of the ForGAN-SegRNN is partially attributed to its ability to filter market noise, a benefit of its segmented approach, which could account for its relative resilience against the inherent volatility in financial data. Nevertheless, it's crucial to acknowledge that these findings are based on a limited sample of just four stocks; a comprehensive conclusion about the superior performance of ForGAN-SegRNN cannot be drawn without broader analysis to determine if the advantages are specifically due to the SegRNN component or the integrated approach of GAN combined with SegRNN.

In the Table 1 below, the ForGAN-SegRNN model stands out with its notably lower MAE and RMSE values across all evaluated stocks, hinting at superior predictive precision when compared with other models, including ForGAN-LS4 and ForGAN-Conv-SVD. Specifically, the MAE for ForGAN-SegRNN for AMZN is at 0.0135 and RMSE at 0.0108, whereas ForGAN-LS4 reports higher values of MAE at 0.5013 and RMSE at 0.5005 for the same stock. The SegRNN model typically outperforms the LS4 in terms of PnL_w as well, with ForGAN-SegRNN achieving a PnL_w

Table 1: Performance Metrics for Different GAN architectures

Type	Ticker	RMSE	MAE	SR scaled	PnL_w	Close-to-Open SR	Open-to-Close SR	Corr
ForGAN-LS4	AMZN	0.5013	0.5005	0.1970	0.8512	0.6623	-0.2617	0.0044
ForGAN-LS4	AZO	0.2083	0.2057	-0.3671	-0.6876	-0.1280	-0.3334	0.0150
ForGAN-LS4	EL	0.5751	0.5744	0.8034	4.544	0.8261	0.3954	-0.0052
ForGAN-LS4	GS	0.2202	0.2186	-0.7721	-1.1931	-0.2568	-0.7046	-0.0078
ForGAN-SegRNN	AMZN	0.0135	0.0108	0.2400	0.7234	0.7115	-0.2551	0.0091
ForGAN-SegRNN	AZO	0.0121	0.0084	0.3270	1.3193	-0.5527	0.5994	0.0393
ForGAN-SegRNN	EL	0.0147	0.0101	0.9423	6.6726	0.9080	0.5516	0.0247
ForGAN-SegRNN	GS	0.0082	0.0055	0.9228	1.6837	0.9430	0.5623	0.0255
ForGAN-LSTM	AMZN	0.0109	0.0073	-0.2891	-0.6320	-0.9237	0.3169	-0.0124
ForGAN-LSTM	AZO	0.0120	0.0080	-0.5504	-1.0979	-0.1168	-0.5376	-0.0348
ForGAN-LSTM	EL	0.0119	0.0075	0.1740	0.3229	1.0347	-0.4214	-0.0457
ForGAN-LSTM	GS	0.0104	0.0084	-0.7087	-3.5766	-0.2349	-0.6459	-0.0230
ForGAN-Conv-SVD	AMZN	0.0194	0.0164	-0.3869	-3.2038	-0.8738	0.1827	-0.0832
ForGAN-Conv-SVD	AZO	0.0138	0.0098	-0.0793	-0.4730	-0.4501	0.1316	0.0043
ForGAN-Conv-SVD	EL	0.0122	0.0080	-0.1807	-0.7321	-0.2708	-0.0428	0.0217
ForGAN-Conv-SVD	GS	0.0080	0.0053	0.1393	0.1492	0.0319	0.1396	-0.0296
ForGAN-F-SVD	AMZN	0.0109	0.0075	0.4302	0.6009	0.0962	0.4718	0.0695
ForGAN-F-SVD	AZO	0.0143	0.0105	0.1657	1.0424	-0.4789	0.4831	-0.0001
ForGAN-F-SVD	EL	0.0134	0.0089	0.3335	2.0238	0.8244	-0.1111	-0.0089
ForGAN-F-SVD	GS	0.0081	0.0054	0.1038	0.1383	-0.3144	0.3584	0.0007
ForGAN-FFT-Conv	AMZN	0.0145	0.0118	0.1945	1.1390	-0.2752	0.4618	0.1016
ForGAN-FFT-Conv	AZO	0.0206	0.0152	-0.1577	-1.1759	0.3007	-0.3581	0.0006
ForGAN-FFT-Conv	EL	0.0166	0.0126	-0.7966	-5.4465	-0.9983	-0.2876	0.0343
ForGAN-FFT-Conv	GS	0.0097	0.0070	1.8309	5.2968	1.0925	1.3474	0.0666
ForGAN-FT-Matrix	AMZN	0.0128	0.0094	1.4069	7.4126	1.8816	0.3016	0.0986
ForGAN-FT-Matrix	AZO	0.0157	0.0124	-0.1551	-1.4587	-0.1584	-0.0895	0.0242
ForGAN-FT-Matrix	EL	0.0137	0.0102	-0.8027	-2.6914	-0.8719	-0.3563	0.0124
ForGAN-FT-Matrix	GS	0.0085	0.0060	-0.4347	-0.9253	-0.3195	-0.3162	0.0100

