Learning to Profit: Prediction-based Deep Learning Strategies for Algorithmic Trading

***Abstract*—Prediction-based algorithmic trading decouples forecasting from decision-making, utilizing a modular design to mitigate the inherent noise and non-stationarity of stock markets. In this work, we explore two prediction-based trading approaches with distinct prediction targets and decision mechanisms using deep learning (DL) frameworks, proposing integrated pipelines: (1) return regression, where Large Language Model (LLM)-inspired architectures feed continuous forecasts into Deep Q-Network agents, and (2) directional classification, where a global tabular model learns binary probabilities across all stocks via embeddings coupled with dynamic per-stock threshold optimization. Under strict walk-forward validation on S&P 500 stocks, directional classification achieves 77.8% accuracy and 40.0% annual returns, substantially outperforming return regression (14.8%) and the S&P 500 benchmark (24.2%). Our results demonstrate the potential of prediction-based trading through modular design, where separating forecasting from profit-optimized decisions enables effective algorithmic trading.**

***Index Terms*—Automated Trading, Financial Forecasting, Large Language Models, Multimodal Learning, Reinforcement Learning, TabNet, Time Series**

I. Introduction

Proactive decision-making under uncertainty appears across diverse fields, where agents act today based on forecasts of future events, from business decisions [1], [2] to weather nowcasting for flash-flood response [3] and proactive network control preventing communication failures by reallocating resources pre-congestion due to future demands or failures [4]. Financial markets pose unique challenges for such desicion-makiing procudue where desicons are highliy concditened of future evenrts and their predictabiililty to forecast marker beghivor complex also for machine learning due to noisy, non-stationary price dynamics and regime-dependent behavior, requiring systems that account for forecast errors and shifts through modular pipelines separating prediction, decision optimization, and risk management rather than end-to-end policies [5]. Rather than only optimizing current trades, prediction-based decisions making explicitly tackles two coupled challenges: learning predictive models of future returns or directions and converting these noisy forecasts into robust trading and risk-management rules [1], [6], [7]. where decisions hinge on uncertain forecasts of future events.

This work explores two distinct prediction-based strategies, each representing a modular two-step approach: forecasting future market behavior, then optimizing trading decisions based on these predictions. The *Return Regression* framework (Stream A) adapts StockTime [8], combining LSTM encoders with frozen LLM embeddings and learnable ticker embeddings via multi-head attention fusion to predict continuous returns. These forecasts feed into a Deep Q-Network (DQN) agent trained via a parallel protocol that prevents data leakage. The *Trend Prediction* framework (Stream B) takes a fundamentally different approach: rather than predicting exact values, a Global TabNet outputs probabilistic confidence scores for binary UP/DOWN movement, capturing cross-asset dependencies via entity embeddings. This probabilistic formulation enables per-stock threshold optimization—trading only when confidence exceeds asset-specific thresholds calibrated on historical returns, effectively filtering low-confidence signals. To evaluate the modular prediction-to-decision strategies, we conduct experiments on both small (19 stocks) and large (99 stocks) subsets across diverse sectors using daily data under walk-forward validation. Our results show that Stream B significantly outperforms both the Stream A alternative and the S&P 500 benchmark. These gains remain consistent across scales, suggesting the approach generalizes well to broader market coverage. **Our main contributions are as follows**:

1) **Prediction-Based Trading Pipelines.** We propose two modular frameworks (Figure 1) that decouple forecasting from decision-making: (a) return regression with an LLM-inspired architecture and DQN agent, and (b) directional classification with Global TabNet and per-stock threshold optimization.
2) **Training and Optimization for Non-Stationary Data.** For Stream A: (i) a parallel training protocol ensuring the RL agent only observes out-of-sample forecasts, eliminating data leakage; and (ii) an accumulative learning strategy with selective weight transfer for regime adaptation. For Stream B: (iii) per-stock confidence thresholds optimized for historical returns within a global cross-stock TabNet, enabling the model to exploit each asset's unique predictability profile while leveraging shared market patterns.
3) **Empirical Findings.** Both forecasting approaches benefit from proposed training strategies: global training outperforms per-stock baselines, accumulative learning accelerates convergence by 10×, and directional accuracy peaks at 77.8% at the 21-day horizon. Our comparative analysis shows these improvements translate to 40.0% annual returns for threshold-based decisions versus 14.8% for RL-based policies, both outperforming the S&P 500 benchmark (24.2%).

The remainder of this paper is organized as follows: Section II reviews relevant literature, while Section IV describes the proposed modular architectures for Multi-Step Return

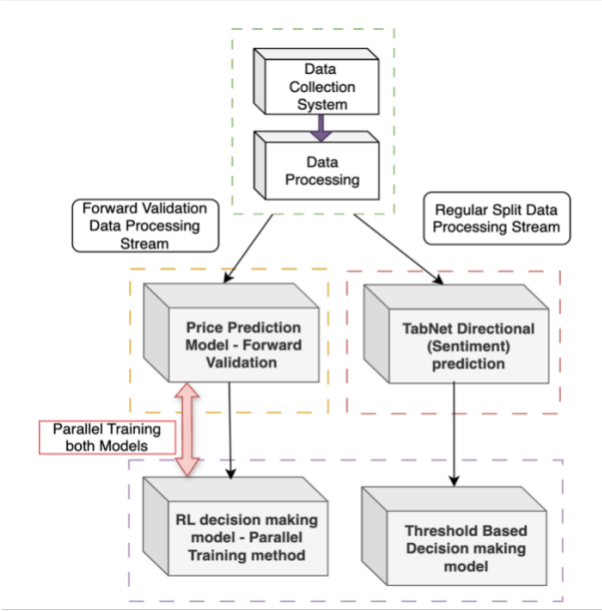


Fig. 1. Overview of the dual-stream framework. Stream A uses StockTime-based multimodal architecture for price prediction with a DQN agent. Stream B uses Global TabNet for directional prediction with threshold-based decisions.

