Modeling Technology Stock Prices with Long Short-Term Memory Neural Networks

Hongli Zhao*
honglizhaobob@uchicago.edu

Abstract

In this note, we discuss and compare machine learning methods for financial time series data using 18 selected members of Nasdaq 100.

1 Raw Dataset

We consider the following stocks listed on Nasdaq, summarized in Table 1. The price data frequency is daily and from April 30, 2015 to April 30, 2024, comprising of 18 stocks over 9 years and totaling 2265 records. The specific choice of date range is due to missing records for BABA prior to the year 2015. The dataset is loaded using Python library `yfinance` [2]. The loaded dataset was verified such that no missing records were present, and the dates match for all stock price time series.

Ticker	Company Name	Market Cap (Q1 2024)
MSFT	Microsoft Corporation	\$3.13T
AAPL	Apple Inc.	\$2.65T
NVDA	NVIDIA Corporation	\$2.13T
AMZN	Amazon.com, Inc.	\$1.88T
GOOG	Alphabet Inc. (Google)	\$1.88T
META	Meta Platforms, Inc.	\$1.24T
TSM	Taiwan Semiconductor Manufacturing Company Limited	\$627.98B
AVGO	Broadcom Inc.	\$602.57B
TSLA	Tesla, Inc.	\$559.85B
TCEHY	Tencent Holdings Limited	\$372.28B
ORCL	Oracle Corporation	\$322.10B
CRM	Salesforce, Inc.	\$261.14B
ADBE	Adobe Inc.	\$199.25B
CSCO	Cisco Systems, Inc.	\$190.23B
QCOM	Qualcomm Incorporated	\$189.28B
INTC	Intel Corporation	\$188.03B
BABA	Alibaba Group Holding Limited	\$179.51B
IBM	International Business Machines Corporation	\$175.06B

Table 1: List of Stock Tickers, Company Names, and Market Cap as of Q1 2024, sorted by Market Cap

The loaded data contains 6 columns: Open, Close, High, Low, Adjusted Close, Volume, as shown in Figure 1. We let P_t denote the adjusted close price on recorded day t ($0 \leq t \leq 2264$), where $t = 0$ refers to April 30, 2015. We note that the recorded dates are irregularly spaced due to the presence of holidays and weekends. For discussion of general methodology, we assume for simplicity that the

*Example Collaborators

Example Affiliation 2024, Example Location, USA

price time series is evenly spaced, and defer data correction methods in the later sections. In general, such discrepancy may have an non-negligible effect in understanding the daily returns and should not be ignored [1].

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-04-30	82.220001	82.650002	80.440002	81.290001	78.505547	14209700
1	2015-05-01	81.029999	82.250000	80.110001	81.169998	78.389656	17910500
2	2015-05-04	81.169998	81.559998	80.120003	80.589996	77.829521	21551500
3	2015-05-05	80.440002	80.900002	77.769997	79.540001	76.815483	32661900
4	2015-05-06	79.650002	80.849998	79.510002	80.000000	77.259735	24953500

Figure 1: Example dataframe for BABA. The data for other stock tickers follow the same format, containing columns "Date", "Open", "High", "Low", "Close", "Adj Close", and "Volume".

While for short periods, log returns and simple returns are both appropriate choices, we consider log returns in order to incorporate continuous compounding and avoid potential numerical underflows over potentially longer periods [10]. We let:

$$R_t := \log\left(\frac{P_t}{P_{t-1}}\right), \quad 1 \leq t \leq T \tag{1}$$

where T denotes the ending day of recorded data.

2 Exploratory Data Analysis

In this section, we compute and examine relevant statistics among the stock price time series, and summarize our observations.

2.1 Daily price changes

We first visualize the daily price changes $\Delta P_t := P_t - P_{t-1}$, plotted in Figure 2. We generally observe that the price changes are clustered around 0, with mean ranging from -0.0016 and 0.55, and standard deviation ranging from 0.61 and 10.01. In particular, the daily returns of TSLA, META, AVGO, ADBE suggest a high degree of variability.

2.2 Stationarity tests

To gain more qualitative insight to the returns data, we normalize them by considering the log returns as defined in (1), and apply both augmented Dickey-Fuller (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests. We briefly describe the setup of each hypothesis test below.

ADF test: The ADF test checks the null hypothesis:

$$H_0^{(i)} := \text{ the log return data of stock } i \text{ is characterized by a unit root process} \tag{2}$$

A linear regression is performed on the returns data with the differences computed with up to p-th lagged series:

$$\Delta R_t \sim \alpha + \beta t + \gamma R_{t-1} + \sum_{k=1}^{p-1} \delta_k \Delta R_{t-k} + \epsilon_t \tag{3}$$

where $\Delta R_{t-k} := R_t - R_{t-k-1}$. The maximum lag order p is determined using the Akaike information criterion (AIC) for each stock. The regression coefficient $\widehat{\gamma}$ is taken as the final test statistic to be checked against a 5% critical value suggested in [7]. In Table 2, we report the test statistic, 5% critical value, and final test result for each stock.

KPSS test: The KPSS test is another statistical test used to check for stationarity in a time series. Unlike other tests that have a null hypothesis of non-stationarity, the KPSS test has a null hypothesis

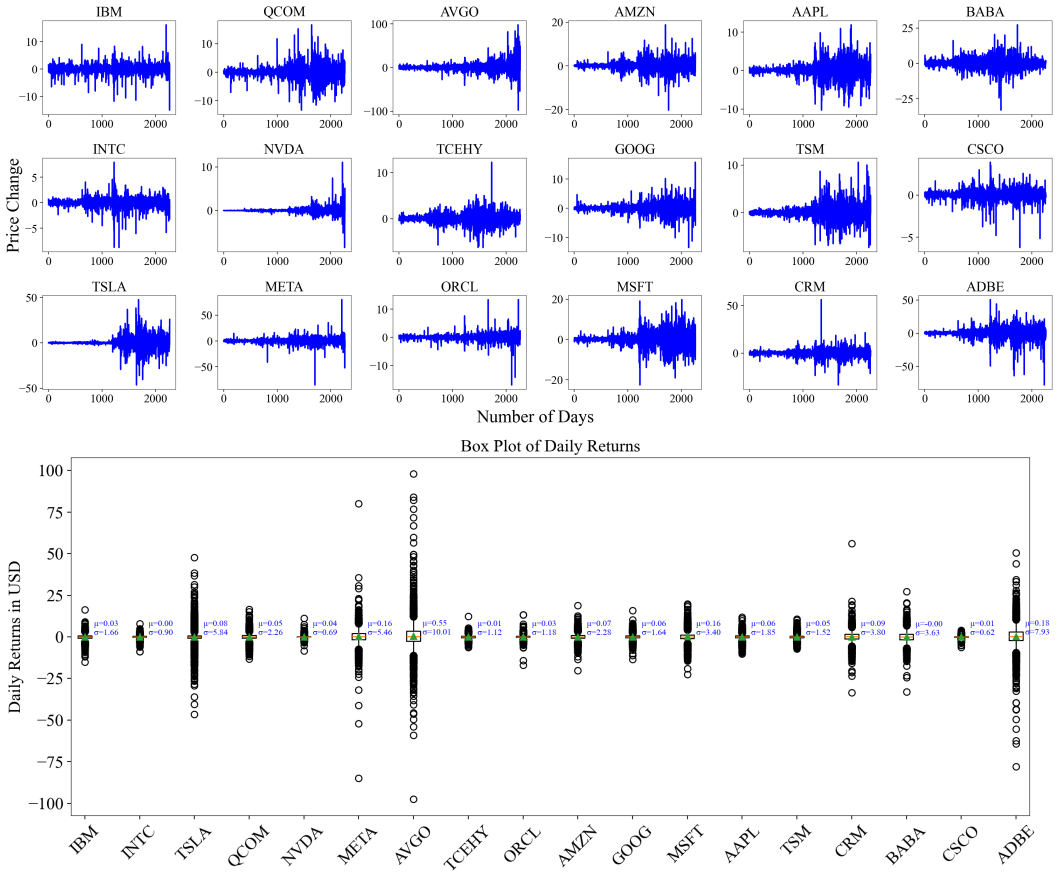

Figure 2: (Top) Visualization of price changes for all investigated stocks. (Bottom) Box plots of price changes for all investigated stocks, with mean and standard deviation annotated.

