Teaching a GAN to Fish: Probabilistic Forecasting in Financial Markets

Report

SN 24220958
COMP0162 Advanced Machine Learning in Finance

ABSTRACT

This study evaluates the effectiveness of a simplified Fin-GAN framework—an adversarial model for financial forecasting—applied to updated stock and ETF return data spanning 2010–2025. Fin-GAN extends traditional GANs with financial-domain loss functions targeting profit and risk optimization. Its performance is benchmarked against LSTM and ARIMA models, using RMSE, Sharpe Ratio, and Profit and Loss as evaluation metrics. Results show that fine-tuned GANs achieve superior risk-adjusted returns in high-volatility sectors, while LSTMs perform better in stable environments. Findings reaffirm that training models with domain-specific financial objectives can outperform traditional error-based optimization, though GANs require careful tuning and remain computationally intensive.

1 Introduction

Time-series forecasting and classification have long been central to both academic and industry applications, particularly within the financial sector, with stock price prediction among the most complex tasks due to its dependence on company's intrinsic value and potential future earnings, investors' sentiment, and volatile macroeconomic factors. Traditional methods typically focus on point estimates, which fail to capture the uncertainty inherent in financial markets.

A more sophisticated approach is presented by Fin-GAN [1] that addresses this limitation using Generative Adversarial Networks (GANs) [2] to model full conditional return distributions. It incorporates a custom, finance-aware loss function that includes objectives such as maximizing Profit and Loss (PnL), improving the Sharpe Ratio, and minimizing volatility.

The idea is to feed the model information about real data, so to place it in a supervised setting, thus making it more suitable for classification tasks and enabling uncertainty-aware decision-making. Building on the ForGAN [3] architecture, Fin-GAN combines GANs with

Recurrent Neural Networks(RNNs) to capture temporal dependencies in financial time series. Empirical evidence shows that Fin-GAN outperforms traditional models like LSTM and ARIMA in terms of risk-adjusted return and achieves mean and median Sharpe Ratios significantly above 1, despite occasionally reporting higher RMSE.

Fin-GAN was tested on 22 stocks and 9 sector ETFs, maintaning strong performance on out-of-sample data. One of its key strengths lies in its use of uncertainty estimates to inform weighted trading decisions, reducing trade sizes when confidence is low, which in turn lowers variance and improves Sharpe Ratios. The approach balances predictive accuracy and financial realism.

This project replicates and simplifies the Fin-GAN framework, using updated financial data (2010-2025) marked by different economic and political conditions and compares its performance against an LSTM model (Long-Short-Term Memory)[4] and a traditional econometric model, ARIMA [5]. In addition, I performed a manual grid-free fine-tuning of key model hyperparameters - originally fixed in the Fin-GAN implementation - to assess the sensitivity of financial performance to architectural and loss function variations.

2 Methodology

2.1 Generative Adversarial Networks (GANs)

GANs, introduced by Goodfellow et al. (2014), are generative models composed of two neural networks—a **generator** G and a **discriminator** D—trained in an adversarial fashion. The generator attempts to produce samples that resemble the real data distribution, while the discriminator learns to distinguish between real and generated data.

The classical GAN loss functions are defined as:

- **Discriminator loss**:

$$\mathcal{L}_D = -\frac{1}{2}E_{x\sim p_{\text{data}}}[\log D(x)] - \frac{1}{2}E_{z\sim p_z}[\log(1 - D(G(z)))]$$

- **Generator loss**:

$$\mathcal{L}_G = -\frac{1}{2}E_{z\sim p_z}[\log D(G(z))]$$

Here, z is a noise vector sampled from a known distribution (typically Gaussian), and x is a real data sample. The generator learns to map noise to the data space: $G(z) \to x$, while the discriminator outputs the probability that a given sample is real.

2.2 Fin-GAN Architecture and Training Procedure

In this work, I adapt the Fin-GAN approach, which builds on the ForGAN framework [3]. ForGAN is a *conditional GAN* architecture designed for time series forecasting, where both the generator and discriminator are conditioned on the previous L values of the time series.