Forecasting (Stream A) and Directional Classification (Stream B). Section V outlines data validation and training protocols. Section VI presents empirical results, architectural trade-offs, and market regime adaptation. Finally, Section VIII concludes and suggests future work.

II. Related Work

A. Deep Learning for Financial Forecasting

Traditional statistical models such as ARIMA struggle with non-linear market behavior [9]. Deep learning has since become dominant, with Recurrent Neural Networks (RNNs) [10], [11], hybrid statistical-neural approaches [12]–[14], and graph-based methods [6] showing strong performance on financial data [2], [15], [16].

Transformer architectures have advanced multi-horizon financial forecasting through efficient attention mechanisms. Informer [17] introduces ProbSparse self-attention to handle long sequences, PatchTST [18] segments time series into independently processed patches, and SAT [19] employs sparse attention to isolate distinct risk factors across forecasting horizons. Beyond temporal modeling, capturing inter-stock dependencies is critical for portfolio-level forecasting. Graph-based architectures address this by modeling relational structures among assets. MDGNN [20], CI-STHPAN [21], and recent spatio-temporal cross-stock models [22] leverage graph neural networks to capture sector-wide movements and cross-asset spillover effects. The emergence of LLMs has further expanded forecasting capabilities. StockTime [23] and Ploutos [24] enable multimodal fusion of numerical price data with textual signals such as news and earnings reports, while Chronos [25] demonstrates strong zero-shot generalization to unseen financial series. For structured tabular inputs, TabNet [26] employs sequential attention to perform interpretable feature selection, with demonstrated effectiveness in stock trend and volatility prediction [27].

In this work, we adopt StockTime for multi-step price prediction and TabNet for directional classification, representing two complementary forecasting paradigms to evaluation of trading frameworks.

B. Learning-Based Trading

While minimizing prediction error is standard in forecasting, lower regression error does not necessarily translate to higher financial returns [28], [29]. Trading strategy optimization has shifted from inefficient manual and grid/random searches to advanced automatic methods like Bayesian optimization, which efficiently navigates high-dimensional black-box objective functions by leveraging prior evaluations to select promising parameter sets [30], [31], and evolutionary computing such as genetic algorithms [32], which evolve near-optimal solutions for complex trading rules without requiring gradient information.

Reinforcement Learning for Trading. Reinforcement learning (RL) formulates trading as a sequential decision process, where agents learn policies to maximize cumulative returns through market interaction [33], [34]. Early direct reinforcement methods optimized risk-adjusted performance without relying on explicit forecasting models, while subsequent deep learning architectures enabled joint learning of financial signal representations and trading policies [35]. Recent advances apply DRL algorithms to portfolio management and position sizing [36], [37]. Value-based methods learn to estimate the expected cumulative reward of taking specific actions (such as buy, hold, or sell) in a given market state, while policy-based methods directly optimize the trading strategy itself. Recent works adapt DRL methods for portfolio management: DeepTrader embeds market conditions and asset correlations via hierarchical graph attention to balance risk-return [38], MetaTrader integrates diverse base policies through imitation and meta-policy selection for non-stationary adaptation [39], and a novel RMS-driven DRL approach optimizes trading portfolios by dynamically adjusting risk metrics [40]. We adopt a DQN agent whose state comprises predicted price trajectories, technical indicators, and current position, decoupling forecasting from policy optimization.

III. Preliminaries

A. Problem Setting

We model market data as a collection of M multivariate time series. For each stock $i \in \{1, \ldots, M\}$, let $\mathbf{X}_t^{(i)} \in \mathbb{R}^d$ denote the feature vector at time t.

a) Step 1: Multi-step forecasting.: We predict future targets (e.g., returns or directions) over the horizon:

$$\hat{\mathbf{Y}}_{t+1:t+h}^{(i)} = f_\theta\left(\mathbf{X}_{t-p+1:t}^{(i)}\right), \tag{1}$$

Given a lookback window of length p, we use the past sequence $\mathbf{X}_{t-p+1:t}^{(i)}$ to predict future outcomes over a horizon

of length h, where f_θ is a predictive model with parameters θ, and $\mathbf{Y}^{(i)}_{t+1:t+h}$ denotes the corresponding future targets, where $\mathbf{Y}^{(i)}_{t+1:t+h}$ may represent future returns, price changes, or directional movements.

b) Step 2: Leveraging predictions for trading.: Forecasts are mapped to trading actions via asset-specific policies:

$$a_t^{(i)} = \pi^{(i)}\left(\hat{\mathbf{Y}}^{(i)}_{t+1:t+h}\right), \qquad a_t^{(i)} \in \{\text{BUY}, \text{HOLD}, \text{SELL}\}. \tag{2}$$

Let $P_t^{(i)}$ denote the closing price of asset i and $\Delta P_t^{(i)} = P_{t+1}^{(i)} - P_t^{(i)}$. We maximize cumulative net return across assets:

$$\max_{\{\pi^{(i)}\}_{i=1}^{M}} \sum_{i=1}^{M} \sum_{t} r_t^{(i)}\left(a_t^{(i)}, \Delta P_t^{(i)}\right), \tag{3}$$

where $r_t^{(i)}(\cdot)$ denotes realized (net) return, including transaction costs. Policies share underlying model parameters while actions and rewards are computed per asset.

