Using Classical Time Series Methods and Diffusion Models for Forecasting Stock Prices: A Comparative Analysis

Dev Tripathy [1] **Nikolai Shchelkov** [1]

Abstract

In this project, we aim to investigate the use of Denoising Diffusion Probabilistic Models (DDPMs) for forecasting stock prices of several major companies. We will train diffusion-based time series models on historical stock data for companies covering technology, finance, energy, industrial, and consumer sectors. The performance of the diffusion approach will be compared against classical statistical forecasting (ARIMA). Evaluation will be based on accuracy metrics such as Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). This proposal outlines the motivation, research questions, methodology, related work, and anticipated challenges of the project.

1. Introduction

Forecasting financial time series remains a critical yet challenging task due to their inherent volatility, non-linearity, and non-stationary behavior. Classical approaches such as ARIMA have long been used for their interpretability and statistical rigor. However, their linear assumptions limit performance in scenarios involving structural breaks, regime shifts, or non-linear dependencies (Adebiyi et al., 2014). In contrast, Denoising Diffusion Probabilistic Models (Ho et al., 2020) represent a modern, generative class of models capable of learning complex data distributions and producing diverse, probabilistic forecasts. These models offer the potential to address limitations of point-estimate models by capturing full distributions over future outcomes. Applying such models to financial data could enable richer uncertainty quantification and improved predictive robustness.

[*]Equal contribution [1]Department of Information Systems, University of Maryland, Baltimore County, Baltimore, USA. Correspondence to: Dev Tripathy <devjyot1@umbc.edu>, Nikolai Shchelkov <yo78454@umbc.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

2. Research Questions

This project investigates the following key research questions.

1. Can DDPMs outperform ARIMA models in forecasting accuracy for stock prices across different sectors?
2. Does model performance vary significantly by industry or firm-specific volatility profiles?
3. What are the trade-offs between diffusion models and ARIMA in terms of interpretability, computational efficiency, and robustness to market volatility?

3. Methodology

We plan to use historical daily stock prices for five major publicly traded companies, each representing a different economic sector. The dataset will be obtained from open financial data sources such as Yahoo Finance. The selected companies – Microsoft, JPMorgan, Exxon Mobil, Caterpillar, and Procter & Gamble – ensure diversity in sector behavior and volatility profiles. Data will be divided into training and test subsets, with the most recent observations reserved for evaluating forecast performance.

The Denoising Diffusion Probabilistic Model will be trained to predict future values in the stock time series, conditioned on a fixed-length historical window. The model will learn to reverse a stochastic noise-adding process, effectively generating possible future trajectories from a learned prior. To provide a benchmark, we will implement ARIMA models tailored to each stock, selecting optimal parameters based on training data diagnostics.

Forecast accuracy will be measured using RMSE and MAE. For DDPMs, we will also explore distributional metrics and assess probabilistic forecast intervals. A comparative analysis will highlight the strengths and weaknesses of each approach, both in predictive performance and practical deployment.

4. Related Work

ARIMA has long been a cornerstone of financial time series forecasting. Despite its limitations, it remains widely used due to its simplicity and interoperability (Adebiyi et al., 2014). However, recent studies have emphasized that while ARIMA models may perform adequately in short-term forecasting, they struggle with non-linearities and interactions that are common in real-world financial data (Falat & Stanikova, 2015; Zhang & Wen, 2022).

Diffusion models, particularly DDPMs introduced by Ho et al. (2020), have achieved state-of-the-art performance in image generation and are now being adapted for sequential modeling tasks. The TimeGrad model by Rasul et al. (2021) represents one of the earliest applications of diffusion models to time series, showing promise in multivariate probabilistic forecasting. More recently, DiffSTOCK by Daiya et al. (2024) extended DDPMs to financial forecasting, demonstrating that these models can better capture uncertainty and complex dependencies in stock market data. Moreover, recent works have emphasized that diffusion models outperform ARIMA in capturing the non-linearity and high variance patterns present in financial markets (Cheung et al., 2023; Tsoku & Metsileng, 2024).

While these works suggest that diffusion models are capable of modeling sequential financial data, they also highlight challenges such as high variance in performance and sensitivity to hyperparameters. Our work builds on this foundation by applying DDPMs to single-stock prediction and benchmarking them against more established methods.

