GARCH-Informed Deep Learning with Shared Parameters for Multi-Asset Volatility Forecasting

Yu-Chieh Tsai
National Taiwan University
Taipei, Taiwan
r13723046@ntu.edu.tw

Yu-Chi Ko
National Taiwan University
Taipei, Taiwan
r14946005@ntu.edu.tw

Ta-Kang Kao
National Taiwan University
Taipei, Taiwan
r12922156@ntu.edu.tw

Ho-Chien Huang
National Taiwan University
Taipei, Taiwan
r14946007@ntu.edu.tw

Jovytta E. Yauwanta
National Taiwan University
Taipei, Taiwan
b12902087@csie.ntu.edu.tw

Yu-Chen Kuo
National Taiwan University
Taipei, Taiwan
b12902061@csie.ntu.edu.tw

Abstract

We study GARCH-informed deep learning for joint volatility forecasting across multiple assets. GARCH-based forecasts are integrated into neural networks through a mixed loss function, and the framework is evaluated using a parameter-sharing TSMixer architecture on daily data for Taiwan 50 constituents from 2005 to 2025, allowing the model to exploit cross-sectional dependence across stocks. Our results show that TSMixer consistently outperforms recurrent architectures and achieves performance comparable to, and often exceeding, traditional GARCH models. The mixing parameter λ plays a central role by governing the trade-off between average forecast accuracy and tail robustness. When appropriately chosen, λ enables the model to inherit the stability of GARCH while exploiting nonlinear and cross-sectional information.

1 Introduction

Traditional financial volatility forecasting has long relied on econometric models such as the GARCH family, which is predicated on the assumption that future volatility is a linear combination of past residuals and variance. However, this framework exhibits significant limitations. First, GARCH models are constrained by a rigid parametric structure, making them ill-equipped to capture the complex, non-linear dynamics prevalent in financial markets [31]. Second, they often fail to adapt to structural breaks such as oil crises or financial meltdowns, due to inherent lag in responding to regime shifts [8][15]. Third, as noted by Lamoureux & Lastrapes, the neglect of structural shifts leads to an overestimation of volatility persistence [21][23]. Given these limitations in flexibility and the inability to effectively integrate heterogeneous data, there has been a paradigm shift towards deep learning models capable of high-dimensional non-linear approximation.

This study investigates the performance boundaries of various temporal deep learning frameworks in volatility forecasting. We construct a multi-dimensional feature space that integrates foundational price-volume data with Zura Kakushadze's Alpha 101 factors, volume momentum, and discrete corporate event indicators, specifically earnings announcement schedules. Uniquely, we employ an experimental design that utilizes GARCH predictions as expert features fed into deep learning architectures. This approach allows for a comparative analysis between pure data-driven models and hybrid models, aiming to verify whether traditional econometric models still offer incremental information gain within modern neural architectures.

The primary contributions of this research are to clarify the role of traditional econometric structure in modern deep learning–based volatility forecasting:

(1) Effectiveness of GARCH-informed training objectives: We empirically examine whether incorporating GARCH-based reference forecasts through the loss function provides a stabilizing inductive structure that improves robustness and generalization, rather than merely increasing model complexity or overfitting risk.
(2) Architecture–performance trade-offs: We systematically compare recurrent and all-MLP architectures (e.g., LSTM and TSMixer) in terms of their ability to capture nonlinear and cross-sectional volatility dynamics relative to their computational costs.
(3) Practical implications for volatility modeling: Our results provide empirical guidance on when and how traditional econometric models should be integrated into deep learning frameworks, highlighting the effectiveness of GARCH-informed objectives over naive inclusion of high-dimensional covariates.

2 Related Work

GARCH Family. Econometric approaches to volatility are dominated by the GARCH framework, which models variance as a linear function of past squared residuals and variances. To overcome the symmetry assumption of the vanilla GARCH, which contradicts the stylized fact that negative shocks impact volatility more significantly than positive ones, various asymmetric extensions have been proposed. EGARCH models the logarithm of conditional variance to avoid non-negativity constraints and captures asymmetric responses through standardized residuals [25]. Similarly, GJR-GARCH and TGARCH introduce indicator functions to partition the impact of positive and negative shocks on variance and standard deviation, respectively [13][32]. The APARCH model offers a generalized framework by introducing a flexible power parameter δ, theoretically nesting GARCH, GJR, and TGARCH as special cases [9].

However, the theoretical flexibility of these models does not consistently translate to forecasting superiority. Theoretical analysis distinguishes between "asymmetry" and true "leverage effects," demonstrating that popular variants like EGARCH and GJR capture the former but often fail to model the latter correctly compared to specifications like AGARCH [4]. Furthermore, empirical evidence from the S&P 500 and NASDAQ during crisis periods (Dot-com, Global Financial Crisis, COVID-19) indicates that simple GARCH models often outperform complex asymmetric variants in out-of-sample tasks [10]. This paradox, where complexity increases fitting but degrades robustness, suggests that while GARCH variants capture valuable signals, they are best utilized as expert features rather than standalone predictors in non-linear market environments.

Other Features for Volatility Prediction. Extensive literature has explored the utility of diverse market indicators in volatility forecasting. Karpoff [18] provided a comprehensive review confirming the positive correlation between trading volume and price volatility. Building on this, French & Roll investigated calendar anomalies such as the "Monday effect" [12], while Norberg [26] examined the interplay between abnormal trading volume and volatility . Furthermore, Atilgan [1] demonstrated the predictive power of volatility spreads regarding abnormal returns, particularly during earnings announcement periods .

In the realm of quantitative trading practice, proprietary factors are rarely disclosed due to the industry's highly secretive nature. A notable exception is the work of Zura Kakushadze, who published explicit mathematical formulations for 101 real-world trading alphas ("Alpha 101") and posited a strong sub-linear relationship between returns and volatility [17].

However, the sustained efficacy of such published factors is debated. McLean & Pontiff observed a "signal decay" phenomenon, suggesting that the predictive power of public signals diminishes over time due to arbitrage overcrowding or potential overfitting [24]. Additionally, Chen & Kawashima highlighted that directly applying the static features of Alpha 101 to modern dynamic models can lead to significant performance deterioration [5].

Recent advancements in deep learning for volatility forecasting have increasingly relied on latent representations automatically extracted by neural networks. While these implicit factors contribute to substantial performance gains, they often suffer from a lack of interpretability, presenting a "black box" challenge that contrasts with the explicit nature of traditional econometric factors [19].