B. Technical Background

Walk-Forward Validation (WFV). Standard cross-validation with random splits is unsuitable for time-series as it causes *look-ahead bias*. WFV respects temporal order: each fold trains on $[t_0, t_k]$ and validates on $[t_k, t_{k+1}]$, then slides forward. We employ WFV in both approaches and extend it in Stream A to prevent data leakage between prediction and trading components.

Markov Decision Process (MDP). An MDP is defined by the tuple (S, A, R, P, γ): state space, action space, reward function, transition dynamics, and discount factor. An agent observes state $s_t \in S$, takes action $a_t \in A$, and receives reward r_t to maximize the expected discounted return.

Deep Q-Networks (DQN). DQN approximates the optimal action-value function $Q^*(s, a)$ via neural networks, stabilized by: (1) *experience replay*: storing transitions (s_t, a_t, r_t, s_{t+1}) to decorrelate training samples; and (2) *target networks*: using separate, slowly-updated weights θ^- to provide stable temporal-difference targets. Exploration is governed by an ϵ-greedy policy.

Global Multi-Stock Modeling. Instead of training independent models per asset, global modeling trains a single architecture f_θ across M stocks simultaneously. Stock-specific patterns are captured via entity identifiers $\mathbf{e}^{(i)}$ (e.g., learnable embeddings or one-hot vectors), while shared parameters θ learn universal market dynamics.

StockTime Architecture. A baseline architecture [8] is a multimodal framework that fuses numerical price sequences with textual market statistics via a dual-stream approach:(i) a **Price Stream** that segments raw input windows of length p into $N = \lfloor p/L \rfloor$ fixed-length patches of size L, processed through an autoregressive encoder to produce embeddings $\mathbf{p}_t$.(ii) a **Text Stream** that utilizes a frozen LLM to embed natural language descriptions of statistical properties (e.g., mean, variance, trends), resulting in context embeddings $\mathbf{c}_t$.

TabNet. TabNet [26] processes tabular inputs through sequential decision steps, each applying sparse attention masks $\mathbf{M}^{[n]}$ to select relevant features, enabling interpretable feature selection via $\mathbf{h}^{[n]} = \text{FeatureTransformer}(\mathbf{M}^{[n]} \odot \mathbf{x})$.

IV. Methods

We propose a dual-stream framework for automated trading unified by a global modeling strategy: (i) Stream A utilizes an adapted StockTime transformer for multi-step return forecasting and reinforcement learning, while (ii) Stream B employs TabNet for directional classification with optimized confidence thresholds.

A. Multi-Step Return Forecasting with RL-Based Trading

Stream A utilizes a modular pipeline that decouples multi-step return forecasting, performed by an LLM-augmented multimodal transformer, from policy optimization using a DQN trading agent.

1) Multimodal Forecasting Architecture (Self-StockTime): Self-StockTime extends StockTime (Fig. 2a) for multi-step forecasting. It introduces structural modifications to improve asset-specific representations and employs modular weight transfer for robust adaptation to changing market regimes.

- **Ticker Embeddings.** A learnable layer maps ticker symbols to dense vectors $\mathbf{z}^{(i)} \in \mathbb{R}^{d_e}$, capturing asset-specific dynamics (e.g., TSLA vs. KO).
- **LLM Context.** The LLM stream utilizes a frozen Llama 3.1 8B model to provide long-term temporal context, complementing the LSTM's short-term focus. It processes natural language descriptions of asset and sector-wide statistical properties, such as mean returns and volatility, to generate dense embeddings $\mathbf{c}_t$. We apply two primary modifications to this stream: (i) Self-Sourced Context: To ensure a self-contained system, we replace external news data with self-generated textual summaries, allowing the model to contextualize price patterns without third-party dependencies. (ii) Stream Simplification: We remove the original second LLM pass to avoid mode collapse and restore predictive diversity.
- **Multi-Head Attention Fusion.** Instead of additive fusion, we stack the three embedding streams and apply multi-head attention with $H = 4$ heads:

$$\mathbf{E} = [\mathbf{p}_t; \mathbf{c}_t; \mathbf{z}^{(i)}] \in \mathbb{R}^{3\times d} \tag{4}$$

$$\mathbf{F} = \text{MultiHead}(\mathbf{E}, \mathbf{E}, \mathbf{E}) \in \mathbb{R}^{3\times d} \tag{5}$$

  Following the self attention mechanism [**?**], we treat the stacked embedding matrix $\mathbf{E}$ as a sequence of three tokens, enabling the model to learn cross-modal dependencies and dynamically weight the contributions of price, context, and ticker streams based on current market conditions.
- **Projection Layers and Prediction Head.** To align the diverse input dimensions, we apply identical projection blocks (Linear $\rightarrow$ ReLU $\rightarrow$ Dropout $\rightarrow$ LayerNorm) mapping each modality to a shared dimension $d = 768$ prior to fusion. The resulting representation passes through a sequential MLP head ($768 \rightarrow 512 \rightarrow 256 \rightarrow L$) to output predicted returns. To prevent historical bias and ensure efficiency, we implement 8-bit quantization and reset the prediction head

weights at each fold of the accumulative learning process to facilitate rapid adaptation to new market regimes.

2) Stream A: Learning Strategies for Non-Stationary Markets: **Stationarity and Scale Normalization.** To address non-stationarity and scale variance, we predict percentage returns $r_t^{(i)} = (P_t^{(i)} - P_{t-1}^{(i)})/P_{t-1}^{(i)}$ rather than raw prices. Unlike standard normalization (e.g., Z-score), which suffers from distribution shift, returns are approximately stationary and scale-invariant across asset price levels.