of 0.3270 for AMZN, in contrast to ForGAN-LS4's -0.6623, suggesting that SegRNN's predictions could be more profitable.In terms of Sharpe Ratios, the SegRNN maintains robust performance: it posts positive values in the Close-to-Open for AZO and AMZN, signaling strong overnight results. The ForGAN-SegRNN demonstrates a Close-to-Open SR of 0.7115 for AMZN, whereas ForGAN-LS4 shows a negative value of -0.6267. Similarly, during the trading day, the SegRNN's Open-to-Close SR_w metrics are mostly positive, with AMZN and GS showing notable performance, indicating effective intraday strategies. ForGAN-SegRNN shows an Open-to-Close SR of 0.2551 for AMZN and 0.9390 for GS, while ForGAN-LS4 shows -0.2617 and -0.7046, respectively.While the LS4 model performs better than some TLNet models, it doesn't quite match the efficacy of the SegRNN. However, it's important to consider that these metrics alone may not capture all scenarios where LS4 could excel, as its performance could vary with different market conditions, financial periods, or may be influenced by the specific stocks considered.

6 Conclusion

We introduce Fin LS4, a powerful generative model with latent space evolution following a state space ODE. Our model is built using GAN architecture with a deep stack of LS4 prior/generative/inference blocks, which are trained via standard sequence VAE objectives. Experimentally, we demonstrate the modelling power of LS4 on financial time series dataset with a wide variety of temporal dynamics and show significant improvement in generation, interpolation, and extrapolation quality. Moreover, our model shows a $\times 100$ speed-up in training and inference time on long sequences. LS4 demonstrates improved expressivity and computational efficiency. Our proposed Fin-GAN methodology was shown to be able to significantly improve Sharpe Ratio performance, shift generated distributions, as well as help alleviate mode collapse issues, the latter of which is a standard challenge in many GAN-based approaches. While our study introduces a novel framework for forecasting time series using the combined power of GAN and LS4, it is important to recognize several limitations. Firstly, our method heavily depends on computational resources, and the scalability of our approach may be constrained by hardware limitations. The process of predicting for all the time series may require substantial computational power, potentially limiting its use in resource-limited settings. Variations in model architectures, dataset compositions, and training methods could impact the results of our approach. Second, we consider daily frequency of data but the high frequency of data may yield different results. Despite these limitations, our study marks a significant advancement in financial time series forecasting, beating existing state-of-the-art in FinGAN literature. Our work raises future directions, including combining transformers with FinLS4, incorporating new loss function terms with different metrics of the portfolio, and moving beyond equity data to explore other classes of assets.

References

[1] Vuletić, M., Prenzel, F., & Cucuringu, M. (2024). Fin-GAN: Forecasting and classifying financial time series via generative adversarial networks. *Quantitative Finance*, 1-25.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS)*.

[3] Gu, A., Goel, K., & Ré, C. (2022). Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

[4] Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., & Ré, C. (2021). Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in Neural Information Processing Systems, 34*, 572-585.

[5] Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

[6] Sirignano, J., & Cont, R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. Quantitative Finance, 19(9), 1449–1459. ISSN 1469-7688.

[7] Chen, C.-T. (1984). Linear system theory and design. Saunders College Publishing.

[8] Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In Advances in neural information processing systems, 28, 2015.

[9] Bayer, J., & Osendorfer, C. (2014). Learning stochastic recurrent networks. arXiv preprint arXiv:1411.7610.

[10] Franceschi, J.-Y., Delasalles, E., Chen, M., Lamprier, S., & Gallinari, P. (2020). Stochastic latent residual video prediction. In International Conference on Machine Learning (pp. 3233–3246). PMLR.

[11] Goodfellow, J. (2017). NIPS 2016 Tutorial: Generative Adversarial Networks. arXiv, abs/1701.00160.