Time-Series Deep Learning Frameworks. Early deep learning approaches for time-series modeling were predominantly based on Recurrent Neural Networks (RNNs). Specifically, Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) introduced gating mechanisms to effectively mitigate the vanishing gradient problem inherent in traditional RNNs [16][7]. This capability allows them to capture long-term dependencies, making them particularly well-suited for forecasting financial asset volatility, which is characterized by significant long-memory properties [11].

In the context of risk management, point forecasts are often insufficient for quantifying tail risk. The DeepAR model, a variant of autoregressive RNNs, addresses this by learning a global model to generate probabilistic distributions rather than single-point estimates [28]. By producing a full conditional distribution, DeepAR is critical for calculating risk metrics such as Value at Risk (VaR) and Expected Shortfall (ES), demonstrating superior stability during periods of high market volatility.

Furthermore, to address volatility spillover effects in multi-asset portfolios, TSMixer proposes an all-MLP (Multi-Layer Perceptron) architecture [6]. By alternately applying "time-mixing" and "feature-mixing" layers, it effectively extracts cross-variate information. Empirical studies indicate that TSMixer achieves performance comparable to state-of-the-art Transformer models on multivariate long-term forecasting benchmarks, while maintaining significantly higher computational efficiency [29].

Regarding interpretability, the N-BEATS architecture utilizes a residual stacking design to decompose predictions into interpretable trend and seasonality components [27]. This provides transparency into the intrinsic structure of volatility, thereby addressing the "black box" criticism often directed at deep learning models.

In summary, current literature trends indicate that deep learning frameworks have evolved from simple curve-fitting tools into a comprehensive ecosystem encompassing probabilistic quantification, structural decomposition, and multivariate synergy. This evolution provides a robust methodological foundation for addressing structural breaks and non-linear volatility in financial markets.

3 Task Definition

Volatility forecasting is an indispensable component of financial trading. Accurate volatility forecasts provide several important benefits, including (1) precise valuation of derivative instruments such as options, (2) portfolio risk management and capital allocation, and (3) anticipation of future market dynamics.

Compared to expected returns, extensive empirical evidence in financial economics suggests that volatility is predictable. In particular, volatility exhibits at least three well-documented stylized facts: (1) serial dependence and volatility clustering, (2) asymmetric responses to market movements, where downside shocks generate larger volatility than upside shocks, and (3) long-term mean reversion. Although these properties are well understood, their practical implications remain unclear. Specifically, the strength and persistence of volatility autocorrelation, as well as the horizon over which mean reversion occurs, are difficult to quantify. The presence of numerous lagged variables renders traditional linear regression models ineffective in out-of-sample forecasting, thereby creating opportunities for machine learning methods to better capture complex volatility dynamics.

Formally, we define the volatility forecasting problem as

$$\sigma_t^2 = f(\mathbf{X}_t) + \varepsilon_t, \tag{1}$$

where $f(\cdot)$ denotes a machine learning model that integrates historical volatility and auxiliary features $\mathbf{X}_t$ to predict future volatility.

4 Data

4.1 Datasets

The dataset used in this study is obtained from *TQuant* [30], a quantitative financial database provided by TEJ. A key feature of

TQuant is its point-in-time (PIT) data structure, which ensures that only information available at each historical point is used, thereby avoiding look-ahead bias. Due to computational constraints, we restrict our analysis to the constituents of the Taiwan 50 Index as of November 2025, together with the market index. The sample period spans from January 2005 to November 2025. For training and implementation convenience, we exclude four stocks with relatively late listing dates. In practice, models with shared parameter structures remain applicable to stocks with shorter trading histories.

Features. Table 1 summarizes the structure of the dataset used in this study. It includes stock-level volatility measures and auxiliary variables. For the market index, auxiliary variables are not available; therefore, only historical volatility is used for modeling.

The auxiliary variables (*Alphas*) consist of the 101 operator-based factors proposed by Kakushadze [17], which are generated using genetic algorithms (GA). In addition, we extend this set by constructing 70 volume-related factors. From a traditional asset pricing perspective, factor premia arise as compensation for bearing risk. Kakushadze further documents that the predictive power of operator-based factors can be partially explained by volatility, which supports the use of these Alphas as auxiliary variables in volatility forecasting. Finally, since the days following earnings announcements and financial statement releases often exhibit significant volatility changes, we include indicator variables for these events as additional features.

Table 1: Dataset structure

Stock ID	Date	σ_t^2	Earnings	Financials	Alpha 001–172
1216	2005-01-01	...	0	0	...
2330	2005-01-01	...	1	0	...
⋮	⋮	⋮	⋮	⋮	⋮
5880	2025-11-26	...	0	0	...
6505	2025-11-26	...	0	0	...

4.2 Data Preprocessing

Preprocessing time-series data is a non-trivial task. Gradient-based optimization and regularization methods in machine learning require input features to be on comparable scales. Moreover, time-series data are subject to distribution drift and potential look-ahead bias. As a result, preprocessing procedures for deep learning models applied to time-series data differ substantially from those used in traditional time-series forecasting or standard machine learning applications.

Log-Volatility Transformation. A large body of empirical evidence suggests that volatility follows an approximately log-normal distribution. Applying a logarithmic transformation improves its statistical properties. Accordingly, we define the volatility target as

$$\ln(1+\sigma_t^2) = \ln\Big(1+(r_t-\hat{\mu}_t)^2\Big), \tag{2}$$

where the additive constant 1 prevents numerical issues caused by zero volatility observations. During evaluation, predicted values are transformed back to the original (non-log) scale. To assess whether the log transformation improves machine learning performance, we compare models trained with and without the log-volatility specification.

In this study, r_t denotes the daily close-to-close *log return*, expressed in percentage terms. The conditional mean return $\hat{\mu}_t$ is estimated using a rolling autoregressive model of order one (AR(1)) fitted over the past 90 trading days. Specifically, for each asset, an AR(1) model is estimated using historical returns $\{r_{t-90}, \ldots, r_{t-1}\}$, and the one-step-ahead forecast is taken as $\hat{\mu}_t$. This rolling estimation procedure relies exclusively on information available prior to time t, thereby avoiding look-ahead bias.

Global Normalization. The distributions of Alpha features are highly heavy-tailed. Following Gu et al. [14], we apply daily cross-sectional rank normalization:

$$\widetilde{\alpha}_{i,t} = 2 \cdot \frac{\mathrm{rank}(\alpha_{i,t}) - 1}{N-1} - 1, \tag{3}$$

which compresses all Alpha distributions into the interval $[-1, 1]$. After normalization, the economic interpretation of an Alpha corresponds to its relative cross-sectional standing among stocks on a given day.

