HyperMamba: A Hypernetwork-Enhanced Meta-Learning Framework for Autonomous Trading Systems

Integrating Lessons from CryptoMamba, HyperMAML, and State-Space Approaches

Affaan Mustafa
University of Washington
Email: affoon@uw.edu

***Abstract*—The growing complexity and volatility of financial markets require trading agents that can learn, adapt, and generalize with minimal data. We introduce HyperMamba, a novel meta-learning framework that extends the Mamba algorithm with hypernetwork-based weight generation to provide rapid policy adaptation. Inspired by *HyperMAML*, which leverages Hypernetworks for more substantial gradient updates, and by the *CryptoMamba* approach for robust state-space modeling in cryptocurrency markets, HyperMamba aims to deliver an end-to-end, autonomous trading pipeline that can ingest diverse data streams, generate accurate predictions, and execute policies that adapt in real time.**

We present the overarching system architecture, an in-depth algorithmic description of HyperMamba, and proposed experimental setups spanning Solana blockchain data, sentiment APIs, macroeconomic indicators, and risk management protocols. While no live or simulation-based experiments have been conducted to date, we outline future HPC-based testing methodologies (e.g., multi-GPU clusters, distributed data loading) to rigorously evaluate the framework's adaptability and performance. We also highlight potential risk management enhancements (e.g., dynamic hedging, drawdown controls) and discuss how state-space models (SSMs) can be integrated to efficiently handle long time-series data in both training and inference.

***Index Terms*—Meta-Learning, Hypernetworks, Model-Based Reinforcement Learning, Autonomous Trading, State Space Models, Solana Blockchain**

I. Introduction

In autonomous trading, *adaptation speed* to new tasks or regimes is paramount. Rapidly changing market conditions, from sudden volatility to emergent macroeconomic factors, can devastate rigid models. While model-free RL systems often require substantial experience before they adapt, meta-learning approaches aim to give agents the ability to adapt with minimal data from new tasks.

Several frameworks have shown promise in this space. The original *Mamba* algorithm leverages model-based meta-RL for efficient adaptation. Building on that, *HyperMAML* employs a hypernetwork to generate parameter updates, allowing for more substantial weight shifts than standard gradient-based fine-tuning. Separately, *CryptoMamba* demonstrated that specialized Mamba-based state-space models can excel in forecasting highly volatile cryptocurrency prices by handling long-range dependencies.

Our contribution, **HyperMamba**, synthesizes these advances. We adapt Mamba with a hypernetwork-based update mechanism, akin to HyperMAML, to facilitate large and efficient changes in model parameters. We further embrace lessons from CryptoMamba's success in cryptocurrency forecasting, particularly with regards to state-space modeling, risk management, and multi-source data ingestion. The result is a comprehensive pipeline for *autonomous trading systems* that unifies data ingestion, prediction, decision-making, execution, and feedback loops.

II. Overall Autonomous Trading System

Figure 1 illustrates the proposed end-to-end architecture for autonomous trading agents, showing how HyperMamba integrates into a broader system:

- **Data Ingestion Layer**: Streams data from *Solana blockchain*, *Sentiment Analysis APIs*, and *Macroeconomic Indicators*, normalizing them into consistent flows.
- **Prediction Module**: Applies advanced neural models (e.g., Transformer-based LLM or Temporal Convolutional Network) or meta-learning modules (HyperMamba) to forecast asset prices or expected returns.
- **Decision-Making Module**: Uses meta-RL approaches (e.g., Hierarchical Meta-Learning, HyperMAML Refinement, or Proximal Policy Optimization) to select optimal actions based on forecasted states.
- **Execution Layer**: Executes trades on-chain via *Smart Contracts on Solana*, applying *Risk Management* constraints.
- **Feedback & Adaptation Layer**: Includes a backtesting engine and a real-time feedback loop to refine HyperMamba parameters and risk thresholds continuously.