This makes the model suitable for one-step-ahead forecasting. Figure 1 shows the model's architecture.

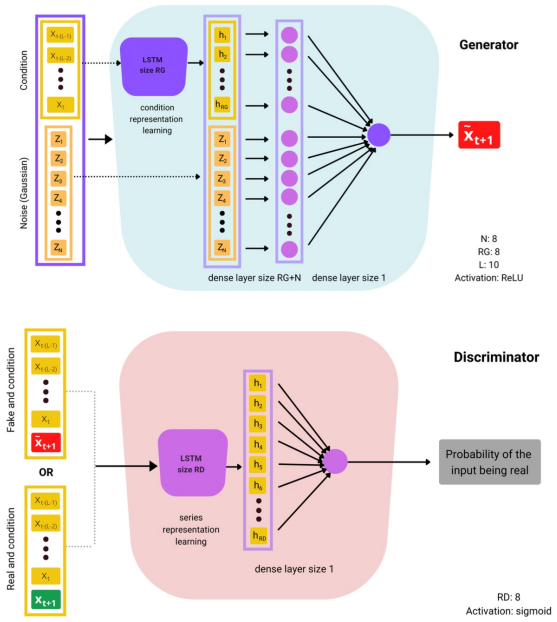

Figure 1: ForGAN architecture using an LSTM cell.

a. LSTM Integration

To model temporal dependencies, both the generator and the discriminator pass the input time series through a Long Short-Term Memory (LSTM) layer. LSTM cells are particularly effective in capturing sequential patterns and long-range dependencies in time series data. Each input is a window of L past returns $x_{t-L+1}, \ldots, x_t$, and the generator predicts the return at time $t+1$, denoted $\hat{x}_{t+1}$.

b. Loss Function with Financial Objectives

Beyond the standard objective of minimizing discriminator loss, Fin-GAN introduces a custom generator loss that aligns training with financial objectives such as maximizing profit and minimizing risk. The full generator loss function is defined as:

$$\mathcal{L}_G^{\text{Fin}} = \mathcal{L}_{\text{BCE}} - \alpha \cdot \text{PnL}^* + \beta \cdot \text{MSE} - \gamma \cdot \text{SR}^* + \delta \cdot \text{STD}$$

Where:

- $\mathcal{L}_{\text{BCE}}$ is the binary cross-entropy loss used in classical GANs.
- PnL^* is a *smooth approximation* of profit and loss, computed as:

$$\text{PnL}_i^* = \tanh(k \cdot \hat{x}_i) \cdot x_i$$

where $\hat{x}_i$ is the predicted return and x_i is the actual return. The use of `tanh` ensures the output is bounded and differentiable.

- MSE is the mean squared error between predicted and true returns:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i - x_i)^2$$

- SR^* is a *differentiable approximation* of the Sharpe Ratio, defined as:

$$\text{SR}^* = \frac{E[\text{PnL}^*]}{\text{Std}(\text{PnL}^*)} = \frac{\frac{1}{n}\sum_{i=1}^{n}\text{PnL}_i^*}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\text{PnL}_i^* - \bar{\text{PnL}}^*\right)^2}}$$

- STD is the standard deviation of generated PnLs, included to penalize volatility and encourage more stable returns.

The coefficients $\alpha, \beta, \gamma, \delta$ control the weight of each loss term and are selected using a *gradient norm matching* procedure. This technique helps balance the contribution of each term during backpropagation, avoiding dominance by any single component.

Rather than always using the full loss formulation, I tested multiple loss combinations by setting different subsets of the weights to non-zero values. For example: BCE + PnL only ($\alpha > 0$, others $= 0$), BCE + MSE + Sharpe ($\beta, \gamma > 0$), BCE + PnL + STD ($\alpha, \delta > 0$), etc. Each configuration is trained separately, and the best-performing model (based on Sharpe Ratio on validation data) is selected.