Training and Validation Protocol. The input features consist of OHLCV price series (Open, High, Low, Close, and Volume). We implement an anchored WFV scheme, utilizing a 3-year training window followed by a 1-year validation period. This design offers two key benefits: (i) Regime Adaptation, as validation occurs on the most recent data relative to training to capture current market volatility, and (ii) Historical Retention, where the anchored start date ensures the model learns new regimes without discarding long-term historical patterns.

Accumulative Learning. Training under WFV presents a dilemma: retraining from scratch discards learned patterns, while full transfer risks overfitting to outdated regimes. We partition the model into *body* (LSTM, embeddings, attention) and *head* (projection layers). At each fold $k > 1$:

$$\theta_{\text{body}}^{(k)} \leftarrow \theta_{\text{body}}^{(k-1)}, \quad \theta_{\text{head}}^{(k)} \leftarrow \text{RandomInit}() \tag{6}$$

As illustrated in Fig. 2b, we adopt a sequential training strategy where the optimal checkpoint from fold k initializes fold $k+1$. Upon each transition, we re-initialize the prediction head to: (i) provide high initial gradients that force the model to escape outdated local minima, and (ii) re-map stable features from the frozen or pre-trained body to the current data distribution. This design maximizes the training data available to the downstream RL agent by generating out-of-sample inferences across all historical folds, effectively eliminating the need for a separate reserved inference window.

3) Predictive DQN Trading Agent: We train a DQN agent whose state incorporates multi-step forecasts $\hat{r}_{t+1:t+h}$ from Self-StockTime, enabling decisions based on anticipated trends rather than historical indicators alone.

MDP Formulation.

1) *State Space S.* The state $s_t \in$ comprises: normalized price $\tilde{P}_t$ (rolling z-score), h-step return forecasts $\hat{r}_{t+1:t+h}$ from Self-StockTime, RSI indicator $\widetilde{\text{RSI}}_t \in [-1, 1]$, current position $\text{pos}_t \in \{-1, 0, 1\}$, and stock identifier:

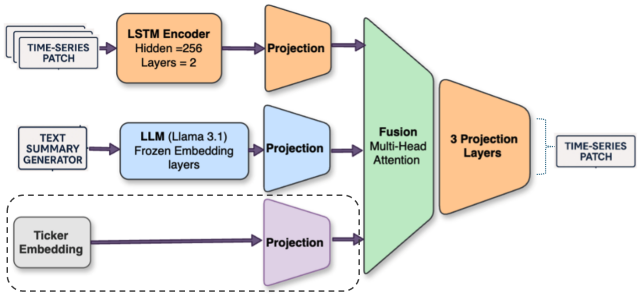
$$s_t = \left[\tilde{P}_t,\ \hat{r}_{t+1:t+h},\ \widetilde{\text{RSI}}_t,\ \text{pos}_t,\ \text{ticker}_t\right] \tag{7}$$

2) *Action Space A.* The agent selects from three discrete actions: $A = \{\text{BUY}, \text{HOLD}, \text{SELL}\}$

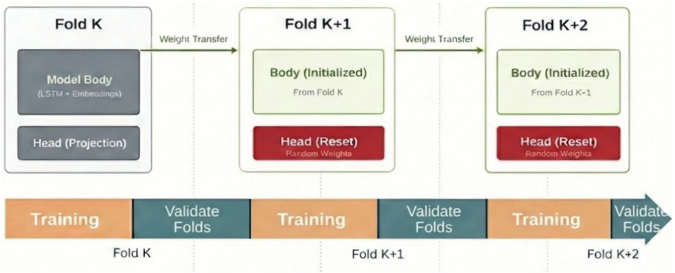
3) *Reward Function R.* The reward balances profit maximization with transaction costs:

$$R(s_t, a_t, s_{t+1}) = \underbrace{(P_{t+1} - P_t) \cdot \text{pos}_{t+1}}_{\text{Profit/Loss}} - \underbrace{\mathbb{1}[a_t \neq \text{HOLD}] \cdot c}_{\text{Transaction Cost}} + \underbrace{R_{\text{final}}}_{\text{Episode End}} \tag{8}$$

where c is the transaction fee, and R_{final} rewards/penalizes the agent based on its final position value at episode termination.



(a) Self-StockTime Architecture



(b) Accumulative Learning Strategy

Fig. 2. Self-StockTime Model: (a) Multimodal fusion of patch-based LSTMs and ticker embeddings. (b) Accumulative learning via modular weight transfer and projection head resets for temporal regime adaptation.

Predictive DQN Architecture. The Q-network $Q(s, a; \theta_Q) : \mathbb{R}^{h+4} \to \mathbb{R}^{|A|}$ consists of projection blocks (Linear $\to$ LayerNorm $\to$ ReLU $\to$ Dropout), trained with experience replay, target network, and ϵ-greedy exploration.

4) Cascaded Walk-Forward Training: A critical challenge in two-stage pipelines is *data leakage*: if the prediction model is trained on the entire dataset prior to RL optimization, the agent receives "look-ahead" forecasts from a model that has already observed the future. This results in inflated training performance and catastrophic generalization failure.

Therefore, we introduce a cascaded walk-forward protocol where both models progress through temporal folds together, ensuring the agent only sees forecasts generated from strictly past data (Fig. 3). For each fold k:

1) Train forecaster on window $[t_{\text{start}}^{(k)}, t_{\text{train}}^{(k)}]$
2) Generate predictions for unseen window $(t_{\text{train}}^{(k)}, t_{\text{RL}}^{(k)}]$
3) Train RL agent on states from these out-of-sample forecasts

The process slides forward: each fold incorporates the previous fold's RL window into forecaster training, and forecasts are pre-computed and cached for efficiency. This ensures zero look-ahead bias, realistic training conditions (forecasts reflect actual model errors, preventing overfitting to idealized predictions), and full data utilization.