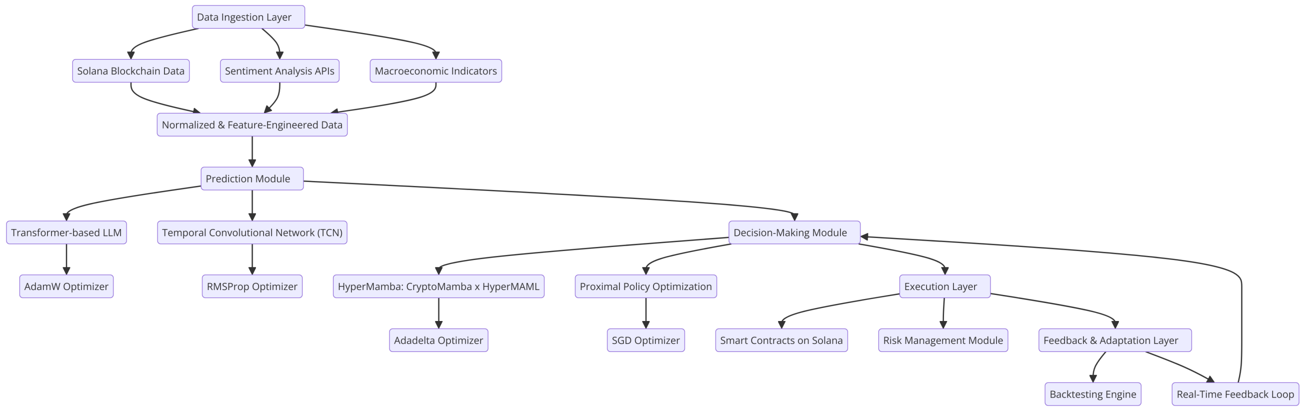

Fig. 1: Proposed End-to-End Autonomous Trading System Architecture. HyperMamba integrates within the *Prediction* and *Decision-Making* modules, leveraging data ingestion and feedback loops for continuous adaptation.

III. Related Work

A. Mamba

Mamba is a model-based meta-RL algorithm that learns world models to achieve fast adaptation across multiple tasks. It pioneered using a state-space representation for multi-step reward optimization. However, Mamba's reliance on simple gradient-based updates can hinder rapid adaptation in tasks needing large parameter shifts.

B. Hypernetworks & HyperMAML

Hypernetworks, introduced by Ha *et al.*, generate the weights for another network dynamically, achieving faster adaptation in few-shot learning. *HyperMAML* [1] extends the MAML framework with a Hypernetwork, enabling the meta-learner to output task-specific weight initializations. This design leads to more substantial and context-sensitive updates, especially relevant for volatile domains like finance.

C. CryptoMamba

Recently, *CryptoMamba* [2] showed how a Mamba-based state-space architecture can excel at forecasting Bitcoin prices and, by extension, other cryptocurrencies. By incorporating volume data, advanced regularization, and specialized hyperparameters, CryptoMamba demonstrated superior accuracy and robust real-world trading outcomes. Though CryptoMamba is primarily a forecasting solution (rather than a fully meta-learning RL system), its approach to modeling volatility with SSM layers informs aspects of HyperMamba's design.

IV. HyperMamba Algorithm and Architecture

A. HyperMamba Overview

HyperMamba merges Mamba's world modeling with hypernetwork-based parameter generation. We treat each new "task" as a shift in market conditions (e.g., volatility, changes in macro indicators). Unlike standard Mamba, we allow more flexible, larger-scale updates via a hypernetwork that outputs adaptation steps.

B. Hypernetwork-Enhanced Updates

Let θ be the base network parameters (analogous to Mamba's learned model parameters) and let ϕ be the hypernetwork parameters. For a new task $\mathcal{T}$, the hypernetwork generates a task-specific set of parameters $\theta_{\mathcal{T}}$:

$$\theta_{\mathcal{T}} = H_{\phi}\big(\text{embedding}(\mathcal{T})\big),$$

where H_{ϕ} is the hypernetwork, and embedding$(\mathcal{T})$ captures relevant features from the market shift (e.g., sentiment, volatility regime, etc.).

C. Algorithm

Algorithm IV-C illustrates the meta-training loop for HyperMamba, inspired by *HyperMAML* and adapted to Mamba.