The GAN is trained using the RMSProp optimizer[1], an adaptive gradient descent method that adjusts the learning rate for each parameter based on a moving average of recent gradient magnitudes, with a learning rate of 0.0001, batch size of 100, and 20 epochs per loss combination. Probabilistic forecasts are generated via Monte Carlo sampling, where 1000 different noise vectors are used to generate a distribution of next-step returns for each input window.

2.3 Long Short-Term Memory (LSTM)

As a baseline deep learning model, a standard Long Short-Term Memory (LSTM) network for time series regression is implemented. LSTMs are a type of recurrent neural network (RNN) specifically designed to capture temporal patterns and long-range dependencies in sequential data, making them well suited for modeling financial time series.

In the chosen setup, the LSTM is trained using mean squared error (MSE) loss to minimize prediction error of the return at time $t + 1$, based on a window of past returns.

A detailed explanation of the LSTM cell architecture, including its internal gating mechanism and update equations, is provided in **Appendix A**.

[1] Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

2.4 ARIMA Baseline

As a benchmark, I compare the implemented models against a classical econometric approach: the AutoRegressive Integrated Moving Average (ARIMA) model. We apply a fixed configuration, ARIMA($p = 1, d = 0, q = 0$), which corresponds to a first-order autoregressive model without differencing or moving average components. The model is fit on the most recent 1000 observations from the training and validation set combined, and evaluated on the test set. Performance is measured using RMSE, PnL, and Sharpe Ratio, consistent with the evaluation metrics used for the deep learning models. A formal definition of the ARIMA model and its equation is provided in **Appendix B**.

2.5 Data Description

This study uses daily stock ETF-excess and daily raw ETF returns from S&P 500 stocks and sector ETFs, based on open and close prices from 22-03-2010 to 14-03-2025. This time window extends beyond the one used in the Fin-GAN paper (Jan 2000–Dec 2021). This allows for an updated performance assessment across more recent and volatile market conditions (including the COVID-19 pandemic and the Russia–Ukraine war).

All time series share a consistent training-validation-test split of 80-10-10, applied chronologically without shuffling, following best practices in financial time series to avoid information leakage. This split mirrors the setup used in Fin-GAN, facilitating comparison.

Returns Computation: To match Fin-GAN's setup, returns are computed sequentially as alternating open-to-close and close-to-open returns. Each unit of time corresponds to one such return. A sliding window approach is used: the condition window spans 10 time units (i.e., five trading days), and the model predicts one time unit ahead. Returns are capped at $\pm 15\%$ to reduce the impact of outliers.

Each dataset (train/validation/test) is represented as a matrix with 11 columns—10 for input features ($x_1, \ldots, x_{10}$) and 1 for the target (x_{11}). A sliding window of one time unit is used to generate overlapping sequences for model input.

Data Cleaning and Sector Selection: Stocks or ETFs with missing values were removed to ensure data consistency across time. The final dataset contains 385 stocks and 9 sector ETFs. To keep the experiment computationally feasible, a subset of sectors was selected based on:

- Number of tickers — to ensure sufficient data for training,
- Volatility of returns — as higher volatility often improves model learnability,
- Economic relevance — based on qualitative assessment of importance during 2010–2025.

A representation of the selection criteria are summarized in Table 1 for the chosen sectors.

Table 1: Sector Selection

Sector	Volatility	Tickers	Relevance	Justification
Information Technology	0.0215	55	1	Strong growth across AI, software, semiconductors
Consumer Discretionary	0.0210	43	1	Pandemic-era shifts; rise of digital retail
Financials	0.0177	66	1	Fintech growth, post-2008 recovery
Health Care	0.0182	54	1	Biotech, pharma innovation; pandemic role
Energy	0.0246	17	1	Oil shocks; renewable transition in focus

The tickers, along with their sector memberships, are listed in **Appendix C**.