B. Stream B: Trend Prediction with Threshold-Based Trading

Stream B predicts directional movement (UP/DOWN) as binary classification over horizon h, using optimized confidence thresholds for trading decisions.

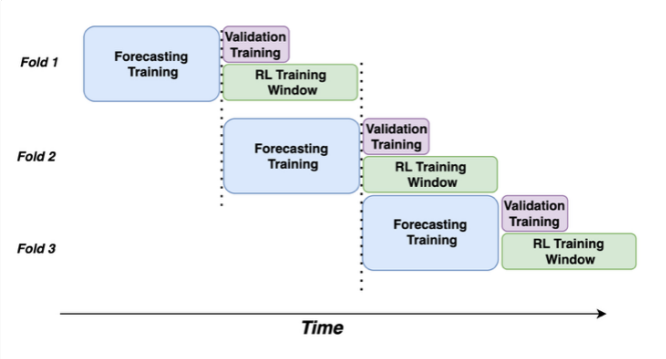


Fig. 3. Parallel training ensures RL agent only sees out-of-sample forecasts, eliminating data leakage.

Pipeline. For each stock i at time t, we define the target $y_{t,h}^{(i)} = \mathbb{1}[P_{t+h}^{(i)} > P_t^{(i)}]$ and train the model to output confidence $\hat{p}_{t,h}^{(i)} = f_\phi(\mathbf{x}_t^{(i)}) \in [0,1]$, enabling per-stock threshold requirements. This probabilistic output serves as a confidence-weighted signal, where trading execution is decoupled from the model's raw prediction and instead governed by a risk-adjusted decision layer.

Feature Engineering. We construct a feature vector $\mathbf{x}_t^{(i)}$ for each stock-day pair:

1) Multi-timeframe returns $r_{t,\delta}^{(i)} = (P_t^{(i)} - P_{t-\delta}^{(i)})/P_{t-\delta}^{(i)}$ for $\delta \in \mathcal{T}$ days.
2) Cross-stock features $\mathbf{x}_{\text{cross}}^{(i)} = [r_{t,\delta}^{(1)}, \ldots, r_{t,\delta}^{(M)}]_\delta$ capturing inter-asset dependencies.
3) Market regime indicators (rolling volatility, trend).
4) Stock identity $\mathbf{e}^{(i)} \in \{0,1\}^M$.

1) TabNet for Trend Prediction.: We adopt TabNet [26] for binary trend classification. The model processes features through sequential decision steps with sparse attention masks $\mathbf{M}^{[n]}$, aggregating representations via:

$$\hat{p} = \sigma(\mathbf{w}^\top \sum_{n=1}^{N_{\text{steps}}} \mathbf{h}^{[n]} + b) \tag{9}$$

where $\mathbf{h}^{[n]} = \text{FeatureTransformer}(\mathbf{M}^{[n]} \odot \mathbf{x})$. The attention masks provide interpretability, revealing which features (e.g., 5-day return of AAPL, 21-day return of SPY) drive each prediction. Unlike multi-step trajectory models, this framework predicts only the probability of profit at day $t+h$, ignoring intermediate price paths. This enables per-stock threshold optimization in the trading strategy.

Objective. We train a single model across all stocks, leveraging shared market dynamics while maintaining stock identity via one-hot encoding. The objective minimizes cross-entropy:

$$\mathcal{L}(\phi) = -\sum_{i=1}^{M}\sum_{t=1}^{T}\left[y_t^{(i)} \log \hat{p}_t^{(i)} + (1 - y_t^{(i)}) \log(1 - \hat{p}_t^{(i)})\right] \tag{10}$$

2) Threshold-Based Trading Strategy: The probabilistic output of TabNet enables a simple trading strategy: enter positions only when model confidence exceeds a stock-specific threshold $\tau^{(i)}$. Different stocks exhibit different predictability and risk profiles, so rather than using a universal threshold, we optimize per-stock thresholds $\tau^{(i)*}$ to maximize historical returns.

Threshold Optimization. For each stock i, we search over candidate thresholds $\tau \in \mathcal{T}_c$ and simulate trading on an optimization period:

$$\tau^{(i)*} = \arg\max_{\tau \in \mathcal{T}_c} \text{Return}_{\text{opt}}^{(i)}(\tau) \tag{11}$$

Portfolio Management Policy. The policy $\pi^{(i)}$ follows: (i) entry when $\hat{p}_t^{(i)} > \tau^{(i)*}$ with UP prediction, (ii) exit when prediction is DOWN or maximum holding period exceeded, and (iii) abstain from trading stocks where no profitable threshold exists. The optimized thresholds $\{\tau^{(i)*}\}$ are deployed on subsequent unseen data.

V. Experimental Setup

We evaluate both streams on S&P 500 historical data spanning January 2000 to January 2024, obtained via Yahoo Finance API, with daily OHLCV features. To investigate scaling effects, we experiment with $M = 19$ and $M = 99$ stocks, randomly selected across diverse sectors (technology, finance, healthcare, consumer, energy, industrials).

Baselines. For Stream A, we compare against PatchTST [18], a state-of-the-art transformer for time series. For Stream B, we evaluate XGBoost [41], LSTM, and TabNet [26] under both local (per-stock) and global training strategies. Trading performance is evaluated by annual returns against an S&P 500 buy-and-hold benchmark.