that the series is stationary. The KPSS test is based on the decomposition of a time series y_t into a deterministic trend μ_t, a random walk r_t, and a stationary error ϵ_t:

$$y_t = \mu_t + r_t + \epsilon_t \tag{4}$$

where $r_t = r_{t-1} + u_t$ and u_t is white noise. The test statistic is based on the residuals obtained from the regression of y_t on μ_t:

$$\widehat{\epsilon_t} = y_t - \widehat{\mu}_t \tag{5}$$

The test statistic is computed as:

$$\text{KPSS} = \frac{1}{T^2} \sum_{t=1}^{T} S_t^2 \tag{6}$$

where $S_t = \sum_{i=1}^{t} \widehat{\epsilon_i}$ is the partial sum of the residuals. This statistic is compared to a 5% critical value to determine whether the null hypothesis of stationarity can be rejected. A higher value of the test statistic indicates a higher likelihood that the series is non-stationary. Table 3 suggests that we cannot reject the hypothesis that the return series is stationary.

2.3 Daily return distributions

While the daily price changes provide preliminary information about the risk profiles of investing in each stock, the standard deviation of price changes incorporates the scale information of each stock (e.g. investing in AVGO may not be inherently more risky than TCHEY, if appropriate position sizing

Ticker	$\widehat{\gamma}$	**5% Critical Value**	H_0
IBM	-10.0361	-2.8628	Reject
INTC	-9.5570		Reject
TSLA	-9.4196		Reject
QCOM	-10.1425		Reject
NVDA	-8.3858		Reject
META	-7.8446		Reject
AVGO	-9.9342		Reject
TCEHY	-8.9981		Reject
ORCL	-10.2929		Reject
AMZN	-9.0711		Reject
GOOG	-9.8161		Reject
MSFT	-10.4754		Reject
AAPL	-8.7442		Reject
TSM	-9.4352		Reject
CRM	-9.1279		Reject
BABA	-9.2986		Reject
CSCO	-9.7470		Reject
ADBE	-9.6017		Reject

Table 2: Results of ADF test for all stock tickers. ADF tests suggest that the return series are not unit roots.

Ticker	KPSS	**5% Critical Value**	H_0
IBM	0.1049	0.463	Fails to Reject
INTC	0.1480		Fails to Reject
TSLA	0.1555		Fails to Reject
QCOM	0.0794		Fails to Reject
NVDA	0.1039		Fails to Reject
META	0.0898		Fails to Reject
AVGO	0.0765		Fails to Reject
TCEHY	0.1888		Fails to Reject
ORCL	0.0843		Fails to Reject
AMZN	0.1845		Fails to Reject
GOOG	0.0430		Fails to Reject
MSFT	0.0538		Fails to Reject
AAPL	0.1001		Fails to Reject
TSM	0.0662		Fails to Reject
CRM	0.0497		Fails to Reject
BABA	0.2955		Fails to Reject
CSCO	0.0763		Fails to Reject
ADBE	0.2219		Fails to Reject

Table 3: Results of KPSS test for all stock tickers, favoring the conclusion that the return series are stationary in time.

was in place). By the results of Section 2.2, we assume that the stock returns are stationary, and therefore their statistics will persist throughout the period of analysis. In Figure 3, we first visualize the return series in histograms for each stock. In particular, we compute a Gaussian maximum likelihood fit to the log returns data. While the data is concentrated around zero and unimodal, in general, we observe that a Gaussian distribution fails to capture its kurtic information. The data for AVGO, BABA, TCEHY, ORCL and META also appear more skewed than a normal distribution. In the Q-Q plot, we find there are general upper- and lower-tail deviations from the expected behavior of a normal distribution. The general "S-shape" of the Q-Q plots indicates the presence of extreme events for investment returns in the investigated technology stocks.

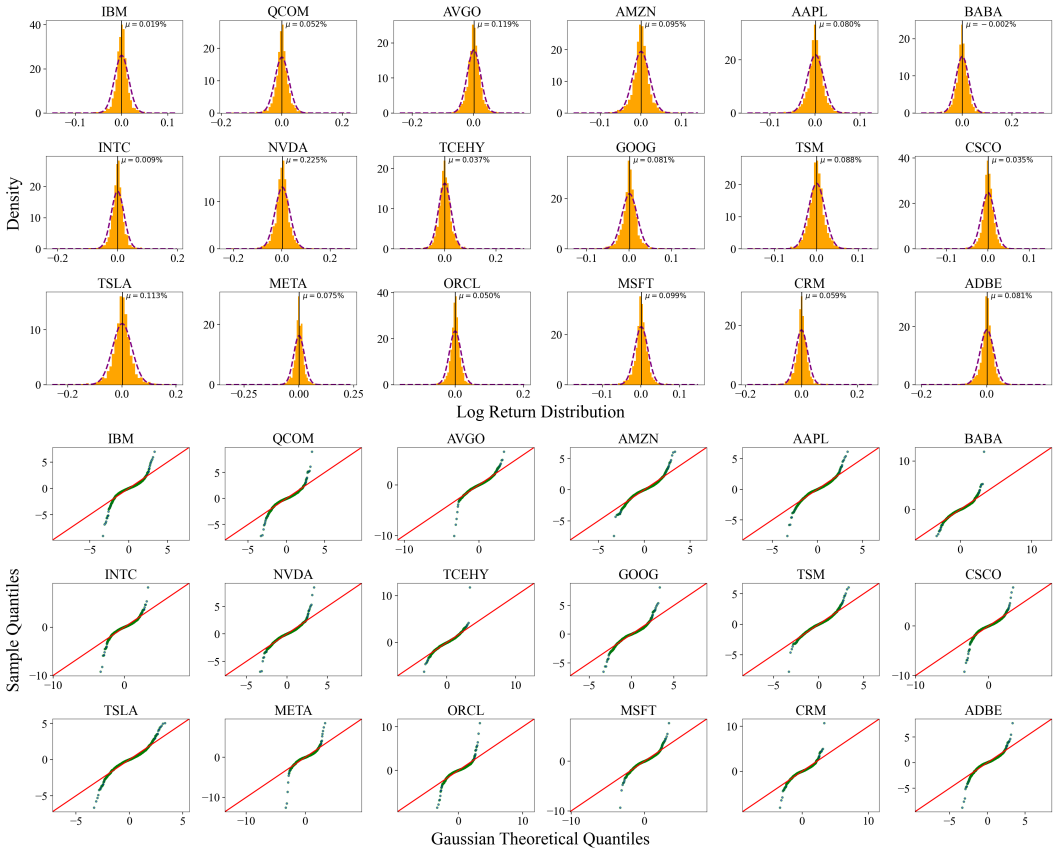

Figure 3: (Top) Distribution of log returns, the purple dash line indicates a Gaussian fit to the data. Furthermore, the mean daily returns are annotated around the mode. (Down) Q-Q plot comparison of log returns data against Gaussian distribution quantiles.