5. Using ARIMA for Stock Prices Forecasting

5.1. Data Collection

We collected daily historical closing prices for five publicly traded companies – Microsoft (MSFT), JPMorgan Chase (JPM), ExxonMobil (XOM), Caterpillar (CAT), and Procter & Gamble (PG) – representing technology, finance, energy, industrial, and consumer goods sectors, respectively. The data was sourced from Stooq – a free financial data platform offering a wide range of market information – and covers a five-year period from May 1, 2020, to April 30, 2025. To ensure consistency across time series, we removed any rows with missing values, retaining only dates with complete data for all tickers.

5.2. Exploratory Data Analysis

Figure 1 presents the historical closing prices for the selected companies.

While all companies exhibit overall price growth over the five-year span, the magnitude and internal structure of these trends vary significantly. For example, Microsoft experienced a notable decline in stock value between 2022 and 2023, driven by a deceleration in its Azure cloud segment and broader economic headwinds (Rogers, 2023). This variability highlights the need for forecasting models that can capture both global and localized dynamics in stock price behavior.

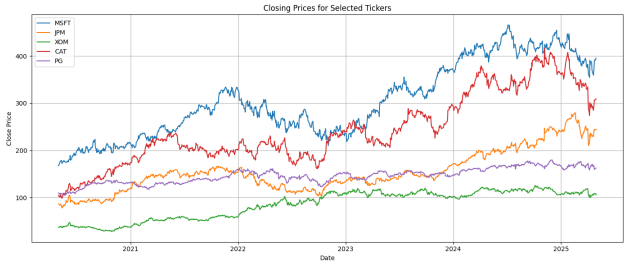


Figure 1. Historical Closing Prices.

5.3. Model Estimation

We employed the auto_arima function from the Python pmdarima library to identify optimal ARIMA model parameters for each stock. The dataset was partitioned into training and test sets using an 80:20 ratio. The resulting model parameters, including the number of autoregressive (AR), differencing (I), and moving average (MA) terms, along with the Akaike Information Criterion (AIC) values, are summarized in Table 1.

Table 1. ARIMA Model Parameters for Each Ticker

TICKER	AR	I	MA	AIC
MSFT	2	1	3	5912.31
JPM	0	1	0	4354.89
XOM	0	1	0	3662.76
CAT	0	1	0	5609.89
PG	1	1	0	3672.33

The ARIMA models selected for each stock reflect differences in their price dynamics and predictability. Microsoft (MSFT) is modeled with ARIMA(2,1,3), indicating a more complex structure involving both autoregressive and moving average components after differencing the series once. This suggests that MSFT's price movements depend on past values and past errors, possibly due to market memory or volatility. Procter & Gamble (PG), with ARIMA(1,1,0), shows a mild autoregressive structure, meaning its immediate past value helps improve predictions, but the model remains simple.

JPMorgan (JPM), ExxonMobil (XOM), and Caterpillar (CAT) are modeled with ARIMA(0,1,0), which typically corresponds to a pure random walk. However, in these

cases the models include a nonzero intercept (drift), meaning they follow a random walk with drift. The resulted predictions are illustrated on Figure 2.

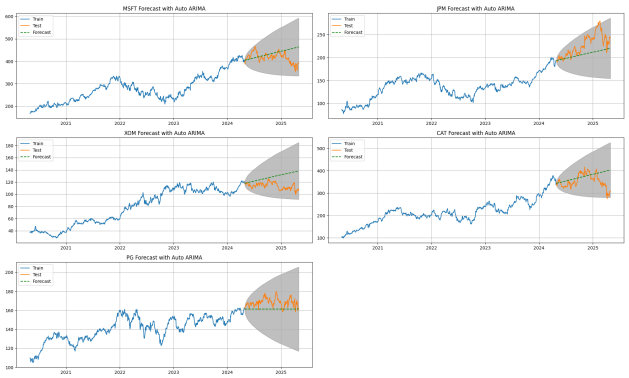


Figure 2. Stock Close Prices Predictions using ARIMA Models.

Visual inspection reveals that although ARIMA models generally captured the overall trend in stock price movements, significant deviations are observed, particularly within local intervals. Therefore, to draw definitive conclusions about model performance, it is essential to evaluate forecast accuracy using quantitative error metrics.

