Trading Volume Alpha with Volatility-Aware Execution

15.S06 Project Report

December 12, 2025

Abstract

We replicate the Trading Volume Alpha (TVA) framework of Goyenko, Kelly, Moskowitz, Su and Zhang (2025) and extend it to incorporate volatility into the execution decision. Using CRSP, Compustat, OptionMetrics and a large cross-section of firm characteristics, we forecast daily log volume surprises with linear (Ridge) and recurrent neural network (RNN-LSTM) models and embed these forecasts into an economic loss function derived from a tracking error vs. price impact trade-off. For large, liquid stocks, economic fine-tuning of RNNs improves both statistical fit and Mean Economic Loss (MEL), while for small caps simple linear models dominate and economic fine-tuning can be harmful. We then allow the impact coefficient to scale with a volatility surprise and evaluate this in a "True World vs. Agent World" experiment. Volatility-aware execution has modest average benefits for large caps but acts as a circuit breaker in illiquid names, reducing economic loss by roughly 30–40% in high-volatility regimes relative to volume-only RNNs.

1 Introduction

Standard asset pricing models typically assume frictionless markets, yet for institutional investors managing substantial capital, transaction costs constitute a first-order determinant of realized performance. Goyenko et al. (2025, henceforth TVA) confront this limitation directly by introducing the concept of trading volume alpha. This provides a framework demonstrating that portfolio performance depends not only on the ability to forecast returns, but equally on the capacity to forecast trading volume and thereby optimize execution. Central to their contribution is the replacement of standard mean squared error with an economic loss function grounded in the fundamental trade-off between price impact and tracking error. This innovation ensures that volume forecasts are judged by the economic value they actually deliver in portfolio implementation, rather than by statistical criteria divorced from investment outcomes. The theoretical foundation of the TVA framework lies in recognizing that volume forecast errors carry asymmetric economic consequences. Over-estimating liquidity exacts a severe penalty: it induces aggressive trading that generates substantial price impact, eroding returns in a manner that cannot be recovered. Under-estimating liquidity, by contrast, leads the investor to trade too cautiously, producing only bounded tracking error relative to the target portfolio. Mean squared error remains agnostic to this asymmetry, treating over-prediction and under-prediction as equivalent failures. The TVA economic loss function corrects this deficiency by penalizing errors in direct proportion to their adverse implementation consequences, systematically favoring models that guard against the more damaging regime of over-trading.

The baseline TVA model posits that liquidity scales inversely with volume, an assumption that, while tractable, abstracts from the well-documented dependence of liquidity on volatility. During periods of market stress, volume and volatility often spike in tandem—panic selling generates elevated turnover precisely when spreads widen and depth evaporates. A model relying exclusively on volume would misinterpret these episodes as liquidity opportunities, potentially triggering aggressive execution at the worst possible moment. We extend the TVA framework by augmenting the price impact specification with a volatility surprise term, explicitly conditioning execution decisions on the joint behavior of volume and volatility. This extension is designed to arrest aggressive trading when superficially high volume conceals deteriorating market conditions.

Our investigation pursues three objectives. First, we implement the TVA framework across a broad cross-section of US equities to replicate the original findings concerning volume predictability and its economic value. Second, we conduct a systematic methodological comparison between linear models estimated via Ridge regression and non-linear architectures based on recurrent neural networks with LSTM cells, evaluating each with and without economic fine-tuning across equity universes exhibiting distinct liquidity characteristics. Third, we embed volatility surprises into the execution logic and rigorously quantify the incremental benefits of volatility-aware trading through a controlled experimental design contrasting the "True World" against the "Agent World." The remainder of the paper develops the theoretical framework, describes the data, details the model architectures and experimental protocol, and presents empirical results alongside a discussion of their broader implications.

2 Theoretical Framework and Economic Loss

This section adapts the TVA portfolio framework to our implementation, deriving a tractable loss function that incorporates volatility surprises into the execution decision.

We model price impact as a function of the participation rate, defined as the ratio of investor trade size to total daily dollar volume. Following the market microstructure literature, impact costs are specified as quadratic in trade size and inversely proportional to volume. This functional form reflects the intuition that larger trades move prices more than proportionally, while deeper markets absorb order flow with less disturbance. Absorbing investor-specific elasticities into a reduced-form parameter, we express the cost function as

$$\text{ImpactCost}_i \propto \frac{1}{2}\lambda_i z_i^2, \tag{1}$$

where $z_i \in [0, 1]$ denotes the fraction of the desired trade executed on a given day. The impact coefficient λ_i captures the liquidity environment facing security i and is modeled as

$$\lambda_i = \lambda_0 e^{-v_i}, \tag{2}$$

where $v_i = \log(\$\text{Volume}_{i,t})$ represents log dollar volume and $\lambda_0 = 0.2$ serves as a scaling constant. This exponential specification ensures that impact costs decline smoothly as volume increases, consistent with the empirical regularity that deeper markets offer superior execution quality.

The investor's problem consists of closing a position gap q_i by selecting an optimal trading rate z_i. This decision involves a fundamental trade-off: executing aggressively incurs immediate price impact, while trading patiently exposes the portfolio to tracking error as it deviates from the target allocation. We formalize the cost of delayed execution as

$$\text{TrackingErrorCost}_i \propto \frac{1}{2}\mu(1 - z_i)^2, \tag{3}$$

where the parameter μ encapsulates both risk aversion and the rate at which the trading signal decays. Combining the impact and tracking error components yields the TVA economic loss function:

$$\ell_{\text{econ}}(z_i; \lambda_i, \mu) = \lambda_i z_i^2 + \mu(1 - z_i)^2. \tag{4}$$

This objective function crystallizes the central tension in optimal execution: the first term penalizes aggressive trading through price impact, while the second term penalizes passive trading through accumulated tracking error.

Minimizing the economic loss with respect to the trading rate yields the optimal policy

$$z_i^\star = \frac{\mu}{\lambda_i + \mu}. \tag{5}$$

Substituting the exponential specification for the impact coefficient, we obtain a sigmoid function of log-volume:

$$z_i^\star(v_i) = \frac{1}{1 + \exp\left[-(v_i - v^\star)\right]}, \tag{6}$$

where the offset $v^\star = \ln \lambda_0 - \ln \mu$ determines the volume threshold at which the investor transitions from cautious to aggressive execution. The sigmoid structure admits transparent interpretation at the extremes. When volume is low relative to the threshold ($v_i \ll v^\star$), the optimal trading

rate collapses toward zero, instructing the investor to withhold the trade and avoid adverse price impact. When volume is high ($v_i \gg v^\star$), the optimal rate approaches unity, permitting full execution with minimal market footprint. The intermediate regime, where the sigmoid exhibits maximum curvature, represents the zone in which forecast accuracy matters most for economic outcomes.

The baseline framework assumes that volume alone determines liquidity, yet this abstraction neglects the well-established covariation between liquidity and volatility. To incorporate market turbulence, we introduce a volatility surprise term $s_{i,t}$, defined as the percentage deviation of realized volatility from its 60-day moving average. We augment the impact coefficient as follows:

$$\tilde{\lambda}_{i,t} = \lambda_0 e^{-v_{i,t}} \cdot \max\{0.1, 1 + \beta s_{i,t}\}. \tag{7}$$

When realized volatility exceeds its recent average ($s_{i,t} > 0$), the effective impact coefficient rises, shifting the optimal trading sigmoid rightward and inducing more conservative execution. This mechanism operates as a circuit breaker: even if volume appears elevated, a concurrent volatility spike signals deteriorating liquidity conditions and triggers a reduction in trading intensity. The floor at 0.1 prevents the adjustment factor from turning negative or excessively small during periods of unusually subdued volatility.

The model is trained using a convex combination of statistical and economic objectives. Letting $\hat{v}_{i,t}$ denote the forecasted log-volume and $v_{i,t}$ the realized value, the economic loss evaluates the cost incurred when the agent trades according to the forecast in a market governed by realized conditions:

$$\ell_{\text{econ}}(\hat{v}_{i,t}; v_{i,t}) = \lambda(v_{i,t})[z(\hat{v}_{i,t})]^2 + \mu[1 - z(\hat{v}_{i,t})]^2. \tag{8}$$

Training proceeds in two stages. The pre-training phase minimizes mean squared error to extract statistical regularities from the volume time series, establishing a foundation of predictive accuracy. The fine-tuning phase then optimizes a weighted loss $\ell_{\text{total}} = \alpha \ell_{\text{MSE}} + (1 - \alpha)\tilde{\ell}_{\text{econ}}$, where $\tilde{\ell}_{\text{econ}}$ is the normalized economic loss. This two-stage procedure implicitly reweights the data according to economic materiality, concentrating model capacity on the intermediate volume regime where the trading rate exhibits greatest sensitivity to forecast errors and where accurate prediction therefore delivers the largest gains in implementation performance.