[12] Wu, S., Pan, F., Chen, G., Long, C., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems.

[13] Wu, J., Li, Z., Liu, Y., Li, Y., & Cucuringu, M. (2023). Symphony in the Latent Space: Provably Integrating High-dimensional Techniques with Non-linear Machine Learning Models. In Proceedings of the 2023 AAAI Conference on Artificial Intelligence.

[14] Mustafa, D., Bard, W., Bhimji, Z., Lukić, Z., Al-Rfou, R., & Kratochvil, J. M. (2019). CosmoGAN: creating high-fidelity weak lensing convergence maps using generative adversarial networks. Computational Astrophysics and Cosmology, 6(1), May 2019. ISSN 2197-7909.

[15] Zhang, G., Zhong, J., Dong, S., Wang, S., & Wang, Y. (2019a). Stock Market Prediction Based on Generative Adversarial Networks. *Procedia Computer Science, 147*, 400–406. ISSN 1877-0509.

[16] Zhang, Z., & Zohren, S. (2021). Multi-horizon forecasting for limit order books: Novel deep learning approaches and hardware acceleration using intelligent processing units. arXiv preprint arXiv:2105.10430.

[17] Zhang, Z., Zohren, S., & Roberts, S. (2019b). Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing, 67*(11), 3001–3012.

[18] Koochali, P., Schichtel, A., Dengel, A., & Ahmed, S. (2019). Probabilistic Forecasting of Sensory Data With Generative Adversarial Networks–ForGAN. *IEEE Access, 7*, 63868–63880.

[19] Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, Haotong Zhang. *SegRNN: Segment Recurrent Neural Network for Long-Term Time Series Forecasting*. ArXiv preprint arXiv:2308.11200 (2023). https://arxiv.org/abs/2308.11200.

[20] Wei Wang, Yang Liu, Hao Sun. *TLNets: Transformation Learning Networks for long-range time-series prediction*. ArXiv preprint arXiv:2305.15770 (2023). https://arxiv.org/abs/2305.15770.