5.4. Evaluation Metrics

To evaluate model performance, we computed both the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) on the test set. These metrics are presented in Table 2.

Table 2. Forecast Accuracy of ARIMA Models

TICKER	MAE	RMSE
MSFT	27.64	37.60
JPM	19.98	26.04
XOM	14.36	17.12
CAT	29.18	39.61
PG	7.12	8.13

The lowest prediction errors were observed for Procter & Gamble and ExxonMobil, possibly due to lower volatility and better alignment between historical and test-period trends. In contrast, Microsoft's more volatile price behavior led to higher forecast errors, despite using a more complex model. These findings indicate that while ARIMA models can capture general trends effectively, their performance deteriorates when local non-linearities or regime shifts dominate.

Overall, the mixed results underscore the limitations of ARIMA models in handling complex financial time series, thus motivating the need for more flexible and robust forecasting approaches, such as diffusion-based generative models.

6. Diffusion Models Forecasting

We now describe the design of the diffusion-based forecasting model, specifically a multivariate Denoising Diffusion Probabilistic Model (DDPM) for time series data.

The model is built around a U-Net 1D architecture embedded in a diffusion training pipeline. The input stock prices, scaled to zero mean and unit variance. The key innovation lies in using a time-indexed stochastic noise process to corrupt the target series and learning to denoise it via conditional modeling. For a detailed mathematical description of the model, refer to appendix A

The architecture includes:

- An LSTM encoder that captures temporal context from the input sequence and maps it to a fixed-size latent vector.
- A linear layer that projects this latent into the same dimension as the input features.
- A U-Net-based decoder (1D convolutional architecture) that denoises a noisy future trajectory given the current context.
- A time embedding module that conditions the model on diffusion step $t \in [0, T]$.

The model is trained using a standard mean squared error (MSE) loss between the added noise and the predicted noise. During training, the forward process samples a timestep t, corrupts the target using a noise schedule β_t, and the model learns to reconstruct the original clean signal. At inference time, the model generates samples from pure noise through iterative denoising.

To obtain forecast intervals, we generate multiple sample trajectories from the learned model and use quantiles to form prediction intervals. For a list of model parameters used , refer to appendix D

Although it has been shown that state-of-the-art performance can be achieved using DDPMs for time series forecasting, the model being implemented here is a much simpler version of those architectures. The reasons are twofold: first one being such complicated architectures are somewhat out of scope for this course project and their implementation would involve a deeper understanding of modern learning techniques. The second one serves the purpose of determining to what extend can a simple diffusion model by itself predict noisy non-stationary complex patterns as observed in the financial markets.

7. Results

The forecasts from the diffusion model described are presented in Figure 3. A quick visual inspection clearly shows that the Multivariate Diffusion model architecture performs poorly on most datasets except for ExxonMobil where it outperforms ARIMA model and the forecast for Procter & Gamble is also decent. This is shown quantitatively in Table 3 in terms of RMSE and MAE.

Table 3. Forecast Error Differences: Diffusion Model vs. ARIMA

TICKER	DIFF_MAE	DIFF_RMSE
MSFT	171.67	172.98
JPM	90.48	93.35
XOM	11.19	12.28
CAT	111.35	115.14
PG	22.42	22.82

Table 4. Diagnostics Summary for Diffusion Model Performance

TICKER	RMSE_DIFF	VOLATILITY	SNR	NL	AC(LAG1)
MSFT	119.54	0.0162	105.70	0.0041	0.9980
JPM	58.81	0.0158	94.49	0.0036	0.9982
XOM	-3.73	0.0182	142.22	0.0027	0.9987
CAT	82.13	0.0181	87.60	0.0039	0.9981
PG	15.06	0.0103	40.73	0.0102	0.9949