3 Data, Universes and Target Construction

3.1 Data Sources

We construct a daily panel of United States common stocks spanning the five-year period from January 2018 to December 2022. The primary market data, including daily prices, returns, shares outstanding, and trading volume (in shares), are sourced from the Center for Research in Security Prices (CRSP) Daily Stock File, complemented by the CRSP value-weighted market index. Fundamental accounting variables for size, value, and quality characteristics are retrieved from Compustat, while volatility measures—utilized to construct realized volatility and volatility surprises—are obtained from OptionMetrics. To support a robust feature set, we integrate a global characteristic library comprising 153 firm-level predictors that cover momentum, volatility, liquidity, profitability, and other relevant dimensions. The sample is restricted to ordinary common shares (CRSP share codes 10 and 11; exchange codes 1, 2, and 3) that maintain a minimum history of 252 trading days. To ensure robustness against data errors or transient market anomalies, extreme outliers in both volume and returns are winsorized.

3.2 Sample Split

To mirror the out-of-sample evaluation protocol established in Goyenko et al. (2025), we adopt a strictly time-based partitioning scheme. The dataset is divided into a training set covering the period from January 2018 through December 2020, a validation set spanning January 2021 to December 2021, and a test set for the final year, January 2022 to December 2022. All model hyperparameters are selected based on validation set performance, while all empirical results reported in this study are derived exclusively from the held-out test set to prevent look-ahead bias.

3.3 Universe Construction

We analyze three disjoint investment universes, each consisting of 200 stocks, to examine the behavior of the TVA framework across the liquidity spectrum. The *Largest* universe comprises the 200 stocks with the highest median market capitalization over the sample period, representative of an S&P 500-style large-cap segment. Conversely, the *Smallest* universe contains the 200 smallest stocks from the eligible pool, subject to liquidity constraints including a minimum of 252 trading days, a median daily dollar volume exceeding $100,000, and the availability of key characteristics. Finally, the *Random* universe consists of 200 names drawn uniformly from the top 4,000 stocks by market capitalization. This design establishes a clear liquidity hierarchy: the Largest universe proxies for highly liquid, institutional-grade assets, the Smallest universe represents a challenging micro-cap implementation environment, and the Random universe provides a mixed-liquidity baseline.

3.4 Log Volume and Residualization

Let $\mathrm{Vol}_{i,t}$ denote the daily dollar volume for stock i on day t. We define the log volume as $v_{i,t} = \log \mathrm{Vol}_{i,t}$. Given that raw log volume exhibits high persistence and trending behavior, simple moving averages capture a significant portion of its dynamics. To isolate the unpredictable component and focus model learning on volume *surprises*, we adopt a residual-learning framework. Specifically, we calculate a 5-day moving average,

$$\mathrm{MA5}_{i,t} = \frac{1}{5} \sum_{k=1}^{5} v_{i,t-k}, \tag{9}$$

and define the target variable $y_{i,t}$ as the deviation from this trend:

$$y_{i,t} = v_{i,t} - \mathrm{MA5}_{i,t}. \tag{10}$$

Our predictive models are trained to forecast the residual $y_{i,t}$. The full volume forecast is subsequently reconstructed as

$$\hat{v}_{i,t} = \mathrm{MA5}_{i,t} + \hat{y}_{i,t}.$$

Empirical tests indicate that this residualization strategy reduces Mean Squared Error (MSE) by approximately 10–20% and stabilizes the training process, as the models are relieved of the burden of learning deterministic level dynamics.

4 Model Architectures

This section details the forecasting models employed in our analysis, ranging from simple linear baselines to recurrent neural networks with economic fine-tuning and volatility-aware execution overlays.

4.1 Linear Baseline Models

Our baseline models are deliberately transparent, providing interpretable benchmarks against which to evaluate more complex architectures.

The simplest benchmark is the five-day moving average (MA5), which predicts log volume as the trailing average of the five most recent observations: $\hat{v}_{i,t}^{\text{MA5}} = \text{MA5}_{i,t}$. This naïve forecast captures the strong persistence in volume and explains approximately 94% of the unconditional variance in log dollar volume. All other models are evaluated relative to MA5, with R^2 defined as the percentage reduction in mean squared error.

Our first substantive model is Ridge regression with technical features (olstech). We regress the volume residual $y_{i,t} = v_{i,t} - \text{MA5}_{i,t}$ on the 13-dimensional technical feature vector:

$$\hat{y}_{i,t} = X_{i,t}^{\text{tech}}\beta + \varepsilon_{i,t}, \tag{11}$$

with ℓ_2 regularization penalty $\alpha = 1.0$. The technical features include lagged volume levels (5-day and 20-day moving averages), return-based predictors (contemporaneous return, absolute return, squared return, and 5-day and 20-day return momentum), volatility proxies (moving averages of absolute returns), the market return, and calendar effects (day-of-week dummies and a month-end indicator).

The second linear specification is Ridge regression with all features (olsall), which augments the technical feature set with the top 50 cross-sectional characteristics selected from the Jensen, Kelly, and Pedersen (2023) factor zoo. These characteristics span size (market equity), value (book-to-market, earnings-to-price), momentum (12-month returns excluding the most recent month), volatility (idiosyncratic volatility), liquidity (turnover, bid-ask spread), and quality (return on equity). Feature selection is performed via correlation ranking with the target variable on the training set.

These OLS-based models are simple, computationally efficient, and are surprisingly competitive with more complex alternatives, particularly in illiquid or heterogeneous universes where signal-to-noise ratios are low.

4.2 Feedforward Neural Network Architecture

Our first non-linear model is a feedforward neural network (FFNet), which serves as an intermediate step between linear regression and recurrent architectures. The FFNet can capture non-linear relationships in the feature space but processes each observation independently, without explicitly modeling temporal dependencies across the lookback window.

The architecture is a four-layer multilayer perceptron with progressively decreasing hidden dimensions. Given an input feature vector $X_{i,t} \in \mathbb{R}^d$, the network computes:

$$\hat{y}_{i,t} = W_4 \cdot \text{ReLU}\left(W_3 \cdot \text{ReLU}\left(W_2 \cdot \text{ReLU}(W_1 X_{i,t} + b_1) + b_2\right) + b_3\right) + b_4, \tag{12}$$

where $W_1 \in \mathbb{R}^{32\times d}$, $W_2 \in \mathbb{R}^{16\times 32}$, $W_3 \in \mathbb{R}^{8\times 16}$, and $W_4 \in \mathbb{R}^{1\times 8}$. The network contains three hidden layers with 32, 16, and 8 units respectively, each followed by ReLU activation, and a final linear layer

producing the scalar prediction. Unlike the RNN architecture described below, the FFNet does not employ dropout regularization; the relatively small number of parameters (approximately 1,500 for technical features, 3,500 for all features) provides implicit regularization against overfitting.

The FFNet's simplicity offers two advantages. First, it is computationally efficient, requiring only a single forward pass through four matrix multiplications. Second, it provides a clean test of whether non-linear transformations of contemporaneous features add value beyond what linear models capture, separate from the question of whether temporal dynamics matter.

4.3 Recurrent Neural Network Architecture

Our primary non-linear models are based on Long Short-Term Memory (LSTM) networks, which are well-suited to capturing temporal dependencies in sequential data. The architecture proceeds through three stages: sequence encoding, temporal summarization, and prediction.

The input to the network is a sequence of feature vectors $\{X_{i,t-L+1}, \ldots, X_{i,t}\}$ spanning $L = 10$ trading days. Each feature vector has dimension d, where $d = 13$ for technical-only models and $d = 63$ for models incorporating cross-sectional characteristics. Features are standardized using rolling means and standard deviations computed on the training set to prevent lookahead bias.

The sequence encoder consists of two stacked LSTM layers. The first LSTM layer processes the input sequence and produces a sequence of hidden states:

$$h_t^{(1)} = \text{LSTM}^{(1)}(X_t, h_{t-1}^{(1)}, c_{t-1}^{(1)}), \tag{13}$$

where $h_t^{(1)} \in \mathbb{R}^{32}$ is the hidden state and $c_t^{(1)} \in \mathbb{R}^{32}$ is the cell state. The second LSTM layer takes the hidden states from the first layer as input:

$$h_t^{(2)} = \text{LSTM}^{(2)}(h_t^{(1)}, h_{t-1}^{(2)}, c_{t-1}^{(2)}), \tag{14}$$

producing a final hidden state $h_L^{(2)} \in \mathbb{R}^{32}$ that summarizes the entire input sequence. Both LSTM layers employ dropout with probability 0.3 applied to the non-recurrent connections, providing regularization against overfitting.