In addition, volatility series exhibit substantial scale heterogeneity across assets. Without series-wise normalization, models tend to focus disproportionately on stocks with persistently higher volatility levels. To address this issue, we apply Z-score normalization to each asset's volatility series, using the mean and standard deviation estimated from the training set. The same statistics are applied to the validation and test sets, and predictions are inverse-transformed when reporting results.

Local Normalization. To further mitigate distribution drift in time-series data, we adopt Reversible Instance Normalization (RevIN) proposed by Kim et al. [20]. RevIN normalizes the input sequence at the model's input layer and applies the inverse transformation at the output layer. This procedure ensures that each time series is standardized to zero mean and unit variance at each time step, substantially alleviating distribution shift and improving training stability.

4.3 Exploratory Data Analysis and Visualization

To understand the underlying structure of our feature space and its relationship with market volatility, we conducted a comprehensive Exploratory Data Analysis (EDA) focusing on the Alpha 101 factors and the target volatility metrics.

Distribution and Statistical Characteristics. The Alpha 101 factors exhibit diverse statistical distributions across the constituents of the 0050 Index. While some factors are bounded within specific ranges (e.g., $[0, 1]$ or $[-1, 1]$), others demonstrate heavy tails and significant kurtosis, which is typical of financial time-series data.

As shown in the histograms, factors such as Alpha001 show distinct behaviors compared to volume-based alphas. These variations in scale and the presence of extreme outliers justify the necessity of robust feature scaling and clipping during the preprocessing stage to prevent gradient instability in deep temporal architectures like TSMixer and LSTM.

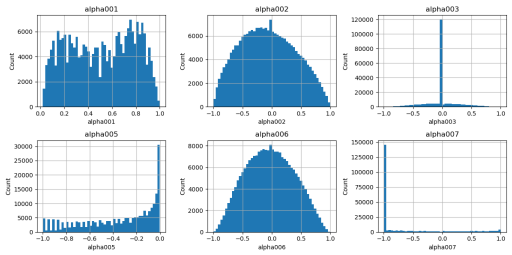

Figure 1: Histograms of selected Alpha 101 factors.

Correlation with Volatility Targets. A critical component of our EDA was evaluating the predictive signal of these alphas relative to our targets: log returns r_t, innovation-based variance σ_t^2, and GARCH-informed variance $\hat{\sigma}^2_{t_{\text{GARCH}}}$.

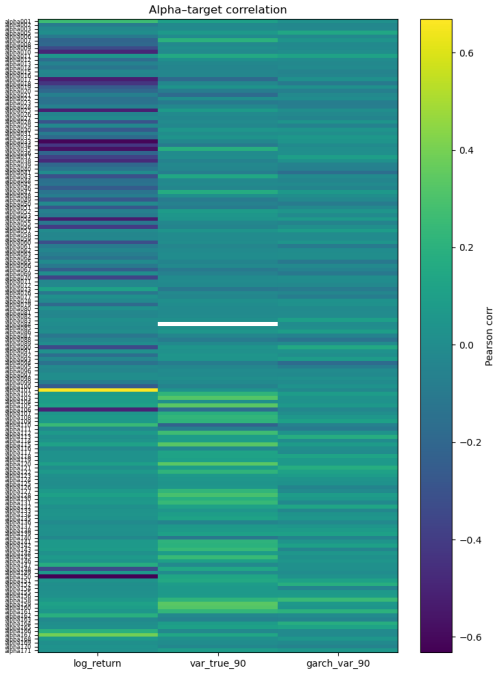

Figure 2: Correlation heatmap between Alpha 101 factors and volatility targets.

Our correlation analysis reveals that while individual Alpha 101 factors were originally designed for alpha generation (predicting returns), many possess a secondary "volatility-predictive" power. The heatmap indicates non-zero correlations with variance targets, suggesting that these alphas contain valuable information for volatility forecasting beyond simple return prediction. For instance, some factors act as leading indicators for market reversals which are often associated with volatility clusters.

Feature Linearity and Decile Analysis. To assess the relationship between features and future market movements, we performed a decile analysis on Alpha001. We calculated the mean future return for each decile to determine if a linear relationship exists.

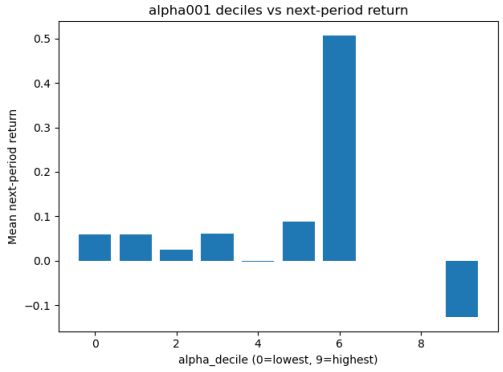

Figure 3: Decile analysis of Alpha001 versus mean future returns.

The resulting bar chart shows a non-monotonic relationship, particularly in the extreme deciles. This behavior suggests that extreme factor values often lead to higher uncertainty and varied return profiles. This empirical evidence justifies the use of non-linear deep learning models over traditional linear econometric frameworks, as neural networks are better equipped to capture these complex dynamics.

Time-Series Stability and Idiosyncratic Noise. Finally, since our dataset encompasses individual stocks from the 0050 Index, we examined the time-series stability of these factors across different assets to distinguish between market-wide trends and individual noise.

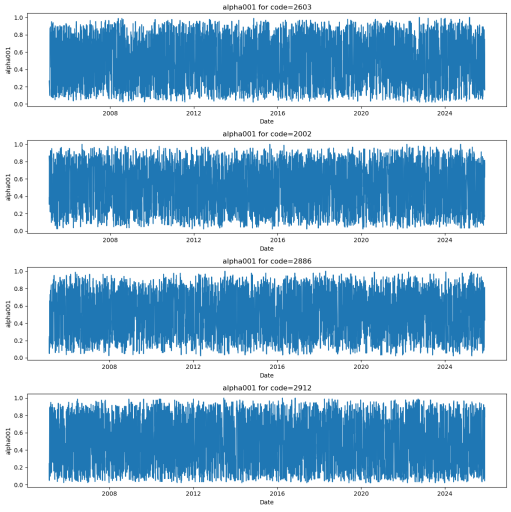

Figure 4: Time-series trajectories of Alpha001 across sample stocks in the 0050 Index.

The time-series trajectories indicate that while the alphas share common "market regime" components (visible when lines move

in tandem), there is significant idiosyncratic noise at the individual stock level. This observation motivates our proposed GARCH-informed loss function: we use the theoretically stable GARCH prior to regularize the noisy individual signals during the training of our deep learning models.