While the diffusion model prediction has richer and complex features better mimicking real time-series data, its values are nowhere close to the actual prices. It is worthwhile to investigate why the model has exceptional performance for ExxonMobil. To explain this, further diagnostics such as Volatility, Signal-to-Noise Ratio (SNR), Non-Linearity (NL), AutoCorrelation (AC) were computed for all the datasets and are shown in Table 4. For a more detailed explanation of the metrics used in these diagnostics, please refer to appendix B. It is found that XOM has the highest Signal-to-Noise Ratio (SNR) which implies there is a strong signal to be learned and that combined with some mild non-linearity becomes an effective condition for the diffusion model to learn. Note that in the absence of any non-linearity, ARIMA would clearly outperform a linear SNR time series. On the other hand, diffusion models are designed to capture non-linear generative processes. If the series has low volatility meaning non-linear patterns are more stable rather than having large erratic jumps (high volatility), the model can learn the structure well. This is seen in the case of PG which has the highest non-linearity (NL) and our multivariate diffusion model performs quite decently. On the other hand for CAT, its high volatility masks the non-linear patterns present in the data and makes it difficult for the model to learn.

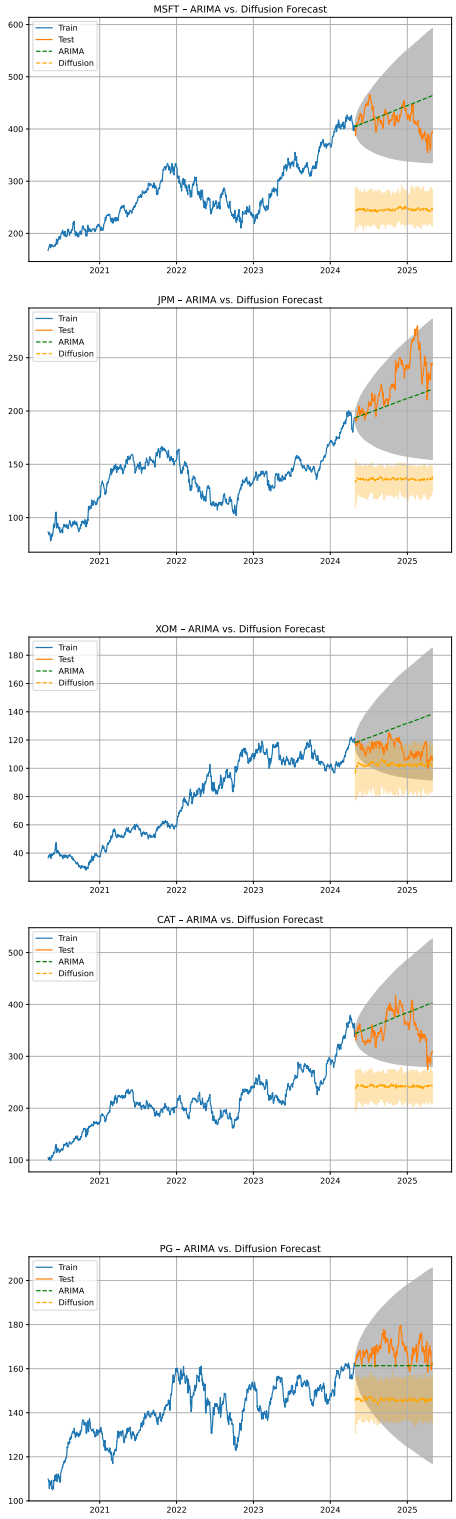


Figure 3. Multivariate Diffusion Forecasts for Stock Closing prices

Algorithm 1 Forecasting using ARIMA and Multivariate Diffusion Model

Input: List of tickers $\mathcal{T}$, time window $[t_0, t_1]$, context length C, forecast horizon H, number of diffusion samples N
Download historical closing prices $\mathcal{D}_t$ for each ticker $t \in \mathcal{T}$ from Stooq between $[t_0, t_1]$
Split $\mathcal{D}_t$ into training and test sets with ratio $(1-\rho) : \rho$

for each ticker $t \in \mathcal{T}$ **do**
  Fit ARIMA model to training data using auto_arima()
  Forecast H steps ahead with confidence intervals at 95%
  Store predicted mean $\hat{y}_t^{\text{ARIMA}}$, and confidence bounds $[\hat{y}_t^L, \hat{y}_t^U]$
end for

Normalize training data using StandardScaler
Construct time series dataset with sliding windows of length C and horizon H
Define and initialize the DiffusionUNetLSTM model

for epoch = 1 to 100 **do**
  for each batch (x, y) from the training set **do**
    Sample time step t_i and generate noisy input $\tilde{x}_t$
    Predict noise using UNet conditioned on LSTM embedding and t_i
    Compute MSE loss between predicted and true noise
    Update model weights using Adam optimizer
  end for
end for