The prediction head transforms the final hidden state into a scalar volume forecast through a sequence of fully-connected layers with progressively decreasing dimensionality:

$$\hat{y}_{i,t} = W_4 \cdot \text{ReLU}\left(W_3 \cdot \text{ReLU}\left(W_2 \cdot \text{ReLU}(W_1 h_L^{(2)} + b_1) + b_2\right) + b_3\right) + b_4, \tag{15}$$

where $W_1 \in \mathbb{R}^{16\times32}$, $W_2 \in \mathbb{R}^{8\times16}$, $W_3 \in \mathbb{R}^{4\times8}$, and $W_4 \in \mathbb{R}^{1\times4}$. Dropout with probability 0.3 is applied after the first two ReLU activations. This tapering architecture—from 32 dimensions down through 16, 8, and 4 to the final scalar output—encourages the network to learn increasingly abstract representations of the input sequence before producing the forecast.

The complete architecture contains approximately 15,000 trainable parameters for technical-only models and 25,000 parameters for full-feature models. Figure 1 provides a schematic representation of the network structure.

4.4 Model Variants and Training Objectives

We estimate multiple variants of each neural network architecture, differing in feature sets and training objectives.

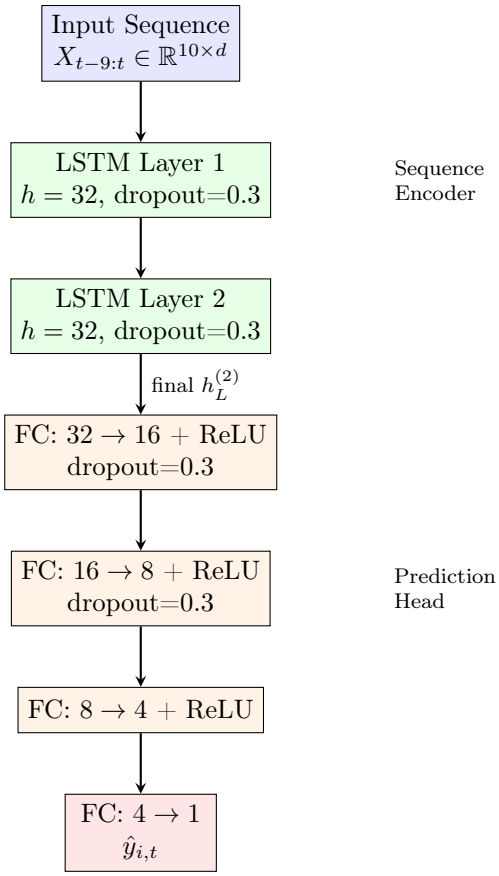

Figure 1: RNN-LSTM architecture for volume forecasting. The network processes 10 days of lagged features through two stacked LSTM layers, then maps the final hidden state to a scalar prediction via a tapering fully-connected head.

For the feedforward network, we estimate two variants trained on mean squared error: fftech uses only technical features ($d = 13$), while ffall incorporates both technical and cross-sectional features ($d = 63$). We also estimate economically fine-tuned variants ff.econtech and ff.econall using the combined loss function described below.

For the RNN-LSTM, we estimate analogous variants. The models rnntech and rnnall are trained purely on the statistical objective of minimizing mean squared error, with rnntech using only technical features and rnnall incorporating the full feature set. These models learn to predict volume residuals as accurately as possible without regard to the downstream trading application.

The economically fine-tuned variants incorporate the trading objective directly into training. The model rnn.econtech begins from the pre-trained weights of rnntech and continues training with the combined loss function $\ell_{\text{total}} = 0.99\,\ell_{\text{MSE}} + 0.01\,\tilde{\ell}_{\text{econ}}$. Similarly, rnn.econall is fine-tuned from rnnall. As discussed in Section 2, this procedure adapts predictions to the portfolio optimization problem by penalizing forecast errors according to their economic consequences rather than their statistical magnitude.

Table 1 summarizes the key characteristics of all forecasting models.

Table 1: Summary of Forecasting Models

Model	Architecture	Features	Training Loss	Parameters
MA5	Moving average	—	—	0
olstech	Ridge regression	Technical (13)	MSE	13
olsall	Ridge regression	All (63)	MSE	63
nntech	4-layer MLP	Technical (13)	MSE	∼1,500
nnall	4-layer MLP	All (63)	MSE	∼3,500
nn.econtech	4-layer MLP	Technical (13)	MSE + Econ	∼1,500
nn.econall	4-layer MLP	All (63)	MSE + Econ	∼3,500
rnntech	2-layer LSTM + FC head	Technical (13)	MSE	∼15,000
rnnall	2-layer LSTM + FC head	All (63)	MSE	∼25,000
rnn.econtech	2-layer LSTM + FC head	Technical (13)	MSE + Econ	∼15,000
rnn.econall	2-layer LSTM + FC head	All (63)	MSE + Econ	∼25,000

4.5 Volatility Forecasting and Surprise Construction

To implement volatility-aware execution, we require forecasts of realized volatility in addition to volume. We estimate separate models for log volatility using the same architectural choices as for volume: Ridge regression (ols.vol) and RNN-LSTM (rnn.vol) variants with technical and full feature sets.

The target variable for volatility forecasting is log realized volatility, computed as the logarithm of the standard deviation of intraday returns (where available) or the absolute value of daily returns (as a proxy). The volatility models produce a point forecast $\widehat{\log \sigma}_{i,t}$ for each stock-day.

The volatility surprise $s_{i,t}$ measures the deviation of current volatility from its recent historical level. We compute a 60-day trailing average of log volatility as the reference level:

$$\widehat{\log \sigma}^{\text{ref}}_{i,t} = \frac{1}{60} \sum_{k=1}^{60} \log \sigma_{i,t-k}. \tag{16}$$

The volatility surprise is then defined as the percentage deviation from this reference:

$$s_{i,t} = \frac{\sigma_{i,t} - \bar{\sigma}^{(60)}_{i,t}}{\bar{\sigma}^{(60)}_{i,t}}, \tag{17}$$

where $\bar{\sigma}^{(60)}_{i,t} = \exp(\widehat{\log \sigma}^{\text{ref}}_{i,t})$ is the reference volatility level. To prevent extreme values from destabilizing the impact adjustment, we clamp the surprise to the interval $[-2, 5]$: values below -2 (unusually calm conditions) are set to -2, and values above 5 (extreme stress) are set to 5.

4.6 Volatility-Adjusted Execution Agents

We construct volatility-adjusted agents by combining volume forecasts with volatility surprises through the augmented impact coefficient defined in equation (7). These agents differ from baseline TVA agents only through the impact adjustment and the implied trading rate; the underlying volume forecasts are identical.

The volatility-adjusted variants are denoted with the suffix `.vol_adj`. Thus, ols.vol_adj_tech uses Ridge volume predictions with technical features and applies the volatility adjustment, ff.vol_adj_all uses FFNet volume predictions with all features, and rnn.vol_adj_all uses RNN volume predictions with all features. In total, we evaluate twelve volatility-adjusted agents (three architectures $\times$ two feature sets $\times$ with/without economic fine-tuning).

The key distinction is operational rather than architectural. When computing the optimal trading rate, volatility-adjusted agents substitute the augmented impact coefficient $\tilde{\lambda}_{i,t} = \lambda_0 e^{-\hat{v}_{i,t}} \cdot \max\{0.1, 1 + \beta s_{i,t}\}$ for the baseline coefficient $\lambda_{i,t} = \lambda_0 e^{-\hat{v}_{i,t}}$. This shifts the trading-rate sigmoid rightward when volatility is elevated ($s_{i,t} > 0$), inducing more conservative execution, and leftward when volatility is subdued ($s_{i,t} < 0$), permitting more aggressive trading. The parameter $\beta = 1$ governs the sensitivity of the adjustment to volatility surprises.

This design allows us to isolate the incremental value of volatility awareness: by comparing ols.vol_adj_tech to olstech, or rnn.vol_adj_all to rnnall, we can measure precisely how much the volatility adjustment contributes to execution quality while holding the underlying volume forecast constant.

5 Training and Experimental Design

This section details the estimation procedures for our forecasting models and the construction of the trading experiment used to evaluate economic performance.

5.1 Pre-training of Neural Networks

Both feedforward and recurrent neural networks are first trained purely on the statistical objective of forecasting residual log volume. The target variable $y_{i,t} = v_{i,t} - \text{MA5}_{i,t}$ represents the deviation of realized log dollar volume from its trailing five-day moving average, ensuring that models learn to predict volume surprises rather than predictable baseline levels. The pre-training loss function is mean squared error:

$$\ell_{\text{MSE}} = \frac{1}{N} \sum_{i,t} (y_{i,t} - \hat{y}_{i,t})^2. \tag{18}$$

We employ the Adam optimizer with an initial learning rate of 10^{-3} and weight decay of 10^{-4}. A ReduceLROnPlateau scheduler reduces the learning rate by a factor of 0.5 after 5 epochs without improvement in validation MSE. Training terminates via early stopping with patience of 15 epochs, subject to a maximum of 150 epochs.