5 Proposed Method

5.1 Models

We employ several econometric and machine-learning models for volatility forecasting.

GARCH(1,1). We first use the GARCH model proposed by Bollerslev [3], which remains a classical benchmark in volatility modeling. The conditional variance is specified as

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2, \tag{4}$$

where $\epsilon_t = r_t - \hat{\mu}_t$ denotes the return innovation.

GJR-GARCH(1,1,1). Second, we consider the GJR-GARCH model proposed by Glosten et al. [13], which extends the standard GARCH framework by allowing negative shocks to exert a larger marginal impact on the conditional variance:

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \gamma_1 \epsilon_{t-1}^2 \mathbf{1}\{\epsilon_{t-1} < 0\} + \beta_1 \sigma_{t-1}^2. \tag{5}$$

TGARCH(1,1,1). Third, we include the Threshold GARCH (TGARCH) model introduced by Zakoian [32], which captures asymmetric volatility responses by modeling the conditional standard deviation:

$$\sigma_t = \alpha_0 + \alpha_1 |\epsilon_{t-1}| + \gamma_1 |\epsilon_{t-1}| \mathbf{1}\{\epsilon_{t-1} < 0\} + \beta_1 \sigma_{t-1}. \tag{6}$$

For all GARCH-type models, parameters are re-estimated using a rolling window of 90 trading days to account for potential time variation in volatility dynamics.

LSTM. Fourth, we employ Long Short-Term Memory (LSTM) networks, which are widely used in time-series forecasting due to their ability to capture nonlinear dynamics and long-range temporal dependencies.

TSMixer. Finally, we adopt TSMixer proposed by Chen et al. [6], a state-of-the-art time-series forecasting model based on a lightweight two-layer MLP architecture. Compared to recurrent models such as LSTM, TSMixer enables efficient parameter sharing and flexible temporal feature mixing, making it well suited for large-scale volatility forecasting tasks.

5.2 Loss Function

We adopt a composite loss function that combines information from a GARCH-based benchmark and the realized volatility target. Let $\hat{\sigma}_\theta^2$ denote the volatility forecast produced by a given model. The total loss is defined as

$$\mathrm{MSE}_{\mathrm{total}} = \lambda \cdot \mathrm{MSE}\left(\sigma_{\mathrm{true}}^2, \hat{\sigma}_\theta^2\right) + (1-\lambda) \cdot \mathrm{MSE}\left(\hat{\sigma}_{\mathrm{GARCH}}^2, \hat{\sigma}_\theta^2\right), \tag{7}$$

where σ_{true}^2 denotes the innovation-based proxy for realized volatility and $\hat{\sigma}_{\mathrm{GARCH}}^2$ is the variance forecast obtained from a fitted GARCH model.

Xu et al. [31] show that this loss function substantially improves out-of-sample forecasting performance by appropriately combining machine learning predictions with a GARCH-based prior. Their empirical results suggest that $\lambda = 0.01$ yields the best out-of-sample performance. Following their approach, we employ the same loss function in this study and re-tune λ for each model to further examine its effectiveness and limitations.

5.3 Hyperparameter Tuning

We select hyperparameters using time-series cross-validation. The hyperparameter search space for each model is summarized in Table 2. The dataset is split chronologically into training, validation, and test subsets, which account for 70%, 20%, and 10% of the total sample, respectively. After hyperparameter optimization, the training and validation sets are merged, and the model is retrained using the optimal configuration. The resulting model is then evaluated on the test set to assess out-of-sample forecasting performance.

We employ Optuna for hyperparameter optimization, which is one of the most widely used hyperparameter tuning frameworks. For the sampling strategy, we use the Tree-structured Parzen Estimator (TPE) sampler. In the initial trials, TPE performs random search and partitions the observed trials into a set of promising and unpromising configurations. Conditional densities are then estimated for both sets, and subsequent trials are selected to maximize the expected improvement. To reduce computational cost, we adopt the MedianPruner as the pruning strategy. During the search process, if the performance of a trial is worse than the median of completed trials and the minimum number of required trials has been reached, Optuna terminates the trial early to avoid unnecessary computation.

Due to computational constraints, LSTM and TSMixer models are trained only once and applied to the entire forecasting period. In contrast, GARCH-type models are re-estimated daily using a rolling-window approach, as they are computationally less demanding. As a result, compared to GARCH models, LSTM and TSMixer are more likely to be affected by distribution drift under this experimental setting.

Table 2: Hyperparameter search space

Hyperparameter	TSMixer	LSTM
Hidden size	[16, 32, 64]	[16, 32, 64]
Dropout rate	[0.1, 0.2, 0.3]	[0.1, 0.2, 0.3]
Feed-forward hidden size	[16, 32, 64]	–
Number of blocks	[1, 2, 4]	–
Number of layers	–	[1, 2, 3]
Learning rate	$[10^{-2}, 10^{-4}]$	$[10^{-2}, 10^{-4}]$
Gamma	0.99	0.99
Learning rate schedule	Exponential	Exponential
Batch size	32	32
Gradient clipping	0.5	0.5
Epochs	10	10
Early stopping patience	2	2
Input length	90	90

5.4 Forecast Evaluation Metrics

To evaluate volatility forecasting performance, we consider three commonly used loss functions. Let y_t denote the realized target and $\hat{y}_t$ its corresponding forecast at time t, for $t = 1, \ldots, T$.

Mean Squared Error (MSE).

$$\mathrm{MSE} = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2, \tag{8}$$

which penalizes large forecast errors more heavily due to the quadratic loss and is widely used in regression-based evaluation.

Mean Absolute Error (MAE).

$$\mathrm{MAE} = \frac{1}{T}\sum_{t=1}^{T}|y_t - \hat{y}_t|, \tag{9}$$

providing a robust measure of average forecast deviation that is less sensitive to extreme outliers.

Quasi-Likelihood Loss (QLIKE).

$$\mathrm{QLIKE} = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{y_t}{\hat{y}_t} - \log\frac{y_t}{\hat{y}_t} - 1\right), \tag{10}$$

which is widely adopted in volatility forecasting because it remains well-defined even when the true variance is latent and is robust to measurement error in volatility proxies.