3 Results

This section presents an analysis of model performance across the implemented approaches, comparing classical and deep learning methods for return forecasting and trading signal generation. The evaluation is structured around three above mentioned key metrics. Quantitative tables and statistical tests are included to validate the significance of observed differences.

3.1 Evaluation Framework

3.1.1 Trading strategy

To translate model forecasts into actionable investment decisions, I adopt the binary classification rule proposed in the Fin-GAN framework. Given a predicted next-period excess return $\hat{r}_{t+1}$, the following position is taken:

$$\text{Position}_{t+1} = \begin{cases} +1, & \text{if } \hat{r}_{t+1} > 0 \\ -1, & \text{otherwise} \end{cases}$$

If the predicted return is *positive* a long position is taken; if *negative*, a short position is initiated. This binary classification transforms the regression output into a tradable signal. It directly aligns the model's objective with directional profitability.

The realized trading performance is computed as:

$$\text{PnL}_{t+1} = \text{Position}_{t+1} \cdot r_{t+1}$$

where r_{t+1} is the actual excess return. This transformation allows return forecasts to be evaluated in terms of trading profitability, rather than predictive accuracy alone, which is in line with the chosen evaluation metrics.

The following performance measures are employed:

- **Root Mean Squared Error (RMSE)** — Measures predictive accuracy:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(\hat{r}_t - r_t)^2}$$

- **Sharpe Ratio (SR)** — Captures risk-adjusted performance:

$$\text{SR} = \frac{E[R]}{\sigma(R)}$$

- **Profit and Loss (PnL)** — Measures average daily returns from directional trades.

3.1.2 Model Comparison

In total five model configurations are implemented: a baseline ARIMA(1,0,0), two LSTM models (base and tuned), and two GAN variants (base and tuned). Table 2 reports the average performance across five sectors.

Table 2: Average performance across all sectors

Model	**Avg. PnL**	**Avg. Sharpe**	**Avg. RMSE**
ARIMA	0.0843	0.0484	0.0115
LSTM (base)	0.3790	0.0354	0.0110
LSTM (tuned)	2.0583	0.2297	0.0109
GAN (base)	0.1681	0.1090	0.0109
GAN (tuned)	0.5245	0.2110	0.0110

From the table we can see that: Fine-tuning significantly improves GAN performance in terms of RMSE, and directionally improves Sharpe Ratio; while LSTM tuning yields strong financial returns (PnL). In general, ARIMA underperforms across all metrics, validating the need for nonlinear, data-driven models.

3.1.3 Fine-Tuning Strategy and Implementation

Fine-tuning was applied to both LSTM and GAN models to investigate their performance and potentially improve financial outcomes.

In line with the FinGAN implementation, for GANs, a gradient-based hyperparameter search was performed using a diagnostic routine that balances generator–discriminator training

stability. This involves modifying loss weights (α, β, γ, δ) to emphasize financial objectives and ensure that GANs are not overfitting to BCE loss. Moreover, as described above, custom loss combination of the metrics were used in the training.

In addition, I performed a manual grid-free fine-tuning of key model hyperparameters, fixed in the original implementation, that involved:

- Increasing hidden layer sizes (hid_g, hid_d), allowing a larger latent space that better supports GANs model return dynamics (in line with the FinGAN's approach).
- Extending training duration from 20 to 30 epochs to allow the generator to converge properly.
- Adjusting learning rates for the generator and discriminator, namely increasing the generator's learning rates help GANs learn richer features without being overpowered by the Discriminator.

LSTMs were trained with fixed architecture in the base case, and fine-tuned with longer training and modified loss functions in the tuned version.

This approach is directly motivated by the Fin-GAN methodology, prioritizes financial interpretability and reward maximization over traditional accuracy-focused objectives. The paper also highlights that GANs can offer superior performance when appropriately optimized for financial metrics (like Sharpe Ratio, PnL) rather than just predictive accuracy (RMSE), which the fine-tunes setup supports.

3.1.4 Statistical Validation

To evaluate whether performance differences between base and tuned models are statistically significant, I performed paired t-tests on Sharpe and RMSE values across all sectors. Table 3 summarizes the results.