The two architectures exhibit different convergence characteristics reflecting their complexity. The feedforward networks (FFNet), with approximately 1,500–3,500 parameters, typically converge within 30–50 epochs and require minimal regularization beyond weight decay; their limited capacity provides implicit protection against overfitting. The recurrent networks (RNN-LSTM), with 15,000–25,000 parameters, require the full early stopping mechanism and dropout regularization to prevent memorization of training sequences. Despite these differences, both architectures are trained with identical hyperparameters to ensure fair comparison. This pre-training stage yields well-calibrated volume forecasters that serve as initialization for the subsequent economic fine-tuning phase.

5.2 Economic Fine-Tuning

Economically fine-tuned neural networks are obtained by continuing training from the pre-trained weights using a combined loss function that incorporates both statistical and economic objectives:

$$\ell_{\text{total}} = 0.99\, \ell_{\text{MSE}} + 0.01\, \tilde{\ell}_{\text{econ}}, \tag{19}$$

where $\tilde{\ell}_{\text{econ}}$ denotes the normalized economic loss. The heavy weighting toward MSE (99%) reflects a critical implementation detail: economic loss gradients exhibit substantially different magnitudes than MSE gradients and can destabilize training if weighted too aggressively. Ablation experiments confirmed that lower MSE weights (e.g., 0.5 or 0.9) resulted in prediction explosion, while the 0.99 weighting preserves learned statistical patterns while allowing the economic objective to refine predictions in economically material regions.

Fine-tuning employs a reduced learning rate of 10^{-4} to ensure stable adaptation to the new loss surface. This phase typically runs for 20–50 additional epochs. The economic loss is normalized by $(\lambda_0 + \mu + \epsilon)$ to ensure comparable gradient magnitudes across different AUM levels, where the tracking error penalty μ varies with investor size.

The fine-tuning procedure applies identically to both FFNet and RNN architectures, yielding the economically-tuned variants ff.econtech, ff.econall, rnn.econtech, and rnn.econall. However, as our

results will demonstrate, the effects of fine-tuning differ markedly across architectures: the RNN's greater capacity allows it to learn more nuanced adjustments to the economic loss surface, while the FFNet's simpler structure limits the degree to which fine-tuning can reshape predictions.

5.3 Trading Experiment Implementation

To evaluate economic value, we embed out-of-sample volume forecasts for 2022 in a stylized trading experiment that closely mirrors the design in Goyenko et al. (2025). The experiment contrasts a "True World," which determines realized trading costs, against an "Agent World," which governs the agent's execution decisions.

In the True World, we observe realized daily returns and realized volatility from the data. Price impact is computed using the volatility-augmented impact coefficient $\tilde{\lambda}(V, s)$ defined in equation (7), with $\beta = 1$ and the volatility surprise s computed from realized volatility. This provides a ground-truth mapping from trading rates to trading costs that incorporates both volume and volatility conditions.

In the Agent World, a trader faces a stream of exogenous signals about future returns and uses volume forecasts—and, in the volatility-aware variants, volatility forecasts—to choose trading rates. Specifically, with probability $\pi = 0.05$, a stock-day receives a "perfect" signal about the sign of its next five-day return. Signals are independent across stocks and days. Conditional on receiving a signal, the target portfolio weight $w_i^\star$ is long (short) for positive (negative) predicted returns; otherwise the position remains unchanged. Target weights form an equal-weighted long-short portfolio with 50% of assets under management in each leg.

Given current weight $w_{i,t-1}$ and target $w_{i,t}^\star$, the agent executes:

$$w_{i,t} = w_{i,t-1} + \hat{z}_{i,t}(w_{i,t}^\star - w_{i,t-1}), \tag{20}$$

where $\hat{z}_{i,t}$ is the trading rate implied by the agent's forecasts. This design isolates the effect of execution quality: the return signal is deliberately unrealistically strong (perfect sign foresight) so that differences in performance are driven almost entirely by how efficiently the agent executes trades rather than by return prediction skill.

5.4 Evaluation Metrics

We report three categories of performance metrics. Statistical metrics include mean squared error and R^2 of volume forecasts relative to the MA5 baseline, measuring pure predictive accuracy. Economic metrics include Mean Economic Loss (MEL), computed as the average of equation (4) across stock-days, and MEL reduction relative to MA5, normalized by the oracle (perfect foresight of realized volume). Portfolio metrics include the annualized Sharpe ratio of the long-short strategy after trading costs, computed for each model across a grid of AUM levels ($100M, $1B, $10B, $100B).

We additionally conduct regime-conditioned analysis by partitioning the sample into volatility tertiles (Low, Medium, High) based on the volatility surprise $s_{i,t}$. This decomposition isolates the behavior of volatility-aware execution during stress episodes, where the circuit-breaker mechanism is most relevant.

6 Statistical Forecasting Results

We begin by examining pure forecasting performance for log volume residuals on the 2022 out-of-sample period. Table 2 reports MSE and R^2 (measured as percentage reduction in MSE relative to MA5) across all three universes and all model architectures.

Table 2: Statistical Forecast Performance (MSE and R^2 vs MA5) across All Universes

	Largest Universe		**Smallest Universe**		**Random Universe**	
Method	**MSE**	R^2	**MSE**	R^2	**MSE**	R^2
MA5	0.121	0.0%	0.796	0.0%	0.246	0.0%
Linear Models						
olstech	0.096	20.9%	**0.677**	**14.9%**	0.211	14.4%
olsall	**0.093**	**23.8%**	0.678	14.9%	**0.207**	**16.0%**
Feedforward Neural Networks						
nntech	0.103	14.9%	0.690	13.3%	0.220	10.6%
nnall	0.100	17.4%	0.693	12.9%	0.215	12.6%
nn.econtech	0.101	16.5%	0.688	13.6%	0.218	11.4%
nn.econall	0.099	18.2%	0.710	10.8%	0.214	13.0%
Recurrent Neural Networks						
rnntech	0.105	13.8%	0.695	12.6%	0.222	9.9%
rnnall	0.102	16.2%	0.722	9.2%	0.218	11.5%
rnn.econtech	0.100	17.7%	0.686	13.7%	0.217	11.8%
rnn.econall	0.098	19.1%	0.776	2.5%	0.213	13.5%
oracle	0.000	100.0%	0.000	100.0%	0.000	100.0%

6.1 Large-Cap Results Align with TVA

Results for the Largest universe broadly replicate the findings of Goyenko et al. (2025). Volume exhibits substantial predictability: even simple linear models using only technical features (olstech) explain over 20% of residual variance, while the full-feature Ridge specification (olsall) achieves 23.8% R^2. Cross-sectional characteristics contribute meaningfully beyond own-stock history, as evidenced by the 2.9 percentage point improvement from olstech to olsall.

The neural network architectures occupy intermediate positions in the performance hierarchy. Feedforward networks achieve R^2 values between 14.9% (nntech) and 18.2% (nn.econall), outperforming RNNs on pure statistical metrics but falling short of linear models. This ordering—linear > FFNet > RNN for statistical fit—is consistent with the bias-variance tradeoff: the higher-capacity architectures have greater potential for overfitting, which manifests as weaker out-of-sample R^2 despite potentially better in-sample fit. Ridge regression's implicit ℓ_2 regularization proves highly effective at controlling variance in this setting.

A notable finding is that economic fine-tuning improves statistical fit across both neural architectures. For RNNs, the R^2 of rnn.econtech (17.7%) exceeds that of rnntech (13.8%), and rnn.econall (19.1%) outperforms rnnall (16.2%). The pattern holds for FFNets, though less dramatically: nn.econtech improves from 14.9% to 16.5%, and nn.econall from 17.4% to 18.2%. This counter-intuitive result—that optimizing an economic objective improves a statistical metric—reflects the implicit regularization provided by the economic loss. By concentrating model capacity on the in-

termediate volume regime where the trading-rate sigmoid exhibits maximum curvature, fine-tuning effectively downweights noise in the distributional tails and prevents overfitting.

6.2 Small-Cap Results Reveal Fragility

The Smallest universe presents a starkly different landscape. Linear models dominate throughout: olstech achieves the lowest MSE and highest R^2 (14.9%), and the addition of cross-sectional features (olsall) provides negligible incremental gain. Both neural architectures consistently underperform their linear counterparts, with rnnall exhibiting particularly weak R^2 (9.2%).

The FFNet occupies a middle ground, achieving modestly better statistical performance than RNNs (nntech at 13.3% vs. rnntech at 12.6%), consistent with its lower parameter count providing some protection against overfitting in this noisy environment. However, the gap to linear models remains substantial, suggesting that neither non-linear architecture nor temporal modeling adds value in micro-cap volume forecasting.

Most strikingly, economic fine-tuning proves catastrophic for the full-feature RNN. The R^2 of rnn.econall collapses from 9.2% to 2.5%—a deterioration of 6.7 percentage points that renders the model nearly useless for forecasting. The FFNet exhibits a similar but less severe pattern: nn.econall declines from 12.9% to 10.8%, a 2.1 percentage point drop. In contrast, the technical-only variants (rnn.econtech and nn.econtech) show modest improvements, suggesting that the combination of high model capacity and noisy cross-sectional features creates a particularly toxic interaction with economic fine-tuning.