Use last C time steps of training data to form context tensor
Generate N probabilistic forecast samples of length H using the trained model
Compute median forecast $\hat{y}_t^{\text{MV}}$ and 95% CI bounds using empirical percentiles
Invert normalization for interpretability

for each ticker $t \in \mathcal{T}$ **do**
  Plot historical, ARIMA, and MV forecasts with confidence intervals
  Compute MAE and RMSE for both models
end for

8. Discussions and Outlook

The analysis presented in this study is an initial step towards identifying optimal forecasting models for stock market applications. We demonstrated and compared the ARIMA and a Multivariate Diffusion model. The main take-aways from our analysis are as follows:

- ARIMA despite being a simpler model is able to capture the overall trends of the Stock Close prices however due to the nature of its simplicity, it is unable to capture complex noisy patterns present in the data
- The Multivariate Diffusion model performs poorly for most stock prices but thrives under two ideal conditions
  - It can denoise and reconstruct accurately when there's non-linear data with low volatility.
  - It learns realistic probabilistic transitions when the signal dominates over noise with mild non-linearity.

This, however, does not imply that Diffusion models cannot be used for time-series forecasting. Our analysis simply shows the kind of datasets that this simple diffusion model can predict. Future work would include improving upon our current architecture based on recent research work in literature. The directions to future work could include:

- Our current model generates the entire forecast window all at once from a fixed context and there is no temporal feedback. As an immediate next step, one could include auto-regressive sampling via recursive sampling updates. In such a setup, the diffusion model generates one step at a time, feeding each generated value back into the LSTM. For a preliminary result, refer to the appendix C.
- Further improvements would include replacing the current UNet with a WaveNet-style residual architecture for the noise added at each timestep, better modeling of long-range dependencies and improving training stability and expressivity.
- Allowing the network to learn predictive uncertainity per timestep improving probabilistic calibration and enabling fine-grained confidence intervals in contrast, to the current model which assumes unit variance in sampling.
- Expanding the dataset to include a larger number of stock tickers over longer time horizons, to better capture diverse company characteristics across sizes and sectors.
- Investigating causal relationships to improve forecasts based on the underlying drivers of market movements. A deeper analysis of company strategies and key business decisions could enhance model adaptability and interpretability.

In conclusion, forecasting stock prices remains a rich and evolving research domain at the intersection of corporate finance and data analytics.

References

Adebiyi, A. A., Adewumi, A. O., and Ayo, C. K. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th Int. Conf. on Computer Modelling and Simulation*, pp. 106–112. IEEE, 2014. doi: 10.1109/UKSim.2014.67.

Cheung, L., Wang, Y., Lau, A. S. M., and Chan, R. M. C. Using a novel clustered 3d-cnn model for improving crop future price prediction. *Knowledge-Based Systems*, 2023. URL https://www.sciencedirect.com/science/article/pii/S0950705122012291.

Daiya, D., Yadav, M., and Rao, H. S. Diffstock: Probabilistic relational stock market predictions using diffusion models. In *ICASSP 2024*, 2024.

Falat, L. and Stanikova, Z. Application of neural network models in modelling economic time series with non-constant volatility. *Procedia Economics and Finance*, 34:194–201, 2015. doi: 10.1016/S2212-5671(15)01620-8.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *Proceedings of ICML 2021*, pp. 8857–8868, 2021.

Rogers, O. Reports of cloud decline have been greatly exaggerated. *Uptime Institute Journal*, 2023. URL https://journal.uptimeinstitute.com/reports-of-cloud-decline-have-been-greatly-exaggerated/.

Tsoku, J. T. and Metsileng, D. Hybrid time series and ann-based elm model on jse/ftse closing stock prices. *Frontiers in Applied Mathematics and Statistics*, 2024. URL https://www.frontiersin.org/articles/10.3389/fams.2024.1454595/full.

Zhang, F. and Wen, N. Carbon price forecasting: a novel deep learning approach. *Environmental Science and Pollution Research*, 2022. URL https://link.springer.com/article/10.1007/s11356-022-19713-x.