Table 3: Paired t-test results: tuned vs. base models

Metric	**Model**	**Sectors with $p < 0.05$**	**Directional Gains**
Sharpe Ratio	LSTM	0/5	4/5
Sharpe Ratio	GAN	0/5	5/5
RMSE	LSTM	0/5	3/5
RMSE	GAN	**4/5**	5/5

These results indicate that fine-tuning GANs leads to consistent and statistically significant reductions in forecast error (RMSE). While improvements in Sharpe Ratio are not statistically significant, they are consistently positive, particularly for GANs. This suggests that longer evaluation windows or ensemble-based strategies may be needed to fully capture risk-adjusted gains.

4 Discussion

4.1 Sector-Specific Performance Insights

From a sectoral perspective, model performance revealed important distinctions. Figures 2, 3, and 4 show sector-level performance for each metric and model architecture. Bars are color-coded by model type, with numerical values annotated to aid comparison. [2]

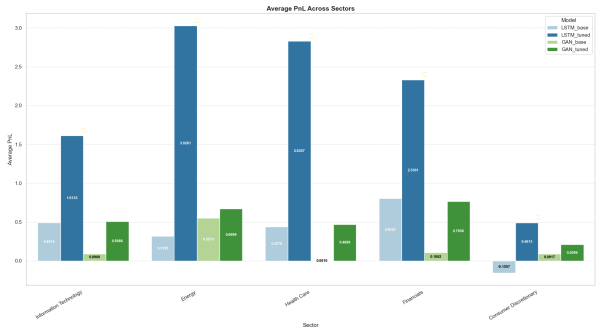

Figure 2: Average PnL across sectors

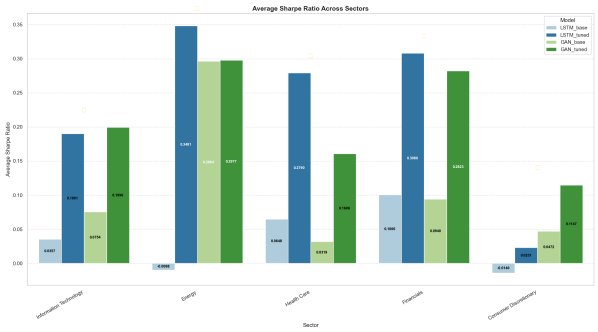

Figure 3: Average SR across sectors

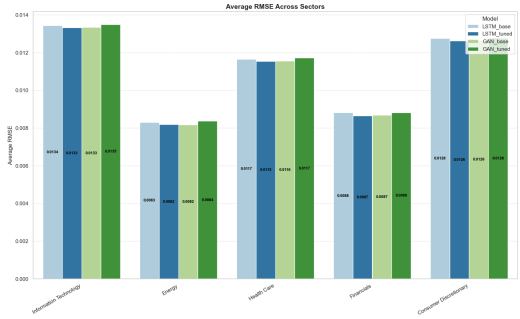

Figure 4: Average RMSE across sectors

The plots show that tuned LSTMs consistently outperformed other models in relatively stable sectors such as **Financials**, **Health Care**, and **Energy**, likely due to their sequential nature and stability under conservative dynamics. On the other hand, fine-tuned GANs (GAN_tuned) performed best in sectors like **Information Technology** and **Consumer Discretionary**, characterized by higher volatility and growth orientation, where capturing distributional nuances and asymmetries may be critical. This aligns with the ability of GANs to model complex, non-Gaussian distributions and capture rare but impactful return patterns. These results confirm how important is to account for underlying properties of each financial environment.

[2]The discussion centers on a critical interpretation of the results obtained only from the main models implemented.

In terms of **evaluation metrics**, Sharpe Ratio and PnL proved more discriminative and financially meaningful than RMSE. While RMSE differences were marginal across models, Sharpe and PnL revealed substantial improvements from tuning, especially for GANs. This confirms the limitations of using prediction error as the sole objective in financial settings, where directional accuracy and risk-adjusted returns are ultimately more relevant.