A. Model Configurations

Hyperparameters are optimized via grid search over the parameter space. Below we detail the configuration for each stream. All experiments use a transaction cost of $c = 0.1\%$. All experiments were conducted on Google Colab with NVIDIA H100/A100/ GPUs.

Stream A Setting. We use WFV with $K = 4$ folds (5-year training, 1-year validation per fold). Self-StockTime uses Llama 3.1 8B (8-bit quantized) as the LLM backbone, a 2-layer LSTM encoder (hidden size 256), and 4-head attention fusion with shared dimension $d = 768$. The prediction head projects 768→512→256. We use lookback $p = 28$ days and horizon $h = 7$ days, optimized with AdamW (lr=10^{-4}). The DQN agent uses an 11-dimensional state (including 20-day rolling Z-score and RSI $\in [-1, 1]$) with hidden layers [64, 32]. Training runs 500 episodes per fold, target network updates every 1000 steps, and ϵ decaying linearly from 1.0 to 0.01.

Stream B Setting. We implement a Global TabNet with $N_{\text{steps}} = 5$ decision steps and feature/attention dimensions of 64. The input vector $\mathbf{x}_t^{(i)}$ includes multi-timeframe returns ($\delta \in \{1, 5, 10, 21\}$ days) across all M stocks and one-hot stock identity $\mathbf{e}^{(i)}$. Training minimizes binary cross-entropy using Adam (lr=2×10^{-2}) with ReduceLROnPlateau scheduling. We use temporal splits: training (pre-2020), threshold optimization (2021–2022), and validation (2023). Per-stock thresholds are

optimized via grid search over $\tau \in \{0.55, 0.60, \ldots, 0.80\}$. The policy enters when $\hat{p}_t^{(i)} > \tau^{(i)*}$ (UP prediction), exits on DOWN prediction or after 21 days, with maximum 80% capital per position.

VI. Results

We measure prediction performance via MSE (Stream A) and directional accuracy (Stream B). Trading performance is evaluated by annual returns against an S&P 500 buy-and-hold benchmark.

A. Prediction Performance

Multi-Step Forecasting (Stream A). We focus on evaluating robustness across market regimes rather than comparative analysis. Table I shows consistent performance across walk-forward folds, with Fold 3 (COVID-19) exhibiting expected degradation. Self-StockTime achieves competitive MSE relative to time series baselines (e.g., PatchTST, LSTM), though comparative analysis is not our primary focus. Scaling from 19 to 99 stocks improved prediction MSE by 21% and nearly doubled RL returns (7.8% $\rightarrow$ 14.8%). The expanded stock universe provides richer cross-asset patterns, while additional episodes enable better exploration of the state-action space. Fig. 4 demonstrates the benefit of accumulative learning: Fold 0 (from scratch) requires multiple epochs to converge, whereas Fold 2 (with transferred weights) starts at $\sim$9$\times$ lower MSE and remains stable.

TABLE I
Self-StockTime Validation MSE

Fold	Validation Period	$M = 19$	$M = 99$
0	01/2005 – 12/2005	0.000314	**0.000236**
1	01/2010 – 12/2010	0.000501	**0.000336**
2	01/2015 – 12/2015	0.000359	**0.000284**
3	01/2020 – 12/2020	0.001127	**0.001098**
Average		0.000575	**0.000488**

Directional Classification (Stream B). Directional Prediction Performance We compare three model families for directional price prediction: XGBoost, LSTM-based sequence models, and TabNet. Each model is evaluated under two training strategies: (i) *Local*, where a separate model is trained per stock, and (ii) *Global*, where a single model is trained jointly across all stocks using shared features and stock identifiers. Table II reports the average directional accuracy across stocks for prediction horizons of 7–30 trading days. Global training improves performance across all model families, confirming the presence of shared market structure across assets. While both XGBoost and LSTM benefit from global pooling, TabNet achieves the highest average accuracy and the strongest performance at longer horizons. In addition, TabNet provides built-in feature selection and interpretability via attentive masks, making it better suited for financial decision-making. For these reasons, we adopt **Global TabNet** as the backbone of Stream B. For TabNet model, increasing from $M = 19$ to $M = 99$ stocks slightly decreased accuracy (77.8% $\rightarrow$ 75.5% at 21-day horizon), suggesting that larger stock universes do not universally improve performance. For TabNet, accuracy slightly decreased from 77.8% to 75.5% when scaling to $M = 99$. While training on more stocks increases data volume, performance remains dependent on individual stock characteristics, and results are comparable across both settings.

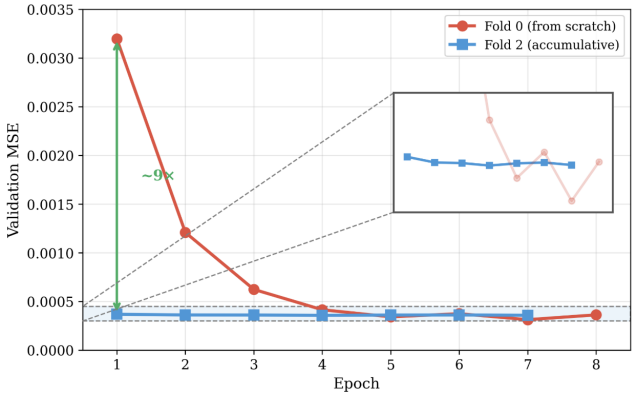


Fig. 4. Self-StockTime Training convergence: Fold 2 starts at 10$\times$ lower MSE than Fold 0 due to accumulative weight transfer.