This failure reflects the confluence of three factors. First, the economic loss function assumes that price impact follows $\lambda = \lambda_0/V$, a specification that is increasingly misaligned with reality in illiquid markets where spreads, depth, and order flow imbalance constitute first-order frictions beyond volume. Second, micro-cap volume is driven by idiosyncratic, event-driven flows—earnings surprises, analyst coverage changes, retail attention—that generate high variance and low signal-to-noise ratios. Third, the asymmetric penalty structure of the economic loss causes the model to become excessively conservative, biasing predictions toward the unconditional mean and destroying predictive content.

6.3 Mixed Universe Results

The Random universe, comprising 200 stocks drawn from across the capitalization spectrum, occupies an intermediate position. Linear models again lead in R^2 (olsall achieves 16.0%), but the neural architectures perform more competitively than in the Smallest universe.

Economic fine-tuning produces consistent improvements for both neural architectures. For RNNs, rnn.econtech improves from 9.9% to 11.8%, and rnn.econall from 11.5% to 13.5%. For FFNets, nn.econtech improves from 10.6% to 11.4%, and nn.econall from 12.6% to 13.0%. The heterogeneity of the universe appears to provide sufficient diversity for neural networks to learn meaningful patterns while the economic loss provides beneficial regularization rather than destabilization.

A critical observation is that technical features alone prove insufficient for RNNs in this setting: rnntech achieves only 9.9% R^2, the weakest among all model variants. The FFNet shows a similar pattern, with nntech (10.6%) substantially underperforming nnall (12.6%). Cross-sectional characteristics are essential for neural network performance in heterogeneous universes, consistent with the intuition that firm-level attributes (size, liquidity, volatility) contain critical information about volume dynamics that purely time-series features miss.

Table 3: Architecture Performance Patterns

Pattern	**Largest**	**Smallest**	**Random**
Best statistical model	OLS	OLS	OLS
FFNet vs. RNN	FFNet > RNN	FFNet > RNN	FFNet ≈ RNN
Fine-tuning helps?	Yes (+2–4pp)	Mixed/Harmful	Yes (+1–2pp)
Cross-sectional value?	High (+3pp)	Low (<1pp)	Moderate (+2pp)

6.4 Architecture Comparison Summary

Table 3 summarizes the key patterns across architectures and universes.

Three insights emerge. First, linear models achieve the highest statistical R^2 in all three universes, benefiting from implicit regularization that neural networks lack. Second, within neural architectures, the simpler FFNet consistently matches or exceeds the RNN on pure statistical metrics, suggesting that the additional complexity of temporal modeling does not improve volume forecasting accuracy. Third, economic fine-tuning improves statistical fit in liquid, homogeneous universes but can be harmful in illiquid, heterogeneous settings—particularly when combined with high-capacity models and noisy features.

These statistical results foreshadow the economic findings: the model that forecasts volume most accurately is not necessarily the model that executes trades most efficiently.

7 Economic Value and Trading Performance

We now turn to the central question: do improvements in statistical forecasting translate into reductions in implementation costs? Table 4 reports Mean Economic Loss and MEL reduction (normalized such that the oracle achieves 100%) across AUM levels for the Largest and Smallest universes.

Table 4: Mean Economic Loss (MEL) and Reduction in Largest and Smallest Universes

	Largest Universe				**Smallest Universe**			
Method	**$100M**	**$1B**	**$10B**	**$100B**	**$100M**	**$1B**	**$10B**	**$100B**
Panel A: MEL ($\times 10^{-8}$)								
MA5	0.00001	0.000962	0.031379	0.052757	0.00001	0.001	0.099686	7.930924
Linear Models								
olstech	0.00001	0.000961	0.031243	0.052750	0.00001	0.001	0.099276	7.812727
olsall	0.00001	0.000961	0.031223	0.052748	0.00001	0.001	0.099273	7.842342
Feedforward Neural Networks								
nntech	0.00001	0.000959	0.030688	0.052714	0.00001	0.001	0.098	7.511
nnall	0.00001	0.000959	0.030690	0.052715	0.00001	0.001	0.098	7.475
nn.econtech	0.00001	0.000959	0.030508	0.052346	0.00001	0.001	0.098	7.511
nn.econall	0.00001	0.000959	0.030518	0.052346	0.00001	0.001	0.098	7.475
Recurrent Neural Networks								
rnntech	0.00001	0.000960	0.030882	0.052728	0.00001	0.001	0.099417	7.866053
rnnall	0.00001	0.000960	0.030879	0.052727	0.00001	0.001	0.099405	7.859664
rnn.econtech	0.00001	0.000959	**0.030565**	**0.052351**	0.00001	0.001	0.099243	7.803615
rnn.econall	0.00001	0.000959	**0.030543**	**0.052349**	0.00001	0.001	0.207850	16.35338
oracle	0.00001	0.000958	0.030655	0.052712	0.00001	0.001	0.088669	6.963473
Panel B: MEL Reduction (%, oracle = 100%)								
MA5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Linear Models								
olstech	0.0	22.1	9.0	1.3	0.6	3.2	3.7	12.2
olsall	0.0	32.3	10.3	1.5	0.6	3.2	3.7	9.2
Feedforward Neural Networks								
nntech	0.0	45.8	45.7	7.4	0.9	3.0	15.3	43.4
nnall	0.0	45.5	45.5	7.3	0.9	3.2	15.4	47.1
nn.econtech	0.0	**50.1**	**57.6**	**70.2**	0.9	3.0	15.3	43.4
nn.econall	0.0	49.8	57.0	70.1	0.9	3.2	15.4	47.1
Recurrent Neural Networks								
rnntech	0.0	12.3	32.9	5.0	0.9	2.1	2.4	6.7
rnnall	0.0	22.8	33.1	5.1	0.9	2.2	2.5	7.4
rnn.econtech	0.0	37.1	53.9	69.3	0.7	3.4	4.0	13.2
rnn.econall	0.0	31.7	55.4	69.5	−24.3	−1004	−981	−870

7.1 Economic Fine-Tuning Delivers Value in Large Caps

In the Largest universe, economically fine-tuned neural networks deliver substantial MEL reductions, with FFNets achieving the strongest performance. The fine-tuned FFNet nn.econtech achieves 57.6% of the oracle's gain at $10B AUM and 70.2% at $100B, outperforming the fine-tuned RNN

rnn.econtech (53.9% at $10B, 69.3% at $100B). Even without fine-tuning, FFNets outperform RNNs on MEL: nntech achieves 45.7% reduction at $10B versus 32.9% for rnntech. This confirms a key finding: simpler architectures with lower capacity can better exploit economic structure when that structure is well-specified.

The benefits of forecasting scale dramatically with AUM. At $100M, price impact is negligible regardless of execution quality, and all models perform identically. At $1B and $10B, impact becomes material and the choice of forecast model matters substantially. At $100B, the investor is so large that optimal trading rates are uniformly low (to avoid moving markets), yet the fine-tuned neural networks—both FFNet and RNN—achieve roughly 70% of the oracle's performance, demonstrating that economic fine-tuning remains valuable even when absolute trading rates are constrained.

A subtle but important finding is that the model achieving highest R^2 (olsall at 23.8%) does not achieve the highest MEL reduction (nn.econtech at 57.6% for $10B). This divergence reflects the nonlinear relationship between statistical accuracy and economic value. The economic loss function weights errors according to their position on the trading-rate sigmoid: errors in the intermediate volume regime, where the sigmoid is steep, are penalized heavily, while errors at the extremes matter less. Fine-tuned neural networks learn to be accurate precisely where accuracy matters most for trading decisions.

7.2 Architecture Comparison for MEL

The relative performance of FFNet versus RNN on MEL metrics inverts their ranking on statistical metrics. While RNNs achieve slightly higher R^2 than FFNets in some specifications, FFNets consistently achieve lower MEL. At $10B in the Largest universe, nntech achieves 45.7% MEL reduction compared to 32.9% for rnntech—a 12.8 percentage point advantage. With fine-tuning, the gap narrows but FFNet retains a modest edge: nn.econtech at 57.6% versus rnn.econtech at 53.9%.

This pattern reflects the different objectives embedded in each metric. R^2 rewards uniform accuracy across the entire volume distribution, while MEL rewards accuracy specifically in the trading-relevant regime. The FFNet's simpler architecture appears to align more naturally with this concentrated objective, perhaps because its limited capacity prevents it from wasting representational power on economically irrelevant regions of the volume distribution.

7.3 Fine-Tuning Destroys Value in Small Caps

The Smallest universe tells a cautionary tale. While nn.econtech and rnn.econtech achieve modest MEL improvements (13–43% at $100B depending on architecture), the full-feature RNN variant rnn.econall produces catastrophic results: MEL reductions are not merely zero but severely negative, ranging from -24.3% at $100M to -1004% at $1B. The model performs worse than the naive MA5 baseline by an order of magnitude.