To be more precise about this empirical observation, we conduct a Kolmogorov-Smirnov (KS) test for normality. The empirical cumulative distribution function (CDF) of sample log returns data is compared against that of a Gaussian distribution with the same sample mean and variance. Then, the KS test statistic can be computed and compared with critical values at both tails of the Kolmogorov distribution [8]. In our study, we set the significance level at 0.05 with null hypothesis:

$$H_0^{(i)} := \text{ the log returns of stock } i \text{ follows a Gaussian distribution} \tag{7}$$

All KS test results suggested the rejection of H_0.

Finally, we consider higher order central moments for the log returns data of each stock, we compute the unbiased Fisher-Perron coefficient (sample skewness) and Pearson kurtosis estimators with a sample size of $T = 2264$. The estimated results for each stock are compared to a Gaussian skewness of 0 and kurtosis of 3. The results are visualized in Figure 5. As expected from previous observations, the log return data of stocks under consideration generally have excess kurtosis (with TSM, TSLA and AMZN having the lowest kurtosis; and META, TCHEHY, CRM having the highest). Similarly, the returns of QCOM, AMZN, and GOOG show the least skewness; while META, CRM, and CSCO demonstrate a high degree of skew. In terms of parameter inference for probabilistic modeling, the log returns are better modeled with an assymetric Laplace distribution. In Figure 4, we present and compare the fitted Gaussian distribution and assymmetric Laplace distributions for META for an example illustration. The parameters are found through maximum likelihood estimation (MLE), and reported in Table 4. Due to random initializations, the MLE fitting process was repeated for 10 different random seeds, and reported as an average.

	Fit 1			Fit 2	
	$\hat{\kappa}$	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
Value	1.025	0.001	0.016	0.001	0.025

Table 4: META log returns: fitted MLE parameters of assymmetric Laplace distribution and Gaussian distribution, averaged across 10 random seeds. In the Laplace distribution, κ is a parameter that controls skewness and kurtosis.

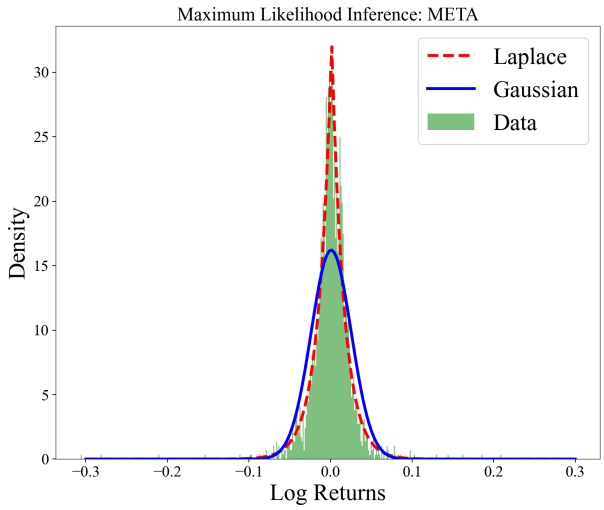

Figure 4: Illustration of Laplace distribution fit for META log returns, compared with Gaussian fit.

2.4 Moving average

In this section, we return to the original price series P_t, and visualize moving average smoothers for the observed data over $T = 2265$ days. The moving averages are linear combinations of lagged differences, and may be used as features for time series forecasting models, explored in later sections. More precisely, a general smoothed time series with window size $k \geq 0$ can be defined as the

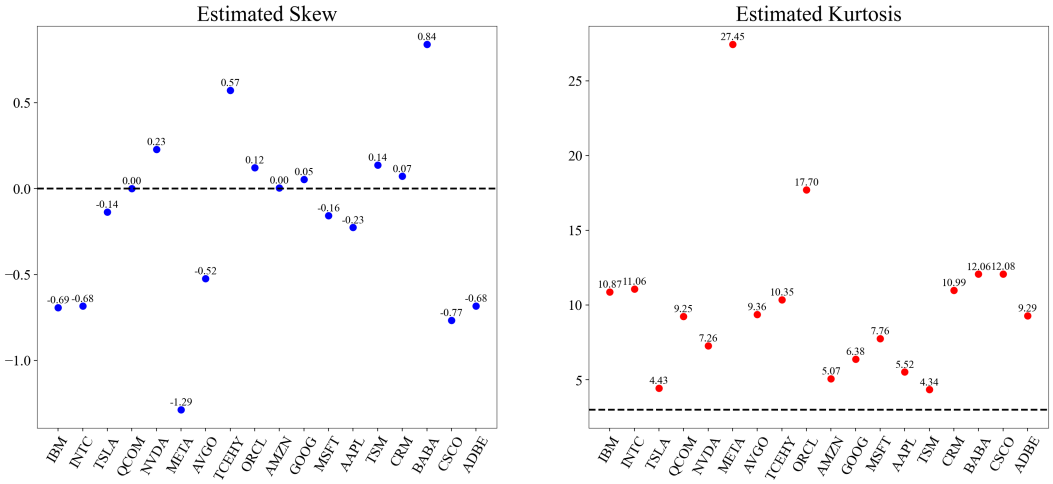

Figure 5: Unbiased estimator of skewness and kurtosis computed from $T = 2264$ log return observations, for each stock. The Gaussian benchmark skewness and kurtosis are represented as dash lines for comparison.

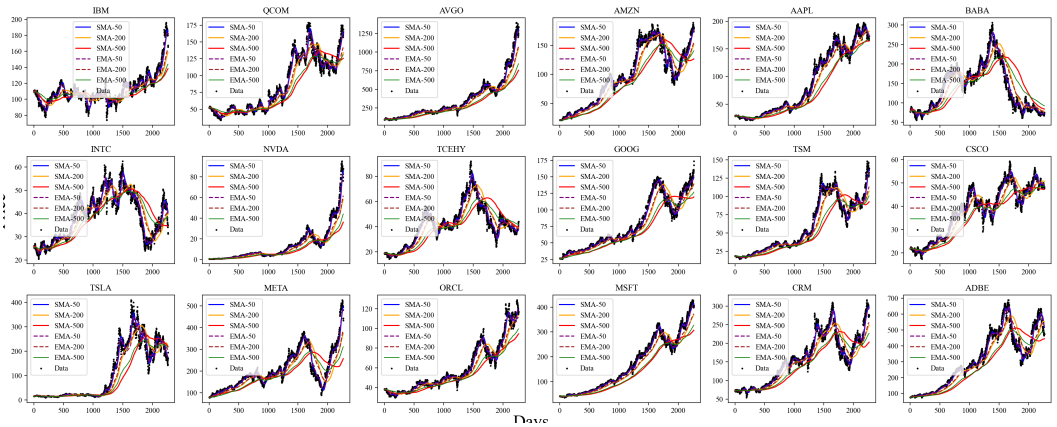

Figure 6: Example moving averages: (1) simple moving average (SMA): SMA-50, SMA-200, SMA-500. (2) exponential moving average (EMA): EMA-50, EMA-200, EMA-500, with $\tau = 1$.

following:

$$M_t := \sum_{j=-k}^{k} a_j P_{t-j} \tag{8}$$

where $\sum_{j=-k}^{k} a_j = 1$, and $a_j = a_{-j} \geq 0$. When used for model predictions, an assymetric moving average is used to avoid look-ahead bias. Furthermore, the weights are computed using a weight function depending on other asymmetric kernels:

$$M_t := \sum_{j=-k}^{0} a_j(t) P_{t-j} \tag{9}$$

where:

$$a_i(t) := \frac{K((t-i)/\tau)}{\sum_{j=-k}^{0} K((t-j)/\tau)} \tag{10}$$

with $K(\cdot)$ being a kernel function; $\tau > 0$ is a pre-specified scaling parameter.