6 Results

6.1 Baseline Model Comparison (Standard Loss, $\lambda = 1$)

In this section, we train five baseline models for comparison. For LSTM and TSMixer, we further consider two variants, with and without auxiliary covariates, where the auxiliary covariates consist of a set of alpha signals and indicator variables incorporated as additional input features during model training. All machine learning models are trained using the standard loss $\mathrm{MSE}\left(\sigma^2_{\text{true}}, \hat{\sigma}^2_{\theta}\right)$, with the volatility target defined as $\ln(1 + \sigma_t^2)$. The training and evaluation protocols follow the same settings described in Section 5.3. Importantly, the baseline models in this section do not employ the proposed mixed loss function, allowing us to assess the standalone performance of each model.

Table 3 reports the out-of-sample performance of the baseline models. The MAE of TSMixer-Covariates is substantially better than both the GARCH family and LSTM. In contrast, its MSE and QLIKE are slightly worse than the GARCH family but better than LSTM. Overall, this suggests that TSMixer significantly outperforms LSTM and achieves performance comparable to the GARCH family.

Given the interpretation of MAE, MSE, and QLIKE, the results indicate that TSMixer tends to be less biased on average, while its performance under extreme outliers is slightly inferior to the GARCH family. This implies a certain degree of complementarity between the two approaches.

When auxiliary covariates are included, we observe that LSTM becomes highly prone to overfitting, whereas TSMixer shows almost no degradation. We conjecture that the auxiliary covariates contain substantial noise, which makes LSTM more susceptible to overfitting. In contrast, TSMixer's cross-sectional residual-mixing mechanism mitigates this issue, leading to significantly better out-of-sample performance than LSTM.

Table 3: Out-of-sample performance of baseline models (standard loss, $\lambda = 1$).

Model	MAE	RMSE	QLIKE
GARCH	5.1794	11.3005	2.2827
GJR-GARCH	5.2712	11.6947	2.2905
TGARCH	5.2545	11.4623	2.2775
LSTM	4.4424	11.7723	2.7493
LSTM-Covariates	26.4135	82.8169	3.9617
TSMixer	4.3267	11.7544	2.6421
TSMixer-Covariates	4.3056	11.7658	2.7974

6.2 GARCH-Informed Loss Function and λ Sensitivity Analysis

As discussed in Section 5.2, we examine the sensitivity of model performance to the choice of the mixing parameter λ. Rather than fixing λ at the value suggested by Xu et al. [31], we conduct a grid search over $\lambda \in [0, 1]$. For each fixed λ, model hyperparameters are re-optimized using Optuna.

Table 4 reports the out-of-sample performance of models trained with the mixed loss function. We observe a clear trade-off pattern as λ decreases: models tend to exhibit larger MAE but smaller RMSE and QLIKE. This indicates that λ governs a trade-off between capturing average volatility dynamics and controlling extreme deviations. Notably, this empirical pattern differs from the findings of Xu et al.[31]. In our setting, no single value of λ consistently dominates across all evaluation metrics.

With λ fixed at 0.3, Figure 5 plots the out-of-sample volatility forecasts for a representative stock (TSMC, stock ID 2330) produced by TSMixer and a standalone GARCH model, together with the realized volatility proxy. The figure shows that GARCH reacts sharply to large volatility shocks, whereas TSMixer produces a smoother forecast path while remaining responsive during high-volatility periods.

Consistent with this visual comparison, at $\lambda = 0.3$, TSMixer achieves RMSE and QLIKE comparable to those of the GARCH family, while reducing MAE by approximately 10%. Moreover, TSMixer consistently outperforms LSTM across all reported metrics. These results suggest that TSMixer provides superior performance for individual stock volatility forecasting relative to both traditional linear models and LSTM-based architectures.

6.3 Results on the Market Index

Previously, we employed TSMixer with shared parameter structures, motivated by the intuition that parameter sharing can mitigate the impact of idiosyncratic noise at the individual stock level. In this section, we extend the analysis to a single asset—the market index— to examine whether TSMixer remains competitive when cross-sectional information and parameter sharing are no longer available.

Table 4: Out-of-sample performance under the mixed loss function for different values of λ.

	λ	0	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95	1
GARCH	MAE							5.18						
	RMSE							11.30						
	QLIKE							2.28						
GJR-GARCH	MAE							5.27						
	RMSE							11.69						
	QLIKE							2.29						
TGARCH	MAE							5.25						
	RMSE							11.46						
	QLIKE							2.28						
LSTM	MAE	4.69	4.99	4.96	4.80	4.67	4.63	4.57	4.53	4.48	4.45	4.45	4.43	4.44
	RMSE	11.27	11.17	11.18	11.20	11.27	11.62	11.41	11.53	11.52	11.63	11.73	11.72	11.77
	QLIKE	2.26	2.26	2.22	2.28	2.30	2.33	2.34	2.41	2.46	2.53	2.68	2.73	2.75
LSTM-Covariates	MAE	5.08	4.99	4.91	4.74	4.78	5.31	6.90	10.75	22.06	15.04	13.64	15.47	26.41
	RMSE	11.25	11.18	11.16	11.30	11.54	12.45	15.91	23.76	49.44	36.85	35.57	40.66	82.82
	QLIKE	2.27	2.26	2.26	2.28	2.33	2.44	2.57	2.75	3.06	3.37	3.53	3.63	3.96
TSMixer	MAE	4.95	4.89	4.83	4.74	4.65	4.58	4.52	4.46	4.42	4.38	4.35	4.34	4.33
	RMSE	11.16	11.16	11.17	11.21	11.29	11.35	11.42	11.49	11.56	11.63	11.69	11.73	11.75
	QLIKE	2.20	2.20	2.21	2.22	2.24	2.27	2.30	2.35	2.40	2.46	2.54	2.59	2.64
TSMixer-Covariates	MAE	4.95	4.86	4.83	4.68	4.58	4.50	4.44	4.38	4.35	4.32	4.30	4.30	4.31
	RMSE	11.24	11.27	11.27	11.30	11.34	11.39	11.40	11.50	11.54	11.62	11.67	11.74	11.77
	QLIKE	2.24	2.24	2.24	2.27	2.28	2.31	2.33	2.42	2.48	2.56	2.65	2.71	2.80

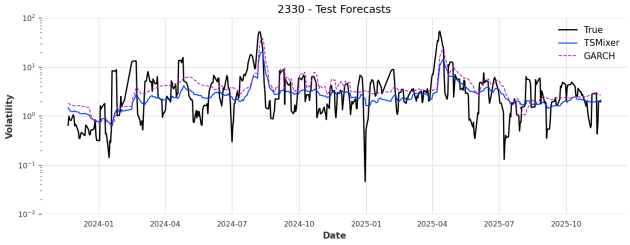

Figure 5: Out-of-sample volatility forecasts for TSMC (stock ID 2330).