A. Appendix: Mathematical Derivation of Multivariate Diffusion Forecasting Model

1. Problem Setup and Notation

Let $\{\mathbf{x}_t\}_{t=1}^{T}$ be a d-dimensional multivariate time series, with $\mathbf{x}_t \in \mathbb{R}^d$. For training, we extract:

$$\mathbf{X} = [\mathbf{x}_t, \ldots, \mathbf{x}_{t+L-1}] \in \mathbb{R}^{L\times d}, \quad \mathbf{Y} = [\mathbf{x}_{t+L}, \ldots, \mathbf{x}_{t+L+H-1}] \in \mathbb{R}^{H\times d}$$

where L is the context window length and H is the forecast horizon.

2. Forward Process (Diffusion)

Let $x_0 \in \mathbb{R}^{H\times D}$ denote the original future target window. The forward process adds Gaussian noise progressively:

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Here, $\beta_t \in (0,1)$ determines the noise variance at step t, and we define:

$$\alpha_t = 1-\beta_t, \quad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

Thus, any noisy state can be sampled directly from the clean input as:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

3. Reverse Process (Generative)

The reverse process learns to remove the noise using a neural network. The true reverse distribution is:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

In our setting, we simplify by having the model directly predict the noise:

$$\epsilon_\theta(x_t, t, c) \approx \epsilon$$

and train using the score matching loss:

$$\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{x_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(x_t, t, c)\|^2\right]$$

4. Conditioning on Historical Context

To forecast conditioned on past data, we include a context window $c \in \mathbb{R}^{L\times D}$. An LSTM encodes c into a hidden representation $h(c) \in \mathbb{R}^D$, and time embeddings $g(t) \in \mathbb{R}^D$ are computed from normalized timestep t/T. The denoising input becomes:

$$\text{cond}_t = x_t + h(c) + g(t)$$

5. Sampling Procedure

At inference time, we start from Gaussian noise:

$$x_T \sim \mathcal{N}(0, I)$$

and sample backward for $t = T, T-1, \ldots, 1$ as:

$$x_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t, c) \right) + \sqrt{\beta_t} z,$$

with $z \sim \mathcal{N}(0, I)$ for $t > 1$ and $z = 0$ for $t = 1$.

This produces probabilistic samples of future trajectories consistent with the context.

6. U-Net Architecture and Its Role in Diffusion Forecasting

U-Net is a convolutional neural network originally introduced for biomedical segmentation and now commonly used in diffusion models due to its ability to model local and global structure.

General Architecture:

- **Encoder (downsampling):** Sequential convolutions and pooling compress the input, extracting coarse features.
- **Bottleneck:** A latent representation bridging encoder and decoder.
- **Decoder (upsampling):** Transposed convolutions reconstruct signal resolution.
- **Skip connections:** Preserve detail by connecting encoder and decoder layers.

Implementation Details in Our Model:

- The U-Net is implemented using 1D convolutional layers to process temporal sequences.
- Input shape: (B, D, H), where B is batch size, D the number of features (stocks), and H the forecast horizon.
- **Conv1:** Applies a 1D convolution with kernel size 3, input channels D, and output channels 64.
- **Conv2:** Applies another 1D convolution with kernel size 3 and stride 2, increasing channels to 128 and reducing temporal resolution.
- **Upsampling:** Uses a transposed 1D convolution (kernel size 2, stride 2) to restore temporal resolution.
- **Conv3:** Concatenates upsampled output with earlier encoder feature map and applies a convolution to integrate.
- **Output:** A final 1D convolution reduces channels back to D, yielding a noise estimate $\hat{\epsilon} \in \mathbb{R}^{B \times H \times D}$.

The U-Net denoises x_t effectively by leveraging both fine-grained and coarse temporal features, yielding accurate recovery of x_0. Its convolutional nature makes it efficient and suitable for multivariate time series with limited data.

B. Appendix: Definitions of Diagnostic Metrics

Volatility: Measures the average magnitude of percentage changes over a rolling window, capturing the local variability of returns. Computed as:

$$\text{Volatility} = \frac{1}{N-w+1} \sum_{t=w}^{N} \text{StdDev} \left(\frac{P_{t-w+1:t}}{P_{t-w+1:t-1}} - 1 \right)$$

where P_t is the price at time t and w is the window size (typically 20).