In our illustration, we consider the kernel functions for (1) simple moving average, K_{sim} and (2) exponential moving average K_{exp}, defined as the following:

$$K_{\text{sim}}(z) \equiv 1, \quad K_{\text{exp}}(z) := e^{-z/\tau} \tag{11}$$

For an illustration, we test simple moving averages with lookback window sizes $k = 10, 20, 50$, as well as an exponential moving average with $k = 10, 20, 50$ and $\tau = 1$. The results are shown in Figure 6. Moving averages create lagged time series, and have been noted as features in time series forecasting models [11], which we explore in Section 3.

2.5 Return correlations

For the purpose of constructing a portfolio of stocks with dampened risk profile, it is often helpful to understand the immediate (in time) effect of the returns of one stock on that of another. In Section 2.2, we have verified that the stock return series are stationary. As a result, we expect the correlation among stocks to maintain across our entire simulation period (i.e. time invariant). The covariance of stock i with stock j at time t is defined as:

$$C_{ij}(t) := \mathbb{E}[(R_t^{(i)} - \mu_t^{(i)})(R_t^{(j)} - \mu_t^{(j)})] \tag{12}$$

where $\mu_t^{(k)}$ denotes the (theoretical) mean of the time series at time t. By the stationarity assumption, the mean and variance can be estimated from the average of observed returns over the simulation period. The correlation is then computed as:

$$\widehat{\rho}_{ij}(t) := \frac{\widehat{C}_{ij}(t)}{\widehat{\sigma}_i(t) \cdot \widehat{\sigma}_j(t)} \tag{13}$$

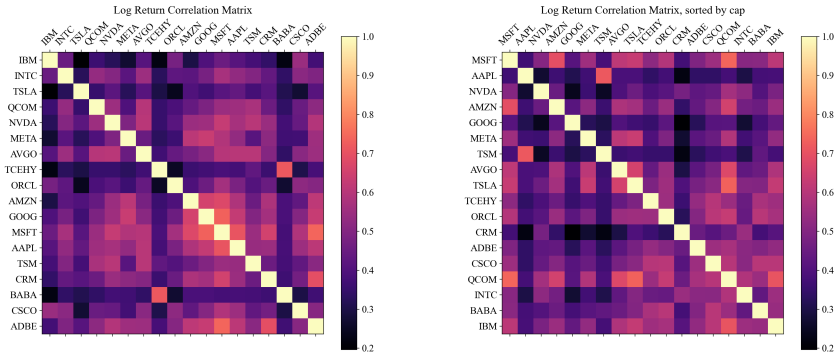

Figure 7: (Left) Heatmap of sample correlation among the log returns of stocks in Table 1, unsorted. (Right) Sorted correlation matrix ordered by market capitalization.

where $\widehat{\sigma}_k(t)$ denotes the estimated standard deviation. The computed sample correlations among stock returns are visualized as a heatmap in Figure 7.

2.6 Risk measures

2.6.1 Maximum and average drawdown

Maximum drawdown (MDD) is a measure of the largest single drop from peak to trough in the value of a financial time series over a specified period. For the price time series P_t, we first define the drawdown D_t:

$$D_t = \max_{0 \le \tau \le t} P_t - P_t$$

Subsequently, the maximum drawdown for a period $[0, T]$ can be defined as:

$$\text{MDD}(T) := \max_{0 \le t \le T} D_t \tag{14}$$

MDD measures the largest loss an investor could have experienced, which serves as a pessimistic measure of risk in a position or portfolio. MDD (or a series of MDDs) can be computed efficiently from historical data, and requires no additional assumptions about return distribution or simulations. We thus consider MDD as a preliminary metric for risk that places an emphasis on extreme conditions. In contrast, average drawdown (ADD) is a measure that provides insight into the typical decline an investment experiences from its peak value over a specified period. While max drawdown focuses on the single largest drop, average drawdown considers the average of all drawdowns, offering a more comprehensive view of an investment's volatility and risk profile.

In our experiment, we fix a window size $k = 252$ days, and continue to monitor the MDD and ADD metrics for each investment throughout the period under consideration. In lieu of using the entire historical trajectories, we emphasize that a fixed window size $k < T$ is chosen. The specific choice of window size assumes that we are only interested in capturing yearly performance of the stock, and extreme historical drawdowns are ignored with a memory of one year.

In Figure 8, we compare running yearly drawdown (starting from 2016), MDD, and ADD, normalized as a percentage of the yearly maximum price. As expected, MDD appears to be a much more conservative measure of risk than ADD. Upon inspecting the MDD plots, a contextual observation can be made that a general sharp decrease of investment returns is present during the approximate period from 2020 to 2022, which is covered by the COVID pandemic period [4]. A yearly buy-and-hold strategy on the stocks under consideration has an extreme event of loss ranging from -40% and -60% of 52-week high.

2.6.2 Gaussian value-at-risk (VaR) estimation

While maximum drawdown provides a quantitative measure for the level of loss an investor may experience in extreme cases, it does not quantify the likelihood of such events [6]. Value at Risk

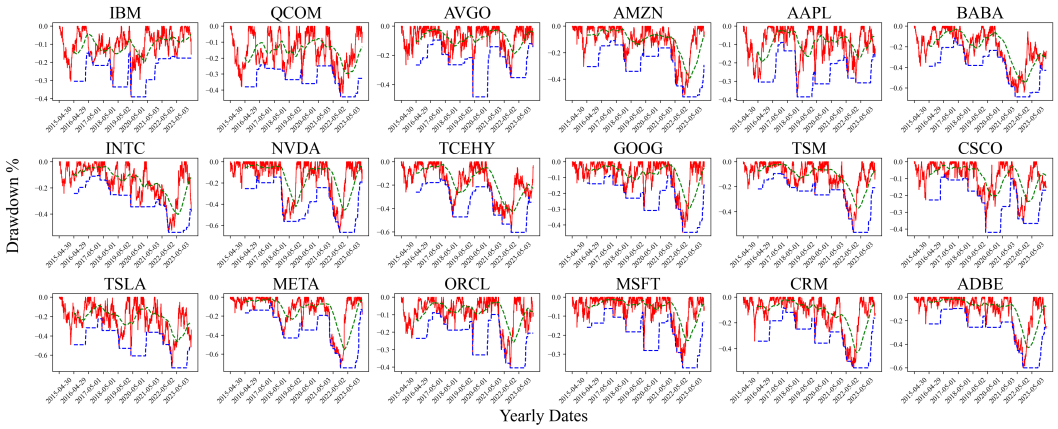

Figure 8: Yearly drawdown metrics over time. In particular, green dashed lines represent ADD, and blue lines represent MDD.

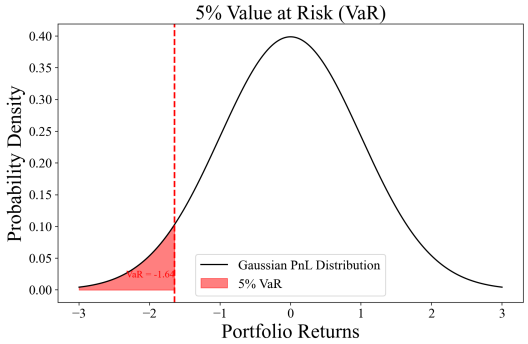

Figure 9: Illustration of a 5% value at risk using standard Gaussian distribution.