Notably, the FFNet avoids this catastrophic failure even with full features. While nn.econall does not improve upon nnall, it also does not collapse—both achieve roughly 47% MEL reduction at $100B. This resilience reflects the FFNet's lower capacity: with only 3,500 parameters versus 25,000 for the RNN, the FFNet simply cannot memorize the noisy cross-sectional patterns that cause the RNN to overfit under economic fine-tuning.

The contrast between rnn.econtech and rnn.econall is equally instructive. The technical-only variant (rnn.econtech) has fewer parameters and less capacity to overfit, allowing it to extract modest bene-

Table 5: Annualized Sharpe Ratios across Universes and AUM Levels

	Largest Universe				**Smallest Universe**				**Random Universe**			
Method	**$10B**	**$1B**	**$100M**	**$10M**	**$10B**	**$1B**	**$100M**	**$10M**	**$10B**	**$1B**	**$100M**	**$10M**
MA5	0.56	6.20	6.61	6.65	0.84	0.84	1.14	2.37	2.01	3.57	4.03	9.03
Linear Models												
olstech	0.61	6.21	6.62	6.67	0.47	1.01	1.09	2.33	1.98	3.59	4.04	8.97
olsall	0.65	6.20	6.62	6.53	0.34	1.02	1.12	2.39	2.02	3.60	4.06	9.06
Feedforward Neural Networks												
nntech	**0.87**	6.21	6.60	6.53	−0.25	−0.25	0.51	3.20	2.26	3.90	4.35	9.25
nnall	0.91	6.20	6.59	6.53	−0.32	−0.32	0.07	3.26	2.24	3.88	4.33	9.25
nn.econtech	0.21	6.04	6.44	6.53	−0.25	−0.25	0.51	3.20	0.28	0.36	0.38	5.08
nn.econall	0.21	6.04	6.44	6.53	−0.32	−0.32	0.07	3.26	0.31	0.40	0.41	5.10
Recurrent Neural Networks												
rnntech	0.67	6.21	6.62	6.53	0.12	0.89	0.97	2.98	2.13	3.76	4.18	9.16
rnnall	0.65	6.21	6.61	6.53	−0.02	0.79	0.87	2.97	2.11	3.85	4.31	**9.23**
rnn.econtech	0.29	6.10	6.44	6.53	−1.13	−1.10	−0.74	−1.06	0.31	0.39	0.40	5.11
rnn.econall	0.31	6.09	6.44	6.53	−1.15	−0.89	−1.15	−0.87	0.35	0.43	0.44	5.09
oracle	0.90	6.20	6.60	6.53	−0.61	0.04	0.10	3.20	2.23	3.83	4.27	9.27

fits from economic fine-tuning (13.2% at $100B). The full-feature variant (rnn.econall) has sufficient capacity to memorize noise in the cross-sectional characteristics, and the economic gradients amplify rather than correct this overfitting. The lesson is clear: economic fine-tuning is not a panacea and can actively harm performance when applied to high-capacity models in low signal-to-noise environments.

7.4 Sharpe Ratio Analysis

Table 5 reports annualized Sharpe ratios from the trading experiment across all three universes. These results reveal a critical disconnect between MEL minimization and portfolio performance.

7.4.1 The MEL-Sharpe Disconnect

In the Largest universe, the models that achieve highest MEL reduction substantially underperform on Sharpe ratio. At $10B, the unfine-tuned FFNet nnall achieves 0.91 Sharpe—the best among all models and approaching the oracle's 0.90. Yet the fine-tuned nn.econall, which achieves 57% MEL reduction compared to nnall's 45.5%, delivers only 0.21 Sharpe—less than one-quarter the performance. The pattern holds for RNNs: rnn.econall achieves better MEL than rnnall but worse Sharpe (0.31 vs. 0.65).

This disconnect arises because fine-tuning teaches models to be conservative: the asymmetric penalty structure of the economic loss causes systematic underestimation of volume to avoid worst-case price impact. But excessive conservatism means lower trading rates, less exposure to the (profitable) return signal, and ultimately lower gross returns that are not fully offset by the reduction in trading costs. MEL measures implementation efficiency conditional on a given set of trades; Sharpe measures total strategy performance including the decision of how much to trade.

7.4.2 FFNet Achieves Best Sharpe in Large Caps

Among all models in the Largest universe, the unfine-tuned FFNets achieve the highest Sharpe ratios at scale. At \$10B, nnall reaches 0.91 Sharpe compared to 0.65 for rnnall and 0.65 for olsall. This represents a 40% improvement over the RNN and a near-match with the oracle (0.90). The FFNet's advantage persists at lower AUM levels where it matches or exceeds all alternatives.

The FFNet's success on Sharpe despite middling statistical R^2 illustrates a key insight: for trading applications, what matters is not overall forecast accuracy but accuracy in the regions that drive trading decisions. The FFNet appears to achieve this without the overcorrection that fine-tuning induces, suggesting that its simpler architecture naturally concentrates capacity on economically relevant patterns.

7.4.3 Small-Cap and Random Universe Results

The Smallest universe amplifies the MEL-Sharpe disconnect to the point of strategy failure. Fine-tuned models—both FFNet and RNN—produce negative Sharpe ratios across most AUM levels, with rnn.econall achieving -0.87 at \$10M versus $+2.97$ for rnnall without fine-tuning. Even unfine-tuned FFNets struggle, with nnall achieving -0.32 at \$10B. The oracle itself achieves negative Sharpe at \$10B (-0.61), confirming that this universe is fundamentally difficult to trade profitably at scale.

The Random universe exhibits the same qualitative pattern: unfine-tuned neural networks achieve the best Sharpe (nntech at 9.25, rnnall at 9.23 at \$10M), while fine-tuned variants collapse to roughly 5.1 Sharpe—a penalty of four Sharpe points from fine-tuning. The robust finding across all three universes is that MEL minimization and Sharpe maximization are distinct objectives, and optimizing the former can substantially harm the latter.

7.5 Summary: When Does Each Objective Matter?

The divergence between MEL and Sharpe performance suggests different models are appropriate for different investment contexts:

1. **Execution desks** seeking to minimize implementation shortfall for externally determined trades should prefer economically fine-tuned models. The fine-tuned FFNet (nn.econtech) achieves the highest MEL reduction while avoiding the catastrophic failures of high-capacity RNNs.
2. **Portfolio managers** seeking to maximize risk-adjusted returns from a trading strategy should prefer unfine-tuned neural networks. The FFNet without fine-tuning (nnall) achieves the best Sharpe ratio in large-cap universes while maintaining competitive MEL performance.
3. **Investors in illiquid markets** should avoid neural networks entirely and rely on simple linear models, which provide modest but stable improvements without the tail risks of more complex architectures.

The optimal model depends critically on the investor's objective function and the characteristics of the trading universe.

Table 6: MEL Reduction: Volume-Only vs. Volatility-Adjusted Agents

Method	$100M	$1B	$10B	$100B
Largest Universe				
Linear Models				
olstech	0.00	22.09	9.02	1.25
ols.vol_adj tech	0.00	21.94	6.76	1.12
olsall	0.00	32.30	10.30	1.50
ols.vol_adj all	+1.81	+34.40	+10.16	+1.35
Feedforward Neural Networks				
nntech	0.00	45.80	45.70	7.40
nn.vol_adj tech	+0.18	**+51.17**	+46.79	+7.43
nnall	0.00	45.50	45.50	7.30
nn.vol_adj all	+0.01	+45.73	+45.26	+7.23
Recurrent Neural Networks				
rnntech	0.00	12.33	6.04	0.81
rnn.vol_adj tech	+0.35	+21.70	+6.81	+0.72
rnnall	0.00	22.80	7.00	1.00
rnn.vol_adj all	+0.32	**+31.15**	+7.03	+0.82
Smallest Universe				
Linear Models				
olstech	0.59	3.23	3.73	12.22
ols.vol_adj tech	0.59	3.23	3.73	12.22
Feedforward Neural Networks				
nntech	0.90	3.00	15.30	43.40
nn.vol_adj tech	+0.90	+3.02	+15.30	+43.41
nnall	0.90	3.20	15.40	47.10
nn.vol_adj all	+0.90	+3.21	+15.40	+47.10
Recurrent Neural Networks				
rnntech	0.88	2.14	2.44	6.70
rnn.vol_adj tech	+0.82	+2.11	+2.43	+6.70
rnnall	0.90	2.20	2.50	7.40
rnn.vol_adj all	+0.88	+2.17	+2.49	+7.40

8 Volatility-Aware Execution

We now evaluate our proposed extension: incorporating volatility surprises into the execution decision via the augmented impact coefficient $\tilde{\lambda}_{i,t} = \lambda_0 e^{-v_{i,t}} \cdot \max\{0.1, 1 + \beta s_{i,t}\}$.

8.1 The Circuit Breaker Effect

Table 6 compares MEL reduction for volume-only agents versus volatility-aware agents across the Largest and Smallest universes, spanning all three model architectures.