Appendix

Cummulative / Overnight/ Intraday PNL - LSTM

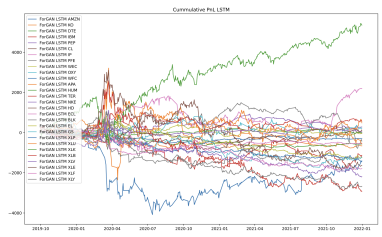

Fig.3.1

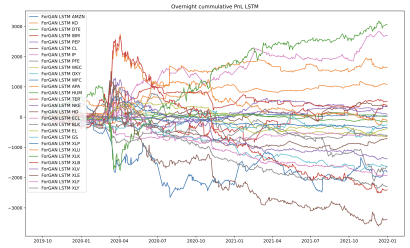

Fig.3.2

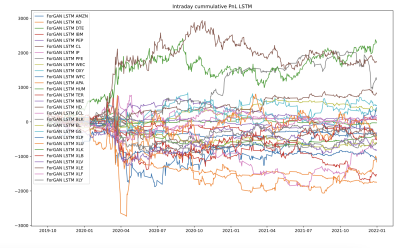

Fig.3.3

Cummulative / Overnight/ Intraday PNL - SegRNN

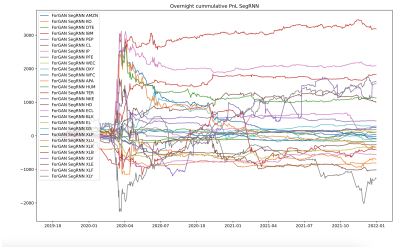

Fig 3.4

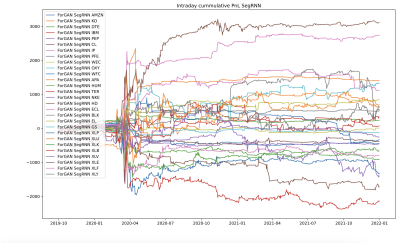

Fig.3.5

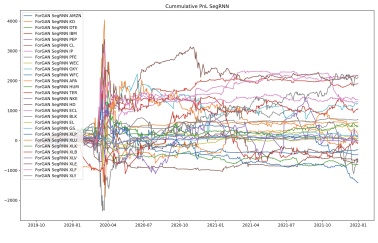

Fig.3.6

Cummulative - TLNets

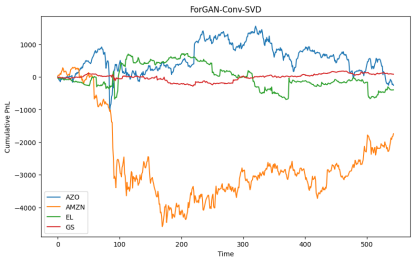

Fig.3.7 - Conv-SVD

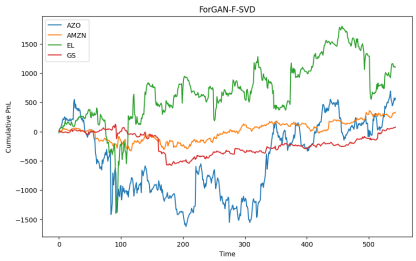

Fig 3.8 - FT-SVD

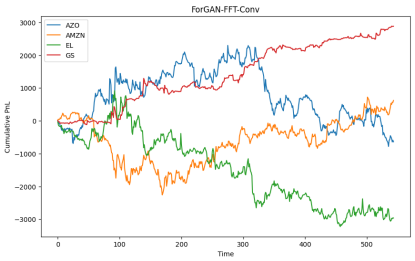

Fig.3.9 - FT-Conv

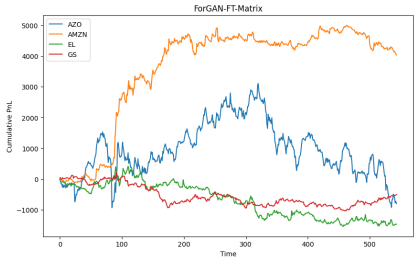

Fig.3.10 - FT-Matrix

Intraday - TLNets

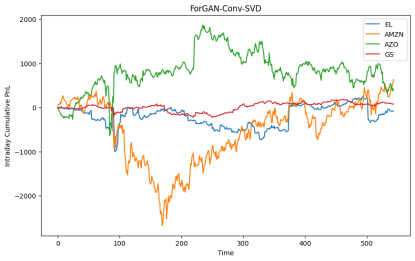