TABLE II
Directional Accuracy (%) by Model, Training Strategy, and Prediction Horizon

Days	XGB (L)	LSTM (L)	TabNet (L)	LSTM (G)	XGB (G)	TabNet (G)
7	55.6	54.7	56.1	65.2	71.9	**70.0**
14	60.2	57.1	56.9	75.0	73.5	**73.6**
21	61.2	59.5	57.5	74.3	75.1	**77.9**
30	60.5	60.1	57.2	73.6	74.3	**76.7**
Avg	59.1	57.9	56.9	72.0	73.7	**74.6**

B. Trading Performance

Table IV compares both strategies against the S&P 500 benchmark. Stream B achieved 40.0% annual return (+65% vs. benchmark), while Stream A achieved 14.8% (–39%). As shown in Table III, RL performance improved substantially when scaling from $M = 19$ to $M = 99$ stocks (7.8% $\rightarrow$ 14.8%), benefiting from richer cross-asset patterns and additional exploration.

The 2.7$\times$ performance gap yields key insights: (1) directional prediction proved more tractable than multi-step forecasting; (2) explicit threshold rules outperformed learned policies in data-limited regimes; (3) prediction errors compound in RL state representations; (4) cross-stock learning via the Global Model captures transferable market patterns.

TABLE III
RL Trading Performance: 19 vs. 99 Stocks

Metric	19 Stocks	99 Stocks
Episodes per Fold	150	500
Annual Return	7.8%	**14.8%**
Win Rate	64.3%	71.8%

TABLE IV
TRADING PERFORMANCE COMPARISON (2023 TEST PERIOD)

Strategy	Annual Return	vs. Benchmark
S&P 500 Buy & Hold	24.2%	—
Stream A: RL Agent (DQN)	14.8%	–39%
Stream B: Threshold Model	**40.0%**	+65%

VII. DISCUSSION

The performance gap between streams (40% vs. 14.8%) highlights important considerations for applying machine learning to financial trading. Our findings suggest: (i) directional prediction may be preferable when trading decisions are the end goal; (ii) global models effectively capture cross-stock patterns; (iii) simple threshold strategies provide strong baselines before deploying complex agents; (iv) temporal validation protocols are essential to prevent data leakage.

Prediction Task. Directional classification (Stream B) reduces the target space to a binary signal, offering robustness to the noise inherent in financial time series. Multi-step regression (Stream A) provides richer predictions but amplifies errors across the forecast horizon.

Architecture Design. Stream B's modular approach separates prediction (TabNet) from decision-making (threshold optimization), enabling independent validation. Stream A's end-to-end RL must learn both market dynamics and trading behavior jointly, increasing learning complexity.

Data Requirements. Threshold-based strategies require minimal parameters per stock, whereas deep RL typically benefits from larger sample sizes than daily financial data provides.

Limitations Several factors constrain our findings: (1) experiments were conducted on a single GPU, limiting RL exploration to 500 episodes per fold, additional compute may improve Stream A performance; (2) evaluation on 2023 reflects specific market conditions; (3) results are limited to S&P 500 large-cap equities; (4) higher transaction costs would impact both strategies differently.

VIII. CONCLUSION

In this work, we presented a predictive-based framework for financial forecasting and automated trading. We addressed the challenges of market noise and non-stationarity by decoupling forecasting from decision-making through two distinct modular pipelines: Return Regression (Stream A), which utilizes multi-step continuous forecasts, and Directional Classification (Stream B), which leverages probabilistic trend movement Our work shows that simplicity wins: threshold-based strategies built on trend probabilities outperform complex regression models across portfolios of varying sizes, suggesting that directional classification is better suited to the data constraints of financial forecasting. Our results highlight the need for predictive pipelines that jointly optimize forecasting and decision-making under risk, an approach that mirrors real-world trading logic but remains largely unexplored in current research.

This work opens the door to future research on uncertainty-aware trading and predictive RL exploration. Future work could develop joint models that refine trading actions based on forecast confidence, compare modular predictive architectures, and investigate alternative RL algorithms alongside extend data sources.

REFERENCES

[1] M. S. M. Bhuiyan, M. A. Rafi, G. N. Rodrigues, M. N. H. Mir, A. Ishraq, M. Mridha, and J. Shin, "Deep learning for algorithmic trading: A systematic review of predictive models and optimization strategies," *Array*, p. 100390, 2025.

[2] J. Wang, T. Sun, B. Liu, Y. Cao, and D. Wang, "Financial markets prediction with deep learning," in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2018, pp. 97–104.

[3] S. Ravuri, K. Lenc, M. Willson, D. Kangin, R. Lam, P. Mirowski, M. Fitzsimons, M. Athanassiadou, S. Kashem, S. Madge *et al.*, "Skilful precipitation nowcasting using deep generative models of radar," *Nature*, vol. 597, no. 7878, pp. 672–677, 2021.

[4] I. Kadota, D. Jacoby, H. Messer, G. Zussman, and J. Ostrometzky, "Switching in the rain: Predictive wireless x-haul network reconfiguration," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 6, no. 3, pp. 1–26, 2022.

[5] X.-Y. Liu, Z. Xia, J. Rui, J. Gao, H. Yang, M. Zhu, C. Wang, Z. Wang, and J. Guo, "Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1835–1849, 2022.

[6] B. Amiri, A. Haddadi, and K. F. Mojdehi, "A novel hybrid gcn-lstm algorithm for energy stock price prediction: leveraging temporal dynamics and inter-stock relationships," *IEEE Access*, 2025.

[7] M. Beniwal, A. Singh, and N. Kumar, "Forecasting multistep daily stock prices for long-term investment decisions: A study of deep learning models on global indices," *Engineering Applications of Artificial Intelligence*, vol. 129, p. 107617, 2024.