Notably, statistical significance tests using paired t-tests revealed that, while most improvements from fine-tuning were not statistically significant at the 5% level, as we can see from the plots, the direction of improvement was consistent across nearly all sectors for Sharpe and PnL. This suggests that, although the improvements may be small in magnitude or noisy, the fine-tuned models—particularly GANs—consistently move in the right direction, especially in high-volatility sectors. This directional consistency is valuable in finance, where marginal improvements can compound over time.

In general, while this study implements a simplified version of the Fin-GAN architecture, the obtained results are consistent with the original findings. In particular, it confirms that incorporating domain-specific, economics-driven objectives into the training process leads to improved financial performance.

However, several **limitations** remain. First, GANs are notoriously challenging to train and require significant computational resources, particularly when tuned per sector and per objective on a large amount of data. Second, their black-box nature makes interpretability more complex, which may hinder adoption in high-stakes or regulated environments. Finally, while the fine-tuning improved performance, it was still performed manually - future implementations could benefit from automated or adaptive loss reweighting to optimize stability and relevance.

Future work could explore cross-ticker or cross-sector GAN training, as well as integrating hybrid LSTM-GAN models to improve robustness. Additionally, incorporating explainability tools - such as SHAP (SHapley Additive exPlanations) values or gradient-based saliency maps- could help with the mentioned harder interpretability of GANs.

References

[1] Marin Vuletić, Franziska Prenzel, and M. Cucuringu. "Fin-GAN: forecasting and classifying financial time series via generative adversarial networks". In: *Quantitative Finance* 24.2 (2024), pp. 175–199. DOI: 10.1080/14697688.2023.2299466.

[2] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in neural information processing systems* 27 (2014). URL: https://arxiv.org/abs/1406.2661.

[3] A. Koochali et al. "Probabilistic Forecasting of Sensory Data With Generative Adversarial Networks–ForGAN". In: *IEEE Access* 7 (2019), pp. 63868–63880.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[5] R. S. Tsay. *Analysis of Financial Time Series*. Hoboken, NJ: John Wiley & Sons, 2005.

[6] Simon Buchner and Christopher Krauss. "Fin-GAN: A Conditional GAN with Sharpe Ratio Optimization for Financial Time Series Forecasting". In: *arXiv preprint arXiv:2201.08948* (2022).

[7] Adriano Koshiyama, Nargiz Firoozye, and Philip Treleaven. "Generative adversarial networks for financial trading strategies fine-tuning and combination". In: *Quantitative Finance* 21 (2020), pp. 1–17.

[8] Thanakorn Leangarun, Pisut Tangamchit, and Sathaporn Thajchayapong. "Stock Price Manipulation Detection using Generative Adversarial Networks". In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2018, pp. 2104–2111. DOI: 10.1109/SSCI.2018.8628755.

[9] Shu Takahashi, Yifei Chen, and Kumiko Tanaka-Ishii. "Modeling financial time-series with generative adversarial networks". In: *Physica A: Statistical Mechanics and its Applications* 527 (2019), p. 121261. DOI: 10.1016/j.physa.2019.121261.

[10] Xiaoyan Zhou et al. "Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets". In: *Mathematical Problems in Engineering* 2018 (2018), pp. 1–11. DOI: 10.1155/2018/6143435.

Appendices

A LSTM Architecture

The LSTM maintains internal hidden and cell states which are updated at each time step via gate mechanisms that control the flow of information. The cell is defined by the following equations:

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) && \text{(forget gate)} \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) && \text{(input gate)} \\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) && \text{(candidate state)} \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) && \text{(output gate)} \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

Here, σ denotes the sigmoid activation function, tanh is the hyperbolic tangent, and $\odot$ represents element-wise multiplication.