Fig.3.11 - Conv-SVD

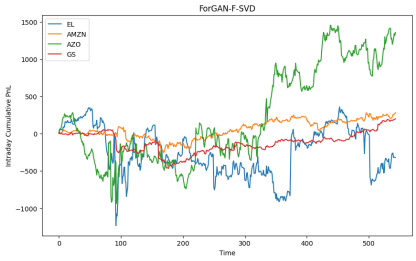

Fig 3.12 - FT-SVD

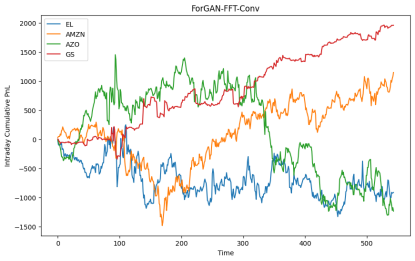

Fig.3.13 - FT-Conv

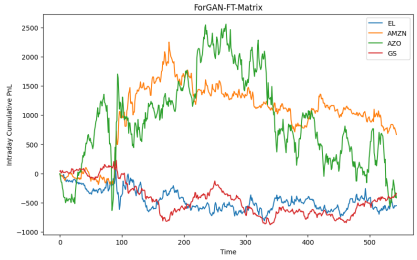

Fig.3.14 - FT-Matrix

Overnight - TLNets

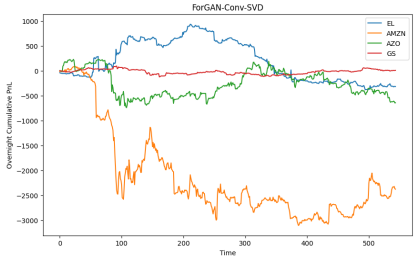

Fig.3.15 - Conv-SVD

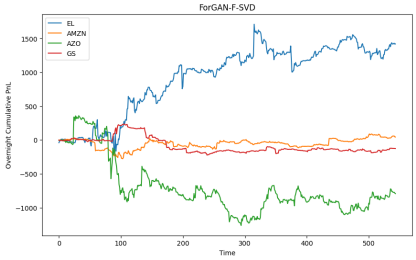

Fig 3.16 - FT-SVD

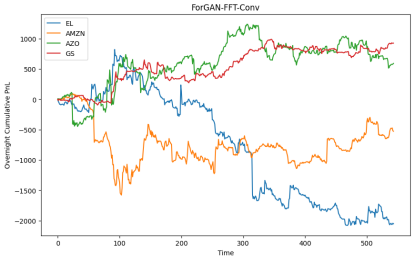

Fig.3.17 - FT-Conv

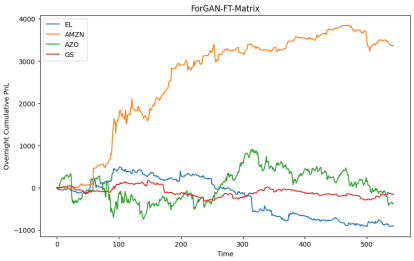

Fig.3.18 - FT-Matrix

PNL distribution- LS4 comparison

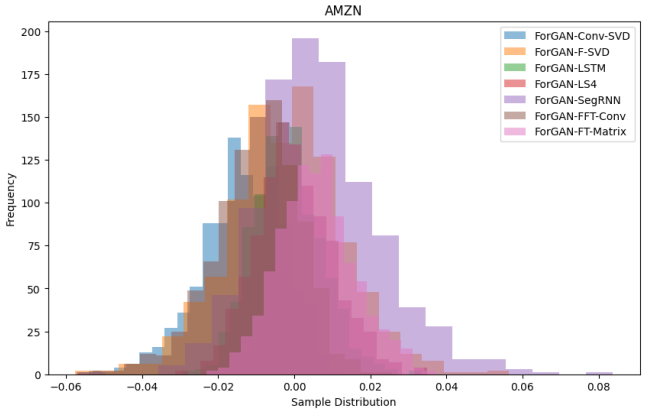

Fig.3.19

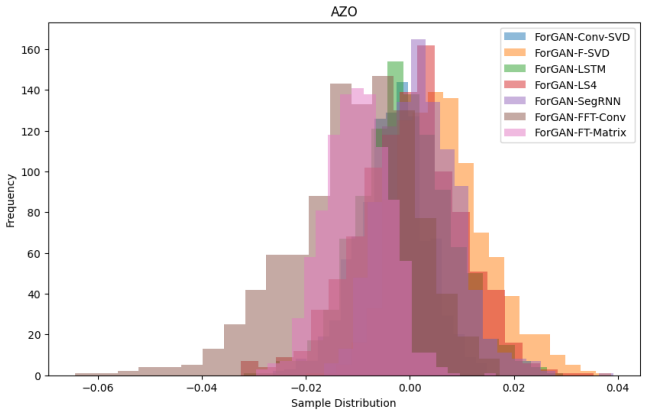

Fig.3.20

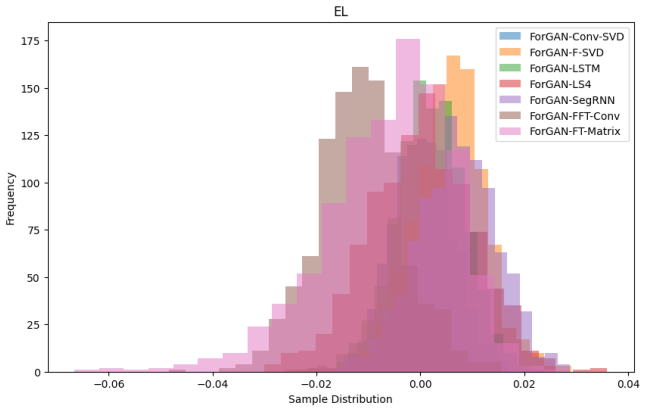

Fig.3.21

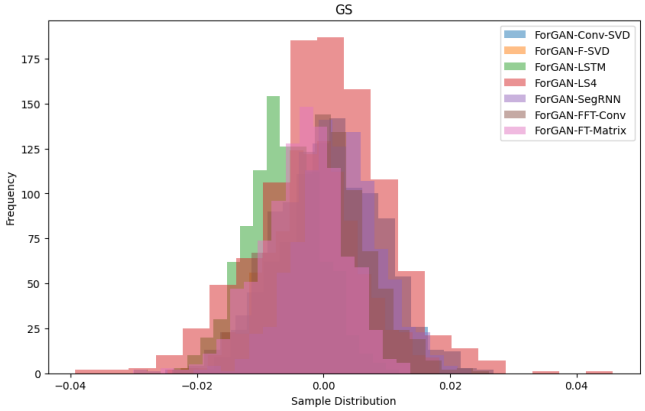

Fig.3.22