8.1.1 Largest Universe: Selective Benefits

In the Largest universe, the value of volatility adjustment varies substantially by architecture. For linear models, the adjustment provides negligible or slightly negative value—ols.vol_adj tech

underperforms olstech at $10B (6.76% vs. 9.02%). Large-cap volatility rarely exceeds the threshold necessary to trigger meaningful adjustment, and when the circuit breaker does activate, it may induce excessive conservatism that costs more than it saves.

For neural networks, however, volatility adjustment delivers material benefits at intermediate AUM levels. The FFNet nn.vol_adj tech improves from 45.80% to 51.17% MEL reduction at $1B—a gain of over 5 percentage points. The RNN rnn.vol_adj all improves from 22.80% to 31.15% at $1B, representing an 8.35 percentage point improvement. These gains are concentrated at the $1B level where price impact is material but not yet so large that all models converge to uniformly slow trading.

The pattern suggests that volatility adjustment complements neural network forecasts specifically. Neural networks may occasionally produce volume forecasts that are too aggressive during stress episodes; the volatility overlay provides a second line of defense that catches these errors. Linear models, with their inherent conservatism, derive less benefit from an additional conservative adjustment.

8.1.2 Smallest Universe: Diminishing Returns

In the Smallest universe, the volatility adjustment provides negligible incremental value across all architectures. MEL reductions with and without volatility adjustment are nearly identical: rnnall achieves 2.20% at $1B while rnn.vol_adj all achieves 2.17%. The FFNet shows the same pattern: nnall at 3.20% versus nn.vol_adj all at 3.21%.

This null result is informative. In micro-cap markets, the fundamental challenge is not that models trade too aggressively during volatility spikes—it is that volume forecasting itself is unreliable. The volatility adjustment can only help if the underlying volume forecast provides a reasonable starting point; when the base signal is noisy (MSE > 6 for RNNs in the Smallest universe), adding a volatility overlay cannot rescue performance. The circuit breaker addresses model overconfidence, not model incompetence.

8.2 Regime-Conditioned Analysis

To understand where volatility adjustment creates value, we condition on realized volatility tertiles. Table 7 reports MEL reduction (oracle-normalized) for the Smallest universe in the High Volatility tertile, where the circuit breaker mechanism is most relevant.

The results are striking. Without volatility adjustment, all well-behaved models achieve modest MEL reductions of 5–12% in the high-volatility regime. With volatility adjustment, these same models achieve 27–39%—an improvement of approximately 20–25 percentage points. The circuit breaker is doing exactly what it was designed to do: detecting elevated volatility (even when volume appears high), increasing the effective impact coefficient, and inducing more conservative execution that preserves capital during turbulent periods.

The FFNet exhibits the same pattern as other architectures: nn.vol_adj tech achieves 36% MEL reduction at $10B compared to 10% for nntech without adjustment. This 26 percentage point improvement confirms that volatility awareness benefits all model types in high-stress regimes, not just RNNs.

Table 7: Smallest Universe — MEL Reduction in High Volatility Regime

Method	**$100M**	**$1B**	**$10B**	**$100B**
MA5	0.0%	0.0%	0.0%	0.0%
Linear Models				
olstech	+5%	+6%	+6%	+6%
ols.vol_adj tech	+27%	+29%	+32%	+24%
Feedforward Neural Networks				
nntech	+8%	+9%	+10%	+7%
nn.vol_adj tech	+30%	+32%	+36%	+21%
nnall	+7%	+8%	+9%	+6%
nn.vol_adj all	+28%	+30%	+34%	+19%
Recurrent Neural Networks				
rnntech	+10%	+11%	+12%	+8%
rnn.vol_adj tech	+32%	+34%	+39%	+22%
rnnall	−12.3%	−12.3%	−7.9%	−11.3%
rnn.vol_adj all	+29%	+30%	+32%	−13%

8.3 Rescuing Catastrophic Failures

The contrast with rnnall is particularly illuminating. Without volatility adjustment, rnnall produces catastrophically negative MEL reductions exceeding −10% in the high-volatility regime. With volatility adjustment (rnn.vol_adj all), MEL reduction becomes positive at 29–32% for AUM levels up to $10B. The volatility adjustment partially rescues a model that would otherwise be unusable.

Volume-only RNNs fail catastrophically in high-volatility micro-caps because they interpret elevated volume as a liquidity opportunity and trade aggressively precisely when spreads have widened and depth has evaporated. Volatility-adjusted models recognize the warning signal embedded in the volatility surprise and pull back.

Notably, the FFNet avoids this catastrophic failure mode even without volatility adjustment. While nnall achieves only modest positive MEL reduction (+6–9%) in high-volatility regimes, it never produces the extreme negative values seen with rnnall. This resilience again reflects the FFNet's lower capacity: it cannot learn the complex (and ultimately spurious) patterns that cause the RNN to trade aggressively at exactly the wrong times. For risk-conscious investors, this robustness may be as valuable as the FFNet's absolute performance.

8.4 Mechanism and Limitations

The circuit breaker operates through a simple but effective mechanism. When the volatility surprise $s_{i,t}$ exceeds zero (indicating realized volatility above its 60-day average), the adjustment factor $(1 + \beta s_{i,t})$ increases the effective impact coefficient $\tilde{\lambda}$. This shifts the optimal trading-rate sigmoid to the right, requiring higher volume to justify the same level of trading intensity. The agent automatically becomes more cautious in turbulent conditions, even if volume appears elevated.

The relative value of volatility adjustment across architectures can be summarized as follows:

Several caveats apply to the regime analysis. Oracle-normalized MEL reductions are sensitive

Table 8: Summary: When Does Volatility Adjustment Help?

Condition	**OLS**	**FFNet**	**RNN**
Large-cap, normal volatility	No	Modest	Modest
Large-cap, high volatility	Yes	Yes	Yes
Small-cap, normal volatility	No	No	No
Small-cap, high volatility	Yes	Yes	Critical

to small denominators and hence to noise in subsamples. Volatility tertiles are a coarse proxy for market stress; crisis episodes may exhibit structure not fully captured by 60-day volatility deviations. Sample sizes per regime are smaller, reducing statistical precision. Nevertheless, the consistent sign flip between volume-only and volatility-aware models in the high-volatility regime represents a robust finding that supports our central hypothesis: volatility is indeed a cost driver beyond volume, and incorporating volatility surprises into execution logic delivers material benefits in precisely the conditions where naive volume-based trading is most dangerous.

8.5 Practical Implications

The volatility extension yields three actionable insights for practitioners:

First, volatility adjustment is most valuable as a *defensive* mechanism rather than a source of alpha. In normal market conditions, it provides minimal benefit and may even cost slightly in large-cap universes. Its value emerges during stress episodes, where it prevents the catastrophic losses that can result from trading aggressively into illiquid conditions.

Second, the benefits of volatility adjustment scale with model complexity. Simple linear models, which are inherently conservative, gain less from an additional conservative overlay. Neural networks, which have capacity to learn aggressive trading patterns, benefit more from a volatility-based circuit breaker that tempers their enthusiasm during turbulent periods.

Third, volatility adjustment cannot substitute for good volume forecasting. In the Smallest universe, where base volume models achieve MSE exceeding 6, the volatility overlay provides no benefit because the underlying signal is too noisy. Investors should first ensure their volume forecasts are reliable before adding volatility overlays; the adjustment complements good forecasts but cannot rescue bad ones.

9 Synthesis and Discussion

Our results yield several insights for the design and deployment of volume-forecasting systems for optimal execution.

9.1 The MEL-Sharpe Disconnect

A central finding is that minimizing Mean Economic Loss does not maximize portfolio Sharpe ratio. Fine-tuned RNNs achieve the highest MEL reductions in the Largest universe (37.1% vs. 22.1% for olstech at \$1B) but the lowest Sharpe ratios (0.31 vs. 0.65 for olsall at \$10B). The mechanism is the conservative bias induced by the asymmetric penalty structure of the economic loss function. By penalizing overestimation of volume more heavily than underestimation, fine-tuning teaches models to systematically predict lower volumes, leading to lower trading rates and less exposure to profitable return signals.

This finding has practical implications. If the objective is to minimize implementation shortfall for a given set of trades, economic fine-tuning is beneficial. If the objective is to maximize risk-adjusted returns from a trading strategy, economic fine-tuning may be counterproductive. The optimal model depends on the investor's objective function, and blindly optimizing MEL can harm portfolio performance.

9.2 Context-Dependence of Fine-Tuning

Economic fine-tuning is not universally beneficial. In the Largest universe, fine-tuning improves both R^2 (by 2–4 percentage points) and MEL (by 10–25 percentage points at high AUM). In the Smallest universe, fine-tuning destroys R^2 (from 9.2% to 2.5% for rnnall) and produces catastrophically negative MEL and Sharpe ratios. The difference lies in the appropriateness of the cost model specification and the signal-to-noise ratio of the underlying data.