(VaR) is introduced as a probabilistic risk measure that quantifies the potential loss in value of a portfolio over a defined period for a given confidence interval. Instead of considering the process P_t directly, we define $L_t = L_t(P_0)$ as the profit or loss (PnL) after holding for t days, with an initial value of P_0. We define the t-period VaR as:

$$\text{VaR}(L_t; \alpha) := -\inf\{u : F_{L_t}(u) > \alpha\} \tag{15}$$

where F_{L_t} denotes the cumulative probability distribution function of random variable L_t; $\alpha \in [0, 1]$ is a pre-defined risk level. An intuitive interpretation of $\text{VaR}(L_t; \alpha)$ is the $(1 - \alpha)$-th percentile of $-L_t$, or absolute value of losses. For a fixed t, α, $\text{VaR}(L_t; \alpha) = c$ implies that there is a probability of α for the loss to be more than c. We also emphasize that it is not necessary for $c > 0$. If $c < 0$, it implies that the investment always has a positive return over the period considered. An illustration of VaR is presented in Figure 9.

Commonly, VaR is estimated by computing the volatility of observed log returns, then estimated using a normal distribution. The central assumption in using normal distribution lies in the stationarity of returns. More precisely, suppose the daily log return R_s $(0 \leq s \leq T)$ be distributed as $\mathcal{N}(0, \sigma^2)$, where σ does not vary in time; then starting from day 0, with an initial value of 0, for simplicity, a t-day return can be characertized as:

$$L_t = \sum_{s=0}^{t} R_s \sim \mathcal{N}(0, t\sigma^2) \tag{16}$$

In other words, if the log returns are assumed to be Gaussian with the same (stationary) variance, L_t will resemble a Brownian motion, with initial value P_0. In particular, the probability distribution of PnL after t days can be fully characterized as long as σ is estimated. In Table 5, we compute a 5%-level VaR estimated under a Gaussian assumption, for holding periods $t = 1, 2, 5, 10$.

	IBM	INTC	TSLA	QCOM	NVDA	META
1 day	-2.51%	-3.55%	-5.91%	-3.80%	-5.02%	-4.03%
2 days	-3.56%	-5.02%	-8.35%	-5.37%	-7.11%	-5.70%
5 days	-5.62%	-7.94%	-13.21%	-8.49%	-11.24%	-9.01%
10 days	-7.95%	-11.23%	-18.68%	-12.01%	-15.89%	-12.75%
$\sqrt{t}\widehat{\sigma}$	1.53%	2.16%	3.59%	2.31%	3.05%	2.45%

	AVGO	TCEHY	ORCL	AMZN	GOOG	MSFT
1 day	-3.65%	-4.02%	-2.81%	-3.38%	-2.99%	-2.84%
2 days	-5.16%	-5.68%	-3.98%	-4.79%	-4.23%	-4.01%
5 days	-8.16%	-8.99%	-6.29%	-7.57%	-6.69%	-6.34%
10 days	-11.54%	-12.71%	-8.89%	-10.70%	-9.46%	-8.97%
$\sqrt{t}\widehat{\sigma}$	2.22%	2.44%	1.71%	2.06%	1.82%	1.72%

	AAPL	TSM	CRM	BABA	CSCO	ADBE
1 day	-3.00%	-3.22%	-3.56%	-4.31%	-2.64%	-3.47%
2 days	-4.24%	-4.56%	-5.03%	-6.09%	-3.74%	-4.90%
5 days	-6.70%	-7.21%	-7.95%	-9.63%	-5.91%	-7.75%
10 days	-9.48%	-10.20%	-11.25%	-13.62%	-8.36%	-10.96%
$\sqrt{t}\widehat{\sigma}$	1.82%	1.96%	2.16%	2.62%	1.61%	2.11%

Table 5: VaR estimation under Gaussian assumption. The PnL and estimated volatility are provided as percentages (of the initial value invested at the start of a period).

For most stocks, the percentage losses increase with the length of the time frame. This is expected as the potential for greater cumulative loss grows over longer periods. Furthermore, stocks with high 1-day losses typically exhibit even higher 10-day losses, which reflects the compounding nature of risk over time. TSLA exhibits the highest percentage losses across all time frames. For example, the 1-day loss is approximately -5.91%, increasing to -18.68% over 10 days. This indicates that Tesla is highly volatile and poses significant risk over short to medium-term periods. IBM, MSFT, AAPL, and CSCO have lower percentage losses across all time frames. For instance, IBM shows a 1-day loss of -2.51% and a 10-day loss of -7.95%. These stocks can be considered less risky and exhibit more steady performance.

2.6.3 Historical Monte Carlo VaR estimation

In Section 2.6.2, we discussed a convenient way to estimate the VaR as a percentile under Gaussian assumption of returns. While the return data is stationary, a Gaussian log return is typically not the case for financial time series, as demonstrated empirically in Figure 4. For general return probability distributions, the t-day PnL, or L_t, probability distribution is typically not available in simple forms.

To account for arbitrary return distributions that may also change over time, we propose to resample the return historical data and generate Monte Carlo samples of possible PnL trajectories. Given a sufficiently large sample size, we may form an empirical probability distribution for L_t, whose percentiles can then be approximated by ranks (of simulated data). In Figure 10, we demonstrate results of bootstrap Monte Carlo samples of "would-be" realization of the NVDA prices. The empirical distributions reveal the presence of more extreme returns and that the PnL cannot be well-captured by a Gaussian distribution. In Figure 11, we applied a historical simulation approach using $N_{\text{MC}} = 10^4$ Monte Carlo simulations for each stock, and aggregating to up to 10 days of holding period by taking the 5th percentile of empirical distributions for $-L_t$. Compared to the values reported in Table **??**, in general, the historical simulation approach projected losses that are more extreme, and thus captures our expectation of a more leptokurtic return.

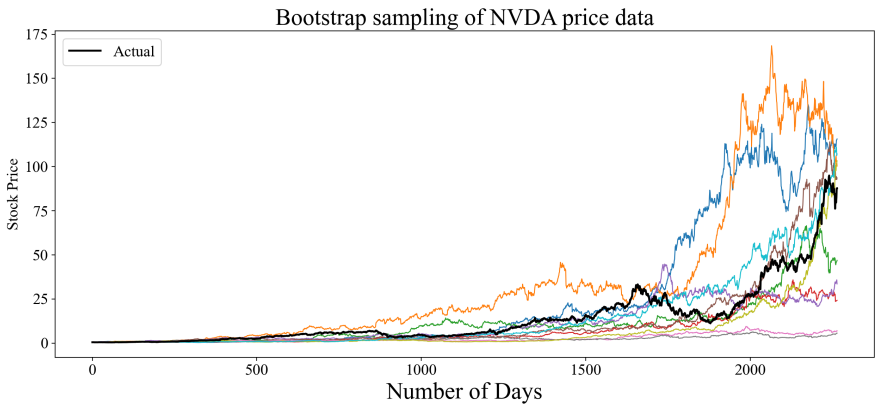

Figure 10: Illustration of 10 resampled stock price path realizations for NVDA.

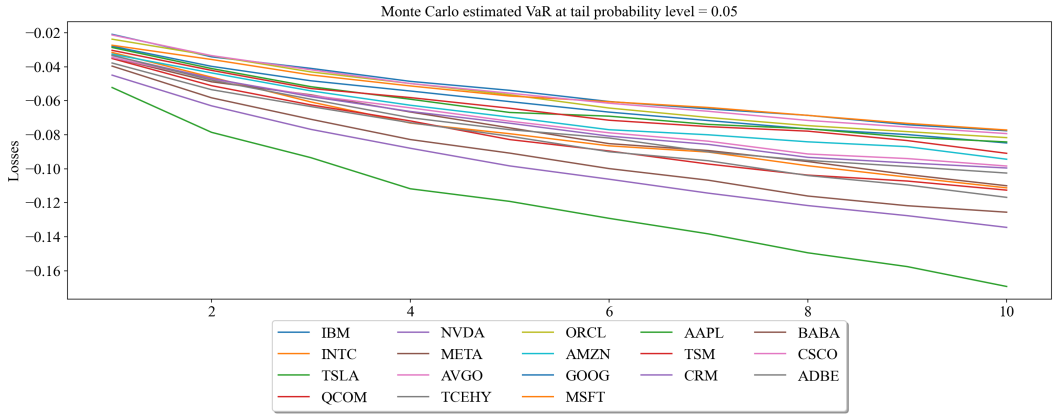

Figure 11: 5% VaR estimation using the historical simulation approach, with $N_{\text{MC}} = 10^4$ realizations for each stock. The results are visualized as a PnL of initial value on April 30th, 2015.