SNR (Signal-to-Noise Ratio): Ratio of the variance of a smoothed (rolling mean) version of the time series to the variance of the residual (noise). Defined as:

$$\text{SNR} = \frac{\text{Var}(\text{RollingMean}(P_t))}{\text{Var}(P_t - \text{RollingMean}(P_t))}$$

TS (Trend Strength): Quantifies the dominance of the trend component relative to the residual noise in a seasonal decomposition:

$$\text{TrendStrength} = 1 - \frac{\text{Var}(\text{Residual})}{\text{Var}(\text{Trend})}$$

Derived using additive seasonal decomposition (e.g., via STL or classical decomposition).

NL (Nonlinearity): Measures the proportion of unexplained variance from a linear one-step-ahead autoregressive model:

$$\text{Nonlinearity} = \frac{\text{Var}(y_{t+1} - \hat{y}_{t+1}^{\text{linear}})}{\text{Var}(y_{t+1})}$$

where $\hat{y}_{t+1}^{\text{linear}}$ is predicted using simple linear regression on y_t.

AC(Lag1): First-order autocorrelation, reflecting how much today's value is linearly dependent on yesterday's:

$$\text{AC}(1) = \frac{\sum_{t=2}^{N} (y_t - \bar{y})(y_{t-1} - \bar{y})}{\sum_{t=1}^{N} (y_t - \bar{y})^2}$$

where $\bar{y}$ is the mean of the series.

C. Appendix: Autoregressive Diffusion Model

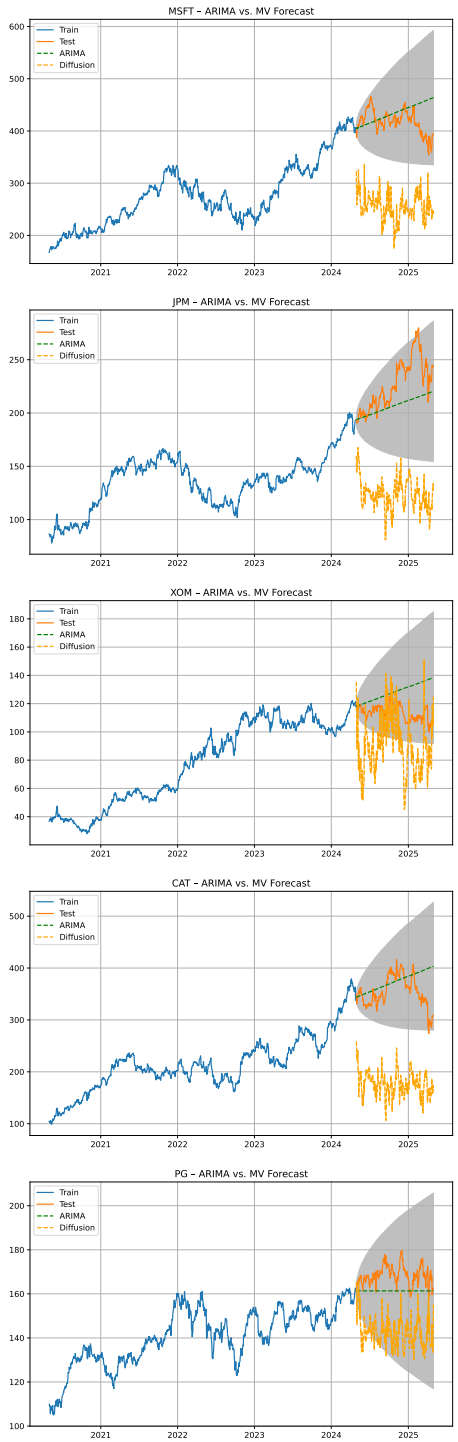
Algorithm 2 Velocity-based Diffusion Forecasting with LSTM Conditioning

Require: Multivariate time series $X \in \mathbb{R}^{T \times D}$, context length C, forecast horizon H, total diffusion steps T_d
Ensure: N sampled trajectories of future stock prices