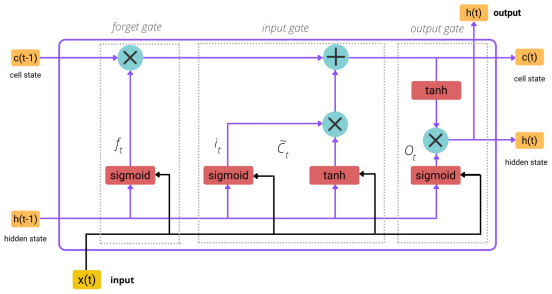

Figure 5: LSTM cell architecture

B ARIMA Model Definition

The ARIMA(p, d, q) model combines three components: autoregression (AR), integration (I), and moving average (MA). In our setup, we use a simplified form with parameters $p = 1$, $d = 0$, and $q = 0$, reducing the model to:

$$y_t = \phi_1 y_{t-1} + \varepsilon_t$$

Where:

- $p = 1$: autoregressive order,

- $d = 0$: no differencing, as returns are already stationary,
- $q = 0$: no moving average component,
- ε_t: white noise error term.

The model is fit using maximum likelihood estimation and selected for its balance of interpretability and computational efficiency.

C Ticker–Sector Mapping

Table 4: Sector, ETF ticker, and constituent stock tickers used in the selected experiments.

Sector	**ETF Ticker**	**Stock Tickers**
Consumer Discretionary	XLY	ABNB, AMZN, APTV, AZO, BBY, BKNG, BWA, CCL, CMG, DG, DHI, DLTR, EBAY, ETSY, EXPE, F, GM, HAS, HD, HLT, IPG, KMX, LEN, LKQ, LULU, MCD, MGM, MOH, NCLH, NKE, NVR, ORLY, PENN, PHM, POOL, RCL, RL, ROST, SBUX, TGT, TJX, TPR, TSLA, UA, UAA, VFC, WHR, WYNN, YUM
Energy	XLE	APA, BKR, COP, CTRA, CVX, DVN, EOG, EQT, FANG, HAL, HES, MRO, MPC, OKE, OXY, PXD, SLB, VLO, WMB, XOM
Financials	XLF	ACGL, AFL, AIG, AIZ, AJG, ALL, AMP, AON, APO, APTV, BAC, BEN, BK, BLK, BRK.B, BRO, CB, CBOE, CEG, CFG, CINF, CMA, COF, DFS, FDS, FITB, FRC, GL, GS, HBAN, ICE, INVH, JPM, KEY, L, LNC, MCO, MET, MKTX, MMC, MS, MSCI, NDAQ, NTRS, PFG, PGR, PNC, PRU, RE, RF, RJF, SCHW, SIVB, SPGI, SRE, SYF, TFC, TROW, TRV, USB, V, VNO, VTRS, WFC, WRB, ZION
Health Care	XLV	A, ABBV, ABT, ALGN, AMGN, BAX, BDX, BIIB, BMY, CAH, CNC, COO, COR, CVS, DHR, DXCM, ELV, GILD, HCA, HOLX, HSIC, HUM, IDXX, ILMN, INCY, IQV, ISRG, JNJ, LH, LLY, MDT, MHK, MRK, MTD, PFE, PKI, REGN, RMD, STE, SYK, TFX, TMO, UHS, UNH, VRTX, WAT, ZBH, ZTS
Information Technology	XLK	AAPL, ACN, ADBE, ADI, ADSK, AKAM, AMAT, AMD, ANSS, APH, AVGO, CDNS, CDW, CHKP, CRM, CRWD, CTSH, DXC, ENPH, EPAM, FFIV, FIS, FISV, FLT, FTNT, GPN, HPQ, IBM, INTC, INTU, JKHY, KLAC, LDOS, LRCX, MCHP, MPWR, MSFT, MSI, MU, NET, NFLX, NOW, NVDA, ON, ORCL, PANW, PAYC, PAYX, QCOM, SNPS, STX, SWKS, TDY, TER, TRMB, TYL, TXN, VRSN, WDC, WU, ZBRA, ZS