3 Methodology

In this section, we describe the methodology used to compare the performance of various LSTM models for forecasting stock prices and predicting derivative prices. We implemented several LSTM-based architectures, including standard LSTM, stacked LSTM, attention-based LSTM, and ensemble LSTM models. Each model was trained on historical stock price data and evaluated based on their predictive accuracy and computational efficiency.

3.1 Experimental setup

For the purpose of strategy backtesting or time series forecasting, the concept of statistical overfitting slightly differs and is more aligned with that of model predictive control and reinforcement learning [13].

To avoid look-ahead bias, the train-test split of data must be done in a "walk-forward" fashion, illustrated in Figure 12. Furthermore, the training data does not permute the time index (i.e. the time ordering of observations is preserved).

For the purpose of this presentation, we fix the general training setups used for all models. We train one separate model for each stock using a 60-40 train-test split using the adjusted close data from April 30, 2015 to April 30, 2021. More precisely, the training data comprises of 907 days of records, ranging from April 30, 2015 to December 3, 2018; the test data comprises of 605 records from December 4, 2018, to April 30, 2021.

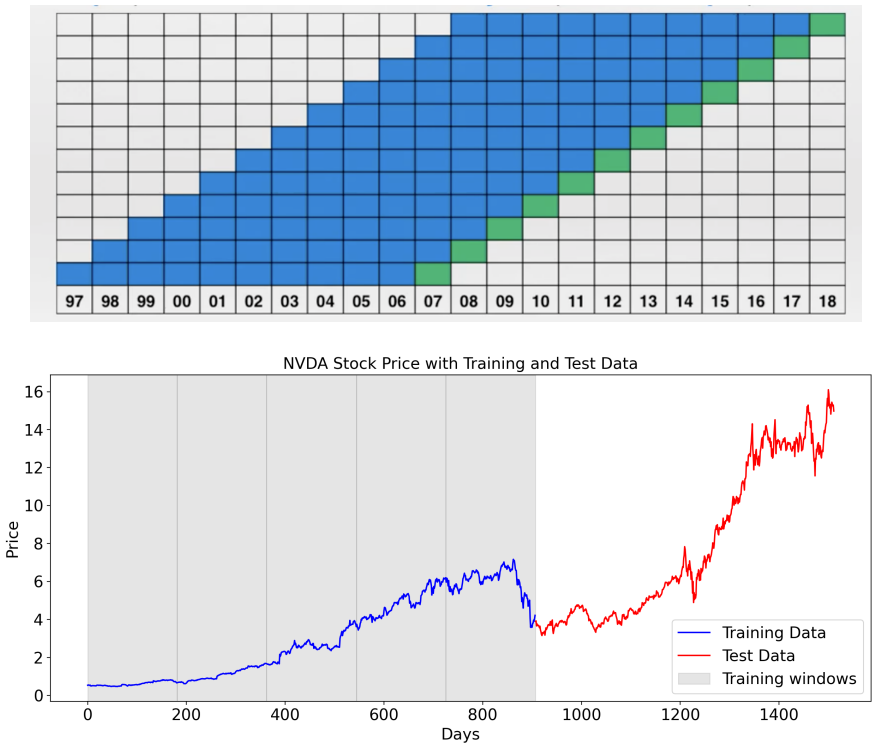

Figure 12: (Top) Illustration of walk-forward train-test split in time series forecasting. On a grid representation of various time series (by year), the blue color part shows data used for training. (Bottom) 60-40 train-test split on NVDA price data from April 2015 to December 2018.

We focus on exhibiting typical model prediction quality rather than the ability to capture rare events. Therefore, the specific range of dates is chosen to avoid the extreme impacts of the COVID event that occurred after early 2021. The window size for training and prediction is fixed as $w = 10$, i.e. $x_t, y_t \in \mathbb{R}^w$. More precisely, $x_t := P_{t:t+w}$ is a slice of the observed price time series. A loss convergence study was performed before setting the training hyperparameters. For each model, we train for 1000 epochs with loss computed every 50 epochs, using the mean squared error (MSE) loss function defined as:

$$\text{MSE} := \frac{1}{K_{\text{train}}} \sum_{k=1}^{K_{\text{train}}} \|\widehat{P}_{(k-1)w:kw} - P_{(k-1)w:kw}\|_2^2 \tag{17}$$

where $K_{\text{train}} := \lfloor T_{\text{train}}/w \rfloor$ is the total number of windows used for training. The ":" subscript is used to represent slicing over time. Unless otherwise mentioned, an Adam optimizer is used with a learning rate of 10^{-4}, with a batch size of 8. To prevent overfitting, a dropout layer with $p = 0.2$ is added before the output layer for all model architectures. The specific architecture and training results are reported separately in each section below.

3.2 Single LSTM Model

The primary concept of Recurrent Neural Networks (RNNs) is to utilize sequential observations from earlier stages to predict future trends. The Long Short-Term Memory (LSTM) model is an advanced version of RNNs, designed to address the limitations of RNNs in capturing long-term dependencies.

The LSTM model introduces memory cells that enable the capture of long-term dependencies among time lags. These memory cells replace the hidden layer neurons in traditional RNNs and filter information through a gate structure to maintain and update their state. The gate structure comprises the input gate, forget gate, and output gate, each playing a crucial role in managing the flow of information within the LSTM [3]. An LSTM layer is illustrated in Figure 13.

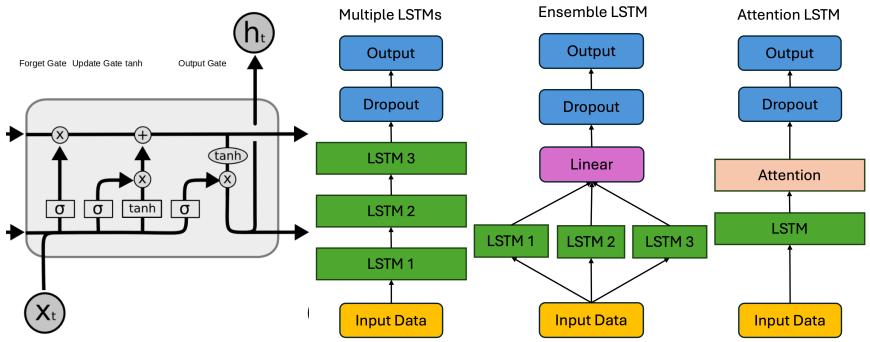

Figure 13: (Left to right) (1) Illustration of an LSTM module, with forget gate, update gate, tanh, and output gate. (2) Illustration of the sequential LSTM architecture, with 3 base LSTM modules (Section 3.3). (3) ensemble LSTM, with 3 base LSTM modules. (4) Base LSTM with an attention layer.

In the module, a forget gate determines what information from the previous cell state should be discarded. The update gate decides which new information should be added to the cell state. The tanh activation function then generates candidate values to update the cell state, which is regulated by the update gate. Finally, the output gate determines what information from the cell state should be outputted and carried to the next cell. These gates collectively enable the LSTM to selectively remember or forget information, thereby addressing the limitations of traditional RNNs in capturing long-term dependencies.