Table 5 reports the out-of-sample performance of baseline models trained with the mixed loss function. The variations of MAE, RMSE, and QLIKE across different values of λ are largely consistent with the patterns observed in previous experiments. Specifically, smaller values of λ tend to reduce RMSE and QLIKE, while slightly increasing MAE, indicating a similar bias–tail-risk trade-off.

Despite the absence of cross-sectional covariates, TSMixer continues to significantly outperform LSTM and achieves slightly better performance than the GARCH family across most evaluation metrics. These results suggest that the two-layer MLP architecture of TSMixer is effective for volatility forecasting even in a single-asset setting.

6.4 Log-Scale versus Original-Scale Prediction

As discussed in Section 4.2, log-transformed volatility targets exhibit more favorable statistical properties. To explicitly compare the learning behavior under different target scales, we retrain TSMixer using both log-volatility and original-scale volatility targets and examine their respective performances.

Table 6 reports the out-of-sample performance of TSMixer under log-scale and non-log-scale volatility targets. Overall, TSMixer trained on log-volatility consistently outperforms its non-log counterpart across most values of λ. However, as λ increases and the model becomes less reliant on the GARCH family, the log-scale TSMixer exhibits progressively worse performance in predicting extreme values, as reflected by deteriorating tail-sensitive metrics. This suggests that log transformation, while improving average predictive accuracy, reduces the model's ability to capture extreme volatility realizations.

Importantly, this limitation can be effectively mitigated by incorporating the mixed loss function. When the GARCH-based component is present, the log-scale TSMixer benefits from the statistical stability of log transformation while retaining robustness against extreme observations. These results highlight a trade-off between bias reduction and tail sensitivity when choosing the volatility target scale, and demonstrate that the proposed mixed loss provides a principled mechanism to balance this trade-off.

7 Discussion

7.1 Deprecated Models

We initially experimented with DeepAR, incorporating alpha factors as dynamic covariates. However, this architecture was subsequently deprecated in favor of TSMixer due to two fundamental structural limitations.

First, as an autoregressive recurrent model, DeepAR generates multi-step forecasts via recursive one-step-ahead predictions, which is inherently prone to error accumulation over extended forecast horizons. In contrast, TSMixer adopts a direct multi-step forecasting strategy, mapping a long historical window to the entire prediction horizon in a single forward pass, thereby mitigating such propagation drift.

Second, and more critically, DeepAR operates as a global univariate model [2]. While it learns shared parameters across many

Table 5: Out-of-sample performance on the market index under the mixed loss function.

Model	Metric	λ=0	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95	1
GARCH	MAE							2.38						
	RMSE							7.64						
	QLIKE							1.57						
GJR-GARCH	MAE							2.45						
	RMSE							7.60						
	QLIKE							1.59						
TGARCH	MAE							2.31						
	RMSE							7.17						
	QLIKE							1.58						
LSTM	MAE	2.27	2.24	2.19	2.18	2.17	2.29	2.25	2.16	2.12	2.19	2.09	2.17	2.19
	RMSE	7.77	7.79	7.79	7.84	7.89	8.08	8.03	8.03	8.10	8.20	7.71	7.78	7.73
	QLIKE	1.55	1.56	1.58	1.61	1.61	1.65	1.61	1.85	1.98	2.24	1.99	2.38	2.66
TSMixer	MAE	2.26	2.23	2.19	2.15	2.11	2.09	2.10	2.10	2.08	2.07	2.05	2.04	2.03
	RMSE	7.57	7.56	7.52	7.61	7.74	7.86	7.92	8.00	8.06	8.10	8.12	8.13	8.14
	QLIKE	1.49	1.50	1.51	1.52	1.54	1.56	1.56	1.62	1.71	1.80	1.90	1.95	2.01

Table 6: Comparison between log-scale and original-scale volatility targets for TSMixer.

Model	Metric	λ=0	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95	1
GARCH	MAE							5.18						
	RMSE							11.30						
	QLIKE							2.28						
GJR-GARCH	MAE							5.27						
	RMSE							11.69						
	QLIKE							2.29						
TGARCH	MAE							5.25						
	RMSE							11.46						
	QLIKE							2.28						
TSMixer (log)	MAE	4.95	4.89	4.83	4.74	4.65	4.58	4.52	4.46	4.42	4.38	4.35	4.34	4.33
	RMSE	11.16	11.16	11.17	11.21	11.29	11.35	11.42	11.49	11.56	11.63	11.69	11.73	11.75
	QLIKE	2.20	2.20	2.21	2.22	2.24	2.27	2.30	2.35	2.40	2.46	2.54	2.59	2.64
TSMixer (no-log)	MAE	5.17	5.17	5.20	5.19	5.15	5.13	5.10	5.08	5.07	5.04	5.03	5.03	5.02
	RMSE	11.13	11.13	11.14	11.15	11.15	11.16	11.17	11.19	11.19	11.21	11.22	11.23	11.24
	QLIKE	2.35	2.27	2.30	2.25	2.27	2.24	2.31	2.25	2.27	2.55	2.32	2.31	2.54

related time series, it treats each target series independently at inference time and does not explicitly model instantaneous interactions between different alpha signals. TSMixer overcomes this limitation through its channel-mixing (feature-mixing) MLP layers, which capture rich cross-sectional dependencies and non-linear correlations among multivariate alpha factors.

7.2 Deprecated Methods

Lakshminarayanan et al. [22] show that retraining a model multiple times with the same hyperparameter configuration but different random seeds, and subsequently aggregating their predictions, leads to consistent performance improvements over a single trained model. This approach, commonly referred to as *Deep Ensemble*, has been shown to improve both predictive accuracy and uncertainty estimation.

An alternative ensemble strategy is *Parameter Ensemble*, which retrains multiple models using the top-K hyperparameter configurations identified during the search process and aggregates their predictions. In practice, ParamsEnsemble often yields even more stable and less biased forecasts than Deep Ensemble, as it captures model diversity arising from both optimization randomness and hyperparameter variation.

In our study, both Deep Ensemble and Params Ensemble were implemented and evaluated in preliminary experiments. However, due to the large number of models and experimental configurations considered, fully incorporating these ensemble methods into all reported results would require substantially greater computational resources. As a result, the empirical results presented in this paper are based on single-model training for each configuration.

Despite this limitation, our preliminary experiments consistently indicate that both Deep Ensemble and Params Ensemble significantly outperform single non-ensembled models. These findings suggest that incorporating ensemble strategies in practical forecasting can further enhance the performance of machine learning models, potentially widening the performance gap between modern learning-based approaches and traditional linear econometric models.