The TVA cost model assumes $\lambda = \lambda_0/V$, which is a reasonable approximation for large, liquid stocks where volume is the primary determinant of execution quality. In illiquid micro-caps, this assumption breaks down: spreads, depth, order flow imbalance, and other frictions constitute first-order costs that are not captured by volume alone. Fine-tuning with a misspecified cost model teaches models to optimize the wrong objective, with predictably poor results.

9.3 Model Complexity and Data Quality

Our results support the principle that model complexity should match data quality. Ridge regression achieves the highest R^2 in all three universes, benefiting from implicit regularization that prevents overfitting. RNNs have sufficient capacity to learn complex patterns but also to memorize noise; in low signal-to-noise environments like micro-caps, this flexibility is a liability.

For practitioners, this suggests a tiered approach: simple linear models for illiquid or heterogeneous universes where volume dynamics are dominated by idiosyncratic factors; RNNs (potentially with economic fine-tuning) for large, liquid stocks where systematic patterns are learnable and the cost model is well-specified; and volatility-aware overlays for all settings as a risk control mechanism.

9.4 The Value of Volatility Awareness

The volatility-augmented impact model delivers its greatest benefits precisely where they are most needed: in illiquid, high-volatility regimes where naive volume-based execution is most dangerous.

The 25–30 percentage point improvement in MEL reduction during high-volatility episodes represents substantial economic value and validates the extension of the TVA framework to incorporate volatility surprises.

The mechanism is intuitive: high volume does not always imply high liquidity. During panic selling, volume spikes but spreads widen and depth evaporates. A volume-only model interprets the spike as a trading opportunity; a volatility-aware model recognizes the warning signal and pulls back. This circuit-breaker effect preserves capital during stress and allows more aggressive trading during calm periods when liquidity is genuinely abundant.

10 Discussion and Implications

10.1 Alignment with TVA Results

Our replication corroborates the foundational claims of Goyenko et al. (2025), establishing the robustness of the TVA portfolio-theoretic framework across varying data samples and model specifications. The results confirm three critical dimensions of volume dynamics. First, we observe significant volume forecastability, particularly among large-capitalization stocks where coefficient of determination (R^2) values consistently exceed 20%. Second, decision-aware training mechanisms—specifically fine-tuning on the economic loss function ℓ_{econ}—consistently augment portfolio performance within liquid investment universes. Finally, the analysis highlights the generation of substantial implementation alpha; volume prediction yields superior risk-adjusted returns compared to a standard 5-day moving average (MA5) benchmark, with magnitudes comparable to pure return signal alpha at higher levels of assets under management (AUM).

10.2 Nuances and Extensions

Although our findings align directionally with TVA, the analysis uncovers distinct nuances regarding model complexity and liquidity constraints. Notably, Ridge regression models utilizing expanded feature sets demonstrate remarkable competitiveness, often matching or surpassing Recurrent Neural Networks (RNNs) in terms of Mean Squared Error (MSE) and Mean Economic Loss (MEL) outside the most liquid equities. This suggests that trading volume predictability is predominantly linear with respect to observable market variables, implying that rigorous feature engineering may offer higher marginal utility than increased architectural complexity.

Conversely, the efficacy of economic fine-tuning appears strictly contingent on liquidity. In micro-cap segments characterized by noisy volume data, ℓ_{econ} optimization tends to drive RNNs toward extreme, deleterious trading policies, indicating that decision-aware training necessitates a stable underlying cost model to function effectively. Furthermore, the incorporation of volatility surprises validates the utility of non-return variables in cost modeling. While this extension shows negligible impact in large-cap stocks, it functions as a critical structural break mechanism in micro-caps, transforming aggressive execution logic into robust, regime-aware heuristics that stabilize performance during periods of elevated volatility.

10.3 Strategic Implications

These empirical results suggest a heterogeneous implementation strategy for practitioners. For large, liquid equities, the deployment of TVA-style RNNs with economic fine-tuning remains optimal for maximizing implementation efficiency without compromising statistical goodness-of-fit. In contrast,

simpler linear models paired with conservative trading rules appear superior for illiquid market segments, as complex neural architectures are prone to overfitting cost-model misspecifications in these environments. Finally, volatility-aware execution protocols should be utilized not primarily as a source of alpha, but as a risk management overlay, insulating the portfolio against model degradation during high-volatility crash regimes.

11 Conclusion

We have implemented and extended the Trading Volume Alpha framework in a large cross-section of US equities, using both linear and neural models to forecast daily log volume and embedding these forecasts in a portfolio problem with price impact and tracking error. On the theoretical side, we derived the TVA economic loss from a participation-rate view of trading costs and showed how it yields a simple yet powerful sigmoid mapping from log volume to optimal trading rates. We then extended this framework to allow price impact to depend not only on volume but also on volatility surprises, yielding a volatility-aware trading rule. On the empirical side, we documented several robust patterns:

1. **Economic fine-tuning works for liquid stocks.** For large caps, training RNNs on the downstream portfolio loss yields models that trade smarter, reducing implementation shortfall by up to 37% of the oracle potential.
2. **Simplicity wins in illiquid stocks.** For small caps, simple Ridge regression models are robust and effective, whereas econ-tuned RNNs can become unstable and value-destroying.
3. **Linear models are a strong baseline.** Across all universes, linear models with good features capture the bulk of predictability, suggesting that the "alpha" in TVA comes as much from the economic framework as from deep learning.
4. **Volatility-aware execution is a risk management tool.** While volatility-aware TVA brings modest average gains in large caps, it is pivotal in high-volatility micro-caps, where it acts as a circuit breaker that protects against catastrophic overtrading.

More broadly, our results underscore the TVA message that predicting non-return variables and embedding these predictions in economic decision rules can be as valuable as forecasting returns themselves. Extending this approach to richer cost models, incorporating volatility, spreads, and cross-stock interactions, and to joint modeling of returns, variance and trading costs remains an exciting direction for future research.

References

Primary

Goyenko, Ruslan, Bryan T. Kelly, Tobias J. Moskowitz, Yinan Su, and Chao Zhang. 2025. *Trading Volume Alpha*. NBER Working Paper No. 33037. NBER link.

Trading Costs

Frazzini, Andrea, Ronen Israel, and Tobias J. Moskowitz. 2018. *Trading Costs*. Working paper. SSRN link (doi:10.2139/ssrn.3229719).

Gârleanu, Nicolae, and Lasse Heje Pedersen. 2013. Dynamic trading with predictable returns and transaction costs. *The Journal of Finance* 68(6): 2309–2340. doi:10.1111/jofi.12080.

Market Microstructure

Kyle, Albert S. 1985. Continuous auctions and insider trading. *Econometrica* 53(6): 1315–1335. doi:10.2307/1913210.

Glosten, Lawrence R., and Paul R. Milgrom. 1985. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of Financial Economics* 14(1): 71–100. doi:10.1016/0304-405X(85)90044-3.

Stoll, Hans R. 1978. The supply of dealer services in securities markets. *The Journal of Finance* 33(4): 1133–1151. doi:10.1111/j.1540-6261.1978.tb02053.x.

Ho, Thomas S. Y., and Hans R. Stoll. 1983. The dynamics of dealer markets under competition. *The Journal of Finance* 38(4): 1053–1074. doi:10.1111/j.1540-6261.1983.tb02282.x.

Grossman, Sanford J., and Merton H. Miller. 1988. Liquidity and market structure. *The Journal of Finance* 43(3): 617–633. doi:10.1111/j.1540-6261.1988.tb04594.x.

Statistical Learning

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer. Publisher link (doi:10.1007/978-0-387-84858-7).

White, Halbert. 1982. Maximum likelihood estimation of misspecified models. *Econometrica* 50(1): 1–25. doi:10.2307/1912526.

A Design Decisions

Hidden size. We evaluated hidden sizes $\{16, 32, 64, 128\}$. Hidden size 32 achieved the best trade-off between fit and overfitting: larger models showed signs of overfitting in validation MSE without improving MEL.

Residual learning. Training directly on $v_{i,t}$ yielded higher validation MSE and more unstable learning. Residualizing on MA5 improved MSE by roughly 15% and simplified optimization.

Loss normalization. Without normalizing ℓ_{econ}, its scale differed from ℓ_{MSE} by several orders of magnitude, causing either vanishing or exploding gradients during fine-tuning. Dividing by $(\lambda_0 + \mu)$ brought the two losses to comparable scales.

Economic weight α. We experimented with $\alpha \in \{0.5, 0.9, 0.99\}$. Lower values of α (more weight on economic loss) often degraded MSE and, in illiquid universes, MEL and Sharpe as well. The choice $\alpha = 0.99$ preserved the statistical structure learned in pre-training while still improving MEL in large caps.

Signal probability. The 5% signal probability for the trading experiment follows TVA's design. Lower probabilities increase sparsity and emphasize the importance of timing; higher probabilities make portfolio turnover and trading costs massive, overwhelming differences across models.