In forecasting financial time series, two components should be taken into account: (1) autocorrelation structure and (2) stochasticity. LSTM manages these aspects by allowing the model to capture long-term dependencies and filter out noise. The forget gate discards irrelevant past information that is attributable to noise fluctuations. On the other hand, the update gate integrates new changes in an online manner. Naturally, the training data for LSTMs should be blocks of the observed time series rather than randomly permuted; in other words, the time ordering in each "slice" of time series is crucial. The base LSTM module is implemented with a single hidden layer with size of 50. The predicted series for each stock are visualized in Figure 14.

In general, the base model does not capture sharp breakouts as manifested in the predictions of AMZN, TSM, TSLA and AVGO. By inspecting the results of the price changes in Figure 2, we observe that the less volatility clustering is present, the better the predictive accuracy of LSTM, such as in the case of IBM, BABA, and CSCO, which corresponds to our general expectation of the deep learning model as a volatility model. In particular, TSLA, AVGO and TSM demonstrate a high degree of clustering, corresponding to their poor predictive accuracy.

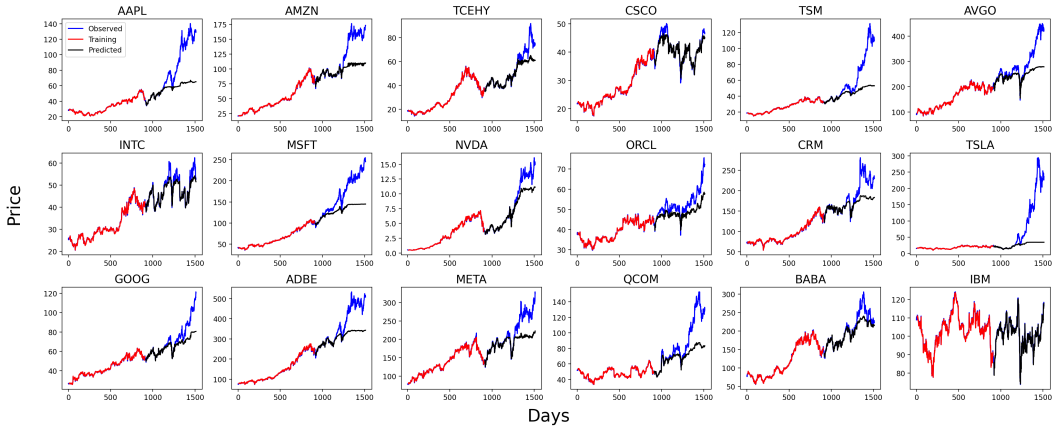

Figure 14: Predicted time series using the base LSTM of hidden layer size 50, each trained in 500 epochs.

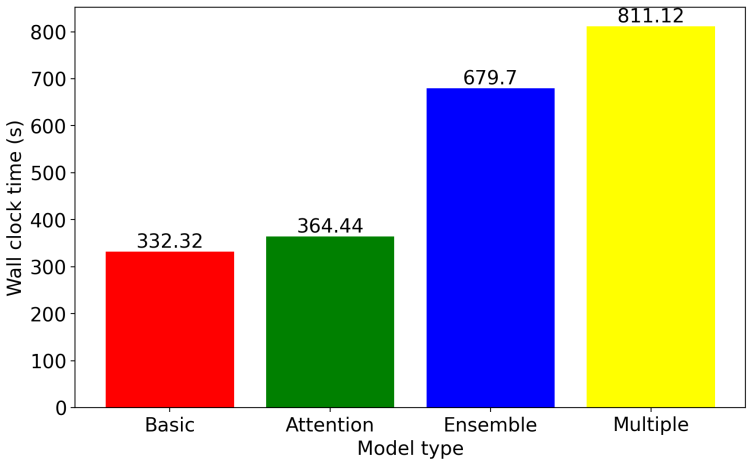

Figure 15: Runtime comparison (in seconds) for training each LSTM architecture for 500 epochs.

For the remainder of the methodology section, we focus our attention on improving model predictive accuracy by applied different architectures. Since the qualitative behavior of predicted time series is similar across all stocks, we henceforth for simplicity only present the comparison of results yielded by different architectures on the TSLA stock.

3.3 Multiple LSTMs

In addition to using only one LSTM layer, a natural extension is to consider passing the input data through multiple LSTM layers sequentially. Using the base LSTM module in Section 3.2, we sequentially connect 3 modules before outputting the prediction. In general, we expect a more complex model to achieve lower error on training data with potential overfitting. The model architecture is illustrated in Figure 13.

3.4 Stacked ensemble LSTM

In order to further reduce the variance of model predictions on test data, we consider an ensemble architecture with 3 base modules (illustrated in Figure 13), where the output of multiple LSTMs are pooled linearly. In particular, this architecture differs from a bagged ensemble, as the base LSTMs are trained simultaneously using one set of training data instead of independently on bootstrap subsamples; furthermore, the linear layer also incorporates trainable weights and bias, whereas a bagged ensemble takes the average of predictions. The particular model choice is made over a bagging ensemble in order to use different window sizes and model complexities to potentially capture varying frequencies of the entire training dataset. In our experiment, we use 3 base LSTMs respectively with window sizes $w = 10, 20, 50$, with one hidden layer of sizes $50, 100, 250$. In the final linear aggregation, we restrict a final window size of 10 and discard the extra predictions coming from LSTMs of larger window sizes.

3.5 Attention LSTM

Machine learning algorithms are frequently inspired by biological systems and human cognitive processes [12]. In human perception, not all information is treated equally; critical information is prioritized and processed first. This selective attention is crucial in financial markets as well, where security prices reflect the varying significance of market information. This prompts the integration of attention mechanism into the LSTM framework. We propose to apply softmax attention to dynamically assign importance to input data block time series and optimize the significance of each element based on learned weights. This enhancement allows the model to process and predict financial time series data with greater accuracy. The attention layer directly follows the LSTM, aiming to extract key features and ignoring the redundant features of the final prediction.