[8] S. Wang, T. Ji, L. Wang, Y. Sun, S. Liu, A. Kumar, and C. Lu, "StockTime: A time series specialized large language model architecture for stock price prediction," *arXiv preprint arXiv:2409.08281*, 2024.

[9] H. Wasserbacher and M. Spindler, "Machine learning for financial forecasting, planning and analysis: recent developments and pitfalls," *Digital Finance*, vol. 4, no. 1, pp. 63–88, 2022.

[10] D. M. Nelson, A. C. Pereira, and R. A. De Oliveira, "Stock market's price movement prediction with lstm neural networks," in *2017 International joint conference on neural networks (IJCNN)*. Ieee, 2017, pp. 1419–1426.

[11] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, "Stock price prediction using lstm, rnn and cnn-sliding window model," in *2017 international conference on advances in computing, communications and informatics (icacci)*. IEEE, 2017, pp. 1643–1647.

[12] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *International journal of forecasting*, vol. 36, no. 1, pp. 75–85, 2020.

[13] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International journal of forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[14] A. Sbrana, A. L. D. Rossi, and M. C. Naldi, "N-beats-rnn: deep learning for time series forecasting," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2020, pp. 765–768.

[15] W. Chen, W. Hussain, F. Cauteruccio, and X. Zhang, "Deep learning for financial time series prediction: A state-of-the-art review of standalone and hybrid models," 2024.

[16] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical transactions of the royal society a: mathematical, physical and engineering sciences*, vol. 379, no. 2194, 2021.

[17] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[18] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *International Conference on Learning Representations*, 2023.

[19] Y. Wang, C. Liu, K. Liu, and D. D. Sun, "Sat: A sparse attention transformer-based approach for predicting market risk factors using stock time series data," in *2025 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2025, pp. 1–8.

[20] H. Qian, H. Zhou, Q. Zhao, H. Chen, H. Yao, J. Wang, Z. Liu, F. Yu, Z. Zhang, and J. Zhou, "Mdgnn: Multi-relational dynamic graph neural network for comprehensive and dynamic stock investment prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 13, 2024, pp. 14 642–14 650.

[21] H. Xia, H. Ao, L. Li, Y. Liu, S. Liu, G. Ye, and H. Chai, "Ci-sthpan: Pre-trained attention network for stock selection with channel-independent spatio-temporal hypergraph," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 8, 2024, pp. 9187–9195.

[22] T. Li, Z. Liu, Y. Shen, X. Wang, H. Chen, and S. Huang, "Master: Market-guided stock transformer for stock price forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 162–170.

[23] S. Wang, T. Ji, L. Wang, Y. Sun, S.-C. Liu, A. Kumar, and C.-T. Lu, "Stocktime: A time series specialized large language model architecture for stock price prediction," *arXiv preprint arXiv:2409.08281*, 2024.

[24] H. Tong, J. Li, N. Wu, M. Gong, D. Zhang, and Q. Zhang, "Ploutos: Towards interpretable stock movement prediction with financial large language model," *arXiv preprint arXiv:2403.00782*, 2024.

[25] A. F. Ansari *et al.*, "Chronos: Learning the language of time series," *arXiv preprint arXiv:2403.07815*, 2024.

[26] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 8, 2021, pp. 6679–6687.

[27] Y. Lv, S. Guo, Y. Chen, and W. Li, "Stock volatility prediction using tabnet based deep learning method," in *2022 3rd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. IEEE, 2022, pp. 665–668.

[28] M. R. Vargas, C. E. Dos Anjos, G. L. Bichara, and A. G. Evsukoff, "Deep leaming for stock market prediction using technical indicators and financial news articles," in *2018 international joint conference on neural networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[29] B. G. Lim, J. Liu, H. J. Ong, J. A. Chan, R. R. Tan, I. King, and K. Ikeda, "Finsir: Financial sir-gcn for market-aware stock recommendation," in *2025 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2025, pp. 1–8.

[30] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[31] D. Snow, "Machine learning in asset management—part 1: Portfolio construction—trading strategies," *The Journal of Financial Data Science*, vol. 2, no. 1, pp. 10–23, 2020.

[32] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules," *Journal of financial Economics*, vol. 51, no. 2, pp. 245–271, 1999.

[33] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE transactions on neural Networks*, vol. 12, no. 4, pp. 875–889, 2001.

[34] S. Sun, R. Wang, and B. An, "Reinforcement learning for quantitative trading," *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 3, pp. 1–29, 2023.

[35] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2016.

[36] M. Tran, D. Pham-Hi, and M. Bui, "Optimizing automated trading systems with deep reinforcement learning," *Algorithms*, vol. 16, no. 1, p. 23, 2023.

[37] Y. Bai, Y. Gao, R. Wan, S. Zhang, and R. Song, "A review of reinforcement learning in financial applications," *Annual Review of Statistics and Its Application*, vol. 12, no. 1, pp. 209–232, 2025.

[38] Z. Wang, B. Huang, S. Tu, K. Zhang, and L. Xu, "Deeptrader: a deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 1, 2021, pp. 643–650.

[39] H. Niu, S. Li, and J. Li, "Metatrader: An reinforcement learning approach integrating diverse policies for portfolio optimization," in *Proceedings of the 31st ACM international conference on information & knowledge management*, 2022, pp. 1573–1583.

[40] A. Sattar, A. Sarwar, S. Gillani, M. Bukhari, S. Rho, and M. Faseeh, "A novel rms-driven deep reinforcement learning for optimized portfolio management in stock trading," *IEEE Access*, 2025.

[41] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.