Training Phase:
for each training sequence $x_0 \in \mathbb{R}^{T \times D}$ **do**
  Compute velocity: $v_t = x_t - x_{t-1}$
  Concatenate: $x_{\text{aug}} = \text{concat}(x_0, v)$
  Encode temporal context: $h = \text{LSTM}(x_{\text{aug}})$
  for $t = 1\ T-1$ **do**
    Sample timestep $t_i \sim \mathcal{U}\{1, T_d\}$
    Add noise: $\tilde{x}_t = \sqrt{\bar{\alpha}_{t_i}} x_t + \sqrt{1-\bar{\alpha}_{t_i}}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$
    Compute condition: $\text{cond}_t = \tilde{x}_t + h2c(h_t) + t2c(t_i)$
    Predict noise: $\hat{\epsilon}_t = \text{UNet}(\text{cond}_t)$
    Accumulate loss: $\mathcal{L} += \|\epsilon - \hat{\epsilon}_t\|^2$
  end for
end for

Sampling Phase (Inference):
Initialize context $x_{1:C}$ from most recent C observations
Encode: $h, c = \text{LSTM}(x_{\text{aug}})$
for $t = 1\ H$ **do**
  Initialize: $x_t \sim \mathcal{N}(0, I)$
  for $i = T_d\ 1$ **do**
    Compute condition: $\text{cond} = x_t + h2c(h_{\text{last}}) + t2c(i)$
    Predict noise: $\hat{\epsilon} = \text{UNet}(\text{cond})$
    Denoise: $x_t \leftarrow \frac{1}{\sqrt{1-\beta_i}}\left(x_t - \frac{\beta_i}{\sqrt{1-\bar{\alpha}_i}}\hat{\epsilon}\right)$
    if $i > 1$ **then**
      Add noise: $x_t \leftarrow x_t + \sqrt{\beta_i} \cdot \mathcal{N}(0, I)$
    end if
  end for
  Append x_t to generated sequence
  Compute velocity: $v_t = x_t - x_{t-1}$
  Update LSTM state with $\text{concat}(x_t, v_t)$
end for Forecasted sequence of H timesteps

The forecasts generated from this algorithm are presented here however their purpose is only to show how to better model real time stock prices rather than just a simple trend. The Diffusion model still predicts the stock prices poorly and so future work would involve making this model even better.



Autoregressive Diffusion Forecasts for Stock Closing Prices

D. Appendix: Diffusion Model Parameters

This section summarizes the key parameters used in the multivariate diffusion forecasting model, including training setup, architecture specifications, and sampling configuration.

Symbol / Variable	Value / Type	Description
T	300	Total number of diffusion timesteps
L	60	Context window length (number of past time points)
H	variable (≈ 250)	Forecast horizon (number of test days)
d	5	Number of features (stocks: MSFT, JPM, XOM, CAT, PG)
n_{samples}	100	Number of samples during reverse diffusion
`ctx_len`	60	Length of context window
`horiz`	test set length	Length of forecasting window
`train_size`	80% of data	Training split proportion
`test_ratio`	0.2	Fraction of data reserved for testing
`batch_size`	32	Batch size for training
`epochs`	100	Total number of training epochs
`learning_rate`	10^{-3}	Learning rate for Adam optimizer
`UNet1D`	ConvNet	U-Net backbone for denoising (1 encoder + 1 decoder layer)
`Conv1`	Conv1D(d, 64, kernel=3, pad=1)	First temporal convolution
`Conv2`	Conv1D(64, 128, kernel=3, stride=2)	Downsampling step
`UpConv`	ConvTranspose1D(128, 64, kernel=2, stride=2)	Upsampling step
`Conv3`	Conv1D(128, 64, kernel=3, pad=1)	Decoder convolution
`ConvOut`	Conv1D(64, d, kernel=1)	Final projection to feature space
`LSTM`	LSTM(d, 128, num_layers=2)	Context encoder for historical input
`h2c`	Linear(128, d)	Maps LSTM hidden state to conditioning vector
`t2c`	Linear(1, d)	Maps normalized timestep t/T to embedding
`scaler`	StandardScaler	Standardizes inputs before training
β_t	linspace$(10^{-4}, 0.02, T)$	Linear noise schedule
α_t	$1 - \beta_t$	Complement of diffusion coefficient
$\bar{\alpha}_t$	$\prod_{s=1}^{t} \alpha_s$	Cumulative product of alphas

Table 5. Model hyperparameters and architectural components used in the diffusion forecasting model.