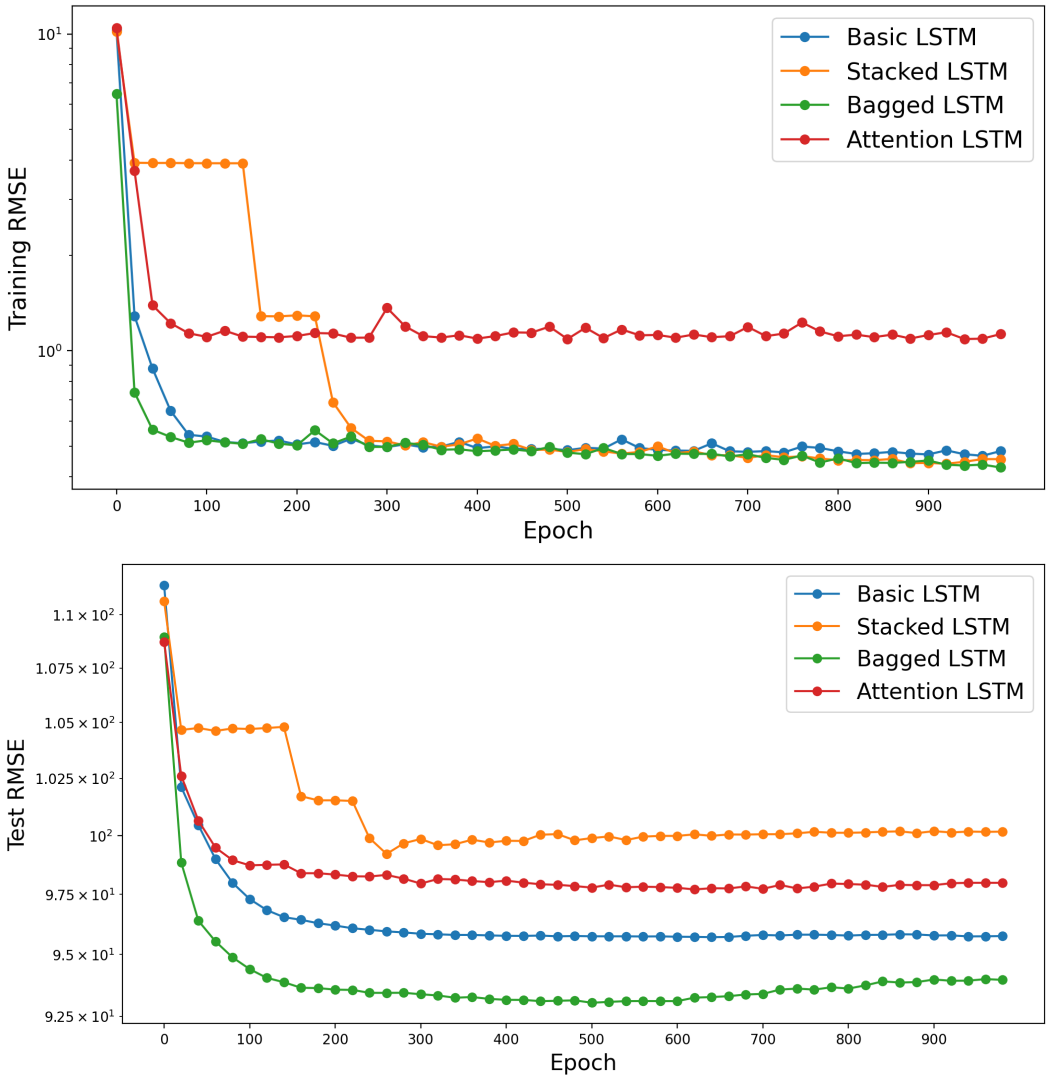

Figure 16: Comparison of LSTM architectures in terms of training error (Top) and test error (Bottom) when trained on the TSLA stock data.

3.6 Results discussion

The train and test loss curves for the architectures described above are plotted in Figure 16, with Figure 15 showing their respective wall clock times for training 500 epochs. While all models suffer to predict the TSLA stock accurately due to the sharp increase in value, we make the following observations for typical trainings of the above architectures:

- While the multiple LSTM with 3 base modules (Section 3.3) contains more neural connections, it shows mild overfitting (as seen by the higher test error but lower training error) compared to the base model, and does not provide much improvement.
- Compared to either basic LSTM (Section 3.2) or multiple LSTM (Section 3.3), the ensemble method shows large improvement both in terms of convergence speed and level of errors in training and test sets.
- While Attention LSTM has been shown to outperform traditional LSTM architectures in time series forecasting [5], in this case of extreme event, the training and test accuracy of Attention LSTM is moderate.

4 Conclusion and future work

- **LSTM for Derivatives Pricing**: Using models like Black-Scholes for European options pricing, the payoff function varies with the derivative type. For example, to predict a European call option's price, one can use the forecasted stock price on the maturity date to calculate the expected payoff as $\max(\widehat{P}_t - K, 0)$, and discount it to the present value. Although LSTM models leverage their ability to capture complex time series dependencies for accurate derivative pricing, their performance depends on the quality and quantity of historical data. Enhancements include incorporating additional market indicators, regular retraining, and using ensemble methods. Future work could explore using pretrained models to warm start derivative price forecasting models to further improve efficiency.
- **Explainable Applications of LSTMs**: Future work could focus on making LSTM models more interpretable by embedding LSTM predictions into linear regression models. This hybrid approach would combine the predictive power of LSTMs with the explainability of linear regression, allowing for better understanding and interpretation of the LSTM's decision-making process especially for portfolio construction and rebalancing purposes. Incorporating attention mechanisms as done in Section 3.5, can assist in highlighting the parts of the input sequence the LSTM is focusing on when making predictions. Finally, in order to quantitatively characterize the behavior of adding more LSTM layers, such as the experiment in Section 3.4, the study of benign overfitting in time series deep learning remains an open problem [9].

References

[1] Peter Fortune. “Are stock returns different over weekends? A jump diffusion analysis of the "weekend effect"”. English. In: *New England Economic Review* (Sept. 1999). Copyright Federal Reserve Bank of Boston Sep/Oct 1999; Last updated - 2023-11-30; CODEN - NWEEAP; SubjectsTermNotLitGenreText - United States–US, pp. 3–20. ISSN: 00284726. URL: http://proxy.uchicago.edu/login?url=https://www.proquest.com/scholarly-journals/are-stock-returns-different-over-weekends-jump/docview/204604144/se-2.

[2] *GitHub - ranaroussi/yfinance: Download market data from Yahoo! Finance's API — github.com*. https://github.com/ranaroussi/yfinance/tree/main. [Accessed 06-07-2024].

[3] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[4] Jefferson M. Jones et al. “Estimates of SARS-CoV-2 Seroprevalence and Incidence of Primary SARS-CoV-2 Infections Among Blood Donors, by COVID-19 Vaccination Status — United States, April 2021–September 2022”. In: *MMWR. Morbidity and Mortality Weekly Report* 72.22 (June 2023), pp. 601–605. ISSN: 1545-861X. DOI: 10.15585/mmwr.mm7222a3. URL: http://dx.doi.org/10.15585/mmwr.mm7222a3.

[5] Sangyeon Kim and Myungjoo Kang. *Financial series prediction using Attention LSTM*. 2019. arXiv: 1902.10877 [cs.LG]. URL: https://arxiv.org/abs/1902.10877.

[6] Pierre Lequeux. “Financial risk forecasting: The theory and practice of forecasting market risk with implementation in R and MATLAB”. In: *Journal of Derivatives & Hedge Funds* 17.3 (Sept. 2011), pp. 279–280. ISSN: 1753-965X. DOI: 10.1057/jdhf.2011.21. URL: http://dx.doi.org/10.1057/jdhf.2011.21.

[7] James G. MacKinnon. *Critical Values For Cointegration Tests*. Working Paper 1227. Economics Department, Queen's University, Jan. 2010. URL: https://ideas.repec.org/p/qed/wpaper/1227.html.

[8] George Marsaglia, Wai Wan Tsang, and Jingbo Wang. “Evaluating Kolmogorov's Distribution”. In: *Journal of Statistical Software* 8.18 (2003), pp. 1–4. DOI: 10.18637/jss.v008.i18. URL: https://www.jstatsoft.org/index.php/jss/article/view/v008i18.

[9] Shogo Nakakita and Masaaki Imaizumi. *Benign Overfitting in Time Series Linear Model with Over-Parameterization*. 2023. arXiv: 2204.08369 [math.ST]. URL: https://arxiv.org/abs/2204.08369.

[10] Miskolczi Panna. “Note On Simple And Logarithmic Return”. In: *APSTRACT: Applied Studies in Agribusiness and Commerce* 11.1-2 (Sept. 2017). DOI: 10.22004/ag.econ.265595. URL: https://ideas.repec.org/a/ags/apstra/265595.html.

[11] Thomas Wong and Mauricio Barahona. *Feature Engineering Methods on Multivariate Time-Series Data for Financial Data Science Competitions*. 2023. arXiv: 2303.16117 [q-fin.ST]. URL: https://arxiv.org/abs/2303.16117.

[12] Kelvin Xu et al. *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*. 2016. arXiv: 1502.03044 [cs.LG]. URL: https://arxiv.org/abs/1502.03044.

[13] Chiyuan Zhang et al. *A Study on Overfitting in Deep Reinforcement Learning*. 2018. arXiv: 1804.06893 [cs.LG]. URL: https://arxiv.org/abs/1804.06893.