7.3 Log-Volatility Targets and the Role of λ

Based on the experimental results presented above, we find that training machine learning models using log-transformed volatility targets consistently reduces average forecast bias. In particular, log-volatility enables models to better capture the overall direction and dynamics of future volatility. However, this improvement comes at the cost of inferior performance in predicting extreme observations, as log transformation tends to compress large volatility realizations.

Combining log-volatility targets with the proposed mixed loss function allows us to mitigate the limitations of both approaches. A plausible interpretation is that the loss function implicitly balances log-scale and original-scale learning through the weighting parameter λ. By adjusting λ, the model is able to jointly optimize average forecast accuracy and robustness to extreme values. This suggests that the mixed loss function effectively enables the model to learn complementary aspects of volatility dynamics from both log-transformed and level-scale information, rather than relying exclusively on a single representation.

7.4 Is the GARCH Family Necessary?

GARCH-type models primarily forecast future volatility through the persistence and autocorrelation structure of past volatility. As a result, they are particularly effective at capturing volatility clustering and extreme observations. However, in practical applications, volatility forecasting does not always require precise prediction of extreme events. In many settings, the ability to anticipate the direction and trend of volatility in advance may be more valuable than accurately reacting after large shocks occur.

Moreover, GARCH models are generally better suited for short-horizon forecasting. As the forecasting horizon increases, GARCH models often exhibit excessive responses, overestimating volatility following large shocks and underestimating volatility during calm periods. In contrast, machine learning models tend to provide more stable forecasts with lower average prediction error over longer horizons.

Taken together, these observations suggest that GARCH models and machine learning models are better viewed as complementary rather than competing approaches. GARCH models are more effective for short-horizon forecasting and tail-event sensitivity, while machine learning models excel at longer-horizon, trend-oriented, and stable volatility prediction.

7.5 Shared Parameter Structures

TSMixer natively supports parameter sharing across assets, which we believe is particularly beneficial in financial markets characterized by high noise, limited data, and non-stationarity. Parameter sharing allows the model to pool information across assets and reduce estimation variance. At the same time, we acknowledge that individual stocks may exhibit distinct volatility cycles that cannot be fully captured by a single shared representation.

The design of the mixed loss function may provide a partial resolution to this tension between shared and asset-specific dynamics. GARCH forecasts are estimated separately for each individual asset, while TSMixer relies on shared parameters across assets. The interaction between these two components effectively introduces a balance between global and idiosyncratic information. This perspective may also explain why $\lambda = 1$, which corresponds to ignoring the GARCH component entirely, is rarely optimal in our experiments.

8 Conclusion

This paper builds on the GARCH-informed learning framework of Xu et al. [31] and examines its effectiveness under a modern all-MLP architecture and a large-scale multi-asset forecasting setting. Building on their findings, we show that incorporating GARCH-based information through the training objective, when combined with an appropriate architecture and a carefully chosen mixing parameter, can achieve predictive performance that, for appropriately chosen configurations, exceeds that of standalone GARCH models across multiple evaluation dimensions.

By replacing recurrent architectures such as LSTM with TSMixer and extending the task from market-index forecasting to joint volatility prediction across multiple individual stocks under a shared-parameter architecture, we demonstrate that deep learning models can exploit nonlinear dynamics and cross-sectional information that are difficult to capture within standard univariate GARCH-type specifications without substantial parametric expansion. Importantly, our results indicate that these gains do not arise from naive inclusion of additional covariates, but from the interaction between model architecture and the GARCH-informed training objective.

A central finding of this study concerns the role of the mixing parameter λ. Rather than serving as a technical tuning constant, λ determines how strongly the model is mixed toward GARCH-based reference forecasts during training. For suitably chosen values of λ, the GARCH-informed TSMixer consistently outperforms the GARCH family across both average-error metrics and tail-sensitive measures, in both multi-stock and market-index settings. This result establishes that the mixed-loss formulation enables deep learning models not only to inherit the tail robustness of GARCH, but also to surpass its predictive limitations through nonlinear representation learning.

We further investigate the role of auxiliary alpha factors in volatility prediction. Our empirical evidence suggests that, when used without explicit screening or structural constraints, high-dimensional alpha signals contribute limited incremental information to volatility forecasting and appear to be weakly informative relative to the dominant volatility dynamics and are easily overwhelmed by noise. This finding indicates that the superior performance of the proposed framework is primarily driven by the interaction between the GARCH-informed loss function and the TSMixer architecture, rather than by naive inclusion of high-dimensional factor inputs. Effective use of alpha signals for volatility forecasting therefore likely requires feature selection, dimensionality reduction, or economically motivated filtering.

Overall, our results suggest a clear hierarchy in volatility modeling. Standalone GARCH models provide a strong but incomplete description of volatility dynamics, while purely data-driven neural networks may struggle with tail robustness. By embedding GARCH forecasts into the learning objective and selecting an appropriate mixing parameter λ, GARCH-informed deep learning models are able to combine the strengths of both approaches, achieving superior and more robust predictive performance across a range of evaluation metrics. The shared-parameter design further allows the model to pool information across assets, mitigating idiosyncratic noise while preserving asset-specific dynamics through the GARCH-informed loss. These findings highlight the effectiveness of incorporating econometric structure into modern neural architectures for large-scale volatility forecasting and risk management applications.

Future research may extend this framework by further refining the design of the training objective to better balance predictive performance under normal and extreme market conditions. While the proposed mixed-loss formulation improves average accuracy and tail-sensitive metrics relative to purely data-driven models, its performance under squared-error criteria such as MSE occasionally lags behind that of GARCH benchmarks. This pattern points to a trade-off between average calibration and sensitivity to high-volatility regimes, rather than a uniform deficiency in tail prediction. Developing regime-aware, asymmetric, or tail-weighted loss functions—such as variants of QLIKE or hybrid objectives that explicitly differentiate between normal and extreme volatility states—offers a promising direction for enhancing the robustness of GARCH-informed deep learning models in practical risk management settings.

Code Availability

The code used in this paper is publicly available at: https://github.com/hochienH/GLoVE. Due to data licensing restrictions, the raw data cannot be shared, but all preprocessing and modeling scripts are provided.

References

[1] Yigit Atilgan. 2014. Volatility spreads and earnings announcement returns. *Journal of Banking & Finance* 38 (Jan. 2014), 205–215. https://doi.org/10.1016/j.jbankfin.2013.10.007

[2] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, François-Xavier Aubet, Laurent Callot, and Tim Januschowski. 2022. Deep Learning for Time Series Forecasting: Tutorial and Literature Survey. *ACM Comput. Surv.* 55, 6, Article 121 (Dec. 2022), 36 pages. https://doi.org/10.1145/3533382

[3] Tim Bollerslev. 1986. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* 31, 3 (1986), 307–327. https://doi.org/10.1016/0304-4076(86)90063-1

[4] Massimiliano Caporin and Michele Costola. 2019. Asymmetry and leverage in GARCH models: a News Impact Curve perspective. *Applied Economics* 51, 31 (July 2019), 3345–3364. https://doi.org/10.1080/00036846.2019.1578853 Publisher: Routledge _eprint: https://doi.org/10.1080/00036846.2019.1578853.

[5] Qizhao Chen and Hiroaki Kawashima. 2025. Sentiment-Aware Stock Price Prediction with Transformer and LLM-Generated Formulaic Alpha. https://doi.org/10.48550/arXiv.2508.04975 arXiv:2508.04975 [cs].

[6] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. 2023. TSMixer: An All-MLP Architecture for Time Series Forecasting. https://doi.org/10.48550/arXiv.2303.06053 arXiv:2303.06053 [cs].

[7] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). Association for Computational Linguistics, Doha, Qatar, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[8] Christian Conrad and Melanie Schienle. 2020. Testing for an Omitted Multiplicative Long-Term Component in GARCH Models. *Journal of Business & Economic Statistics* 38, 2 (April 2020), 229–242. https://doi.org/10.1080/07350015.2018.1482759

[9] Zhuanxin Ding, Clive W. J. Granger, and Robert F. Engle. 1993. A long memory property of stock market returns and a new model. *Journal of Empirical Finance* 1, 1 (June 1993), 83–106. https://doi.org/10.1016/0927-5398(93)90006-D

[10] M. N. Dol. 2021. *Comparison of the GARCH, EGARCH, GJR-GARCH and TGARCH model in times of crisis for the S&P500, NASDAQ and Dow-Jones.* Ph. D. Dissertation. https://thesis.eur.nl/pub/59759

[11] Thomas Fischer and Christopher Krauss. 2018. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research* 270, 2 (Oct. 2018), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

[12] Kenneth R. French and Richard Roll. 1986. Stock return variances: The arrival of information and the reaction of traders. *Journal of Financial Economics* 17, 1 (Sept. 1986), 5–26. https://doi.org/10.1016/0304-405X(86)90004-8

[13] Lawrence R. Glosten, Ravi Jagannathan, and David E. Runkle. 1993. On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. *The Journal of Finance* 48, 5 (1993), 1779–1801. https://doi.org/10.1111/j.1540-6261.1993.tb05128.x _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1993.tb05128.x.

[14] Shihao Gu, Bryan Kelly, and Dacheng Xiu. 2020. Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies* 33, 5 (Feb. 2020), 2223–2273. https://doi.org/10.1093/rfs/hhaa009

[15] Zhongfang He and John M. Maheu. 2010. Real time detection of structural breaks in GARCH models. *Computational Statistics & Data Analysis* 54, 11 (Nov. 2010), 2628–2640. https://doi.org/10.1016/j.csda.2009.09.038

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[17] Zura Kakushadze. 2015. 101 Formulaic Alphas. https://doi.org/10.2139/ssrn.2701346

[18] Jonathan M. Karpoff. 1987. The Relation Between Price Changes and Trading Volume: A Survey. *The Journal of Financial and Quantitative Analysis* 22, 1 (1987), 109–126. https://doi.org/10.2307/2330874 Publisher: [Cambridge University Press, University of Washington School of Business Administration].

[19] Bryan T. Kelly, Boris Kuznetsov, Semyon Malamud, and Teng Andrea Xu. 2023. Deep Learning from Implied Volatility Surfaces. https://doi.org/10.2139/ssrn.4531181

[20] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2021. Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift. In *International Conference on Learning Representations*. https://openreview.net/forum?id=cGDAkQo1C0p

[21] Franc J.G.M. Klaassen. 1998. Improving Garch Volatility Forecasts. *SSRN Electronic Journal* (1998). https://doi.org/10.2139/ssrn.138094

[22] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6405–6416.

[23] Christopher G. Lamoureux and William D. Lastrapes. 1990. Heteroskedasticity in Stock Return Data: Volume versus GARCH Effects. *The Journal of Finance* 45, 1 (1990), 221–229. https://doi.org/10.2307/2328817 Publisher: [American Finance Association, Wiley].

[24] R. David Mclean and Jeffrey Pontiff. 2016. Does Academic Research Destroy Stock Return Predictability? *The Journal of Finance* 71, 1 (2016), 5–32. https://doi.org/10.1111/jofi.12365 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jofi.12365.

[25] Daniel B. Nelson. 1991. Conditional Heteroskedasticity in Asset Returns: A New Approach. *Econometrica* 59, 2 (1991), 347–370. https://doi.org/10.2307/2938260 Publisher: [Wiley, Econometric Society].

[26] Olle Norberg and Vincent Boukov. 2024. The relationship between trading volume, market capitalization, and volatility : Normal versus abnormal market conditions. Backup Publisher: Linnaeus University, Department of Management (MAN) Pages: 57.

[27] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. https://doi.org/10.48550/arXiv.1905.10437 arXiv:1905.10437 [cs].

[28] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (July 2020), 1181–1191. https://doi.org/10.1016/j.ijforecast.2019.07.001

[29] Hugo Gobato Souto, Storm Koert Heuvel, and Francisco Louzada Neto. 2024. Time-mixing and feature-mixing modelling for realized volatility forecast: Evidence from TSMixer model. *The Journal of Finance and Data Science* 10 (Dec. 2024), 100143. https://doi.org/10.1016/j.jfds.2024.100143

[30] Taiwan Economic Journal. 2024. TQuant Lab. https://tquant.tejwin.com/ Accessed: 2025-12-21.

[31] Zeda Xu, John Liechty, Sebastian Benthall, Nicholas Skar-Gislinge, and Christopher McComb. 2024. GARCH-Informed Neural Networks for Volatility Prediction in Financial Markets. In *Proceedings of the 5th ACM International Conference on AI in Finance (ICAIF '24)*. Association for Computing Machinery, New York, NY, USA, 600–607. https://doi.org/10.1145/3677052.3698600

[32] Jean-Michel Zakoian. 1994. Threshold heteroskedastic models. *Journal of Economic Dynamics and Control* 18, 5 (Sept. 1994), 931–955. https://doi.org/10.1016/0165-1889(94)90039-6