[!t] **Require:** Distribution over tasks $p(\mathcal{T})$, step size α, hypernetwork H_{ϕ}, base model f_{θ}

1) **Initialize** ϕ (hypernetwork), θ (base model).
2) **while** not converged **do**
   a) Sample batch of tasks $\{\mathcal{T}_i\}_{i=1}^{B} \sim p(\mathcal{T})$
   b) **For each** $\mathcal{T}_i$ **in** batch:
      - Obtain task embedding $e_{\mathcal{T}_i}$
      - Generate adapted parameters: $\theta_i \leftarrow H_{\phi}(e_{\mathcal{T}_i})$
      - Compute support loss $\mathcal{L}_{\text{support}}(f_{\theta_i}, \mathcal{T}_i)$
      - Gradient-based refinement on θ_i:

$$\theta_i' \leftarrow \theta_i - \alpha \nabla_{\theta_i} \mathcal{L}_{\text{support}}(f_{\theta_i}, \mathcal{T}_i)$$

      - Evaluate query loss $\mathcal{L}_{\text{query}}(f_{\theta_i'}, \mathcal{T}_i)$
   c) Aggregate meta-loss:

$$\mathcal{L}_{\text{meta}} \leftarrow \sum_{i=1}^{B} \mathcal{L}_{\text{query}}(f_{\theta_i'}, \mathcal{T}_i)$$

   d) Update hypernetwork parameters:

$$\phi \leftarrow \phi - \beta \nabla_{\phi} \mathcal{L}_{\text{meta}}$$

3) **end while**

Key Differences from Standard Mamba:

1) Instead of using a single set of parameters for adaptation, we rely on a hypernetwork to generate θ_i per task.
2) We incorporate a brief gradient-based refinement step (similar to MAML) to allow minor tweaks.
3) The meta-update (step 2(g)) optimizes ϕ, i.e. the hypernetwork.

State-Space Extension (Optional): HyperMamba can optionally incorporate a *State-Space Model* (SSM) layer or module inside f_θ to handle long time-series efficiently. This is motivated by *CryptoMamba* [2], where SSM-based architectures showed strong performance in non-stationary crypto environments. When used here, the SSM simply becomes another component of f_θ; the hypernetwork can generate or fine-tune certain SSM parameters based on the current market regime, thus improving the agent's ability to deal with extended temporal dependencies.

V. Proposed Experiments & Evaluation

A. Data Streams

a) Blockchain Data.: Solana-based historical trading pairs for various tokens (normalized to consistent intervals).

b) Sentiment Analysis.: Real-time sentiment from social media or news APIs (tokenized, aggregated per time interval).

c) Macroeconomic Indicators.: FX rates, interest rates, CPI indices, yield curves, etc.

All data are intended to be segmented into mini "tasks" reflecting shifts in volatility, major news cycles, or macro events.

B. Experimental Setup

While no experiments have been conducted at this time, we plan for the following:

- **Base Model Configurations**: Evaluate Transformer-based vs. TCN vs. SSM to see which architecture best complements the hypernetwork adaptation mechanism in volatile market conditions.
- **Baselines**:
  1) *Vanilla Mamba* (no hypernetwork)
  2) *HyperMAML* in a purely model-free RL setup (omitting Mamba's world model)
  3) *CryptoMamba*-style forecast model for direct predictions (not a full RL pipeline, but a strong forecasting baseline)
- **Metrics**:
  – *Trading performance*: Return, drawdown, Sharpe ratio, Sortino ratio
  – *Adaptation efficiency*: Time or updates needed to recover performance after abrupt regime changes
  – *Forecast error* (if relevant): RMSE, MAPE

C. Risk Management and Execution Layer (Proposed)

For future testing, we plan to:

1) Incorporate a **Risk Management Module** that enforces drawdown-based halts and optional dynamic hedging:
   - *Drawdown limit*: If equity drops by X%, the system automatically reduces risk or exits positions.
   - *Dynamic hedging*: A side RL agent or rule-based engine might open offsetting positions in futures or options to cap downside.
2) **On-Chain Execution**: Deploy a prototype smart contract on Solana (written in Rust) to handle decentralized trades, verifying that off-chain HyperMamba signals align with on-chain risk checks.
3) **Front-Running Mitigation**: Consider randomizing transaction submission times or using specialized pools that reduce MEV exposure.

D. High-Performance Computing (HPC) Environment

To rigorously evaluate HyperMamba at scale, we plan to follow HPC guidelines similar to those in CryptoMamba:

- **GPU Configuration**:
  1) 4–8 NVIDIA A100 GPUs (80GB memory each) for parallel training and large batch processing.
  2) Use PyTorch Distributed or similar frameworks (e.g., Horovod) for data parallelism across GPUs.
- **CPU and Memory**:
  1) High-core-count CPU (e.g., dual AMD EPYC) with at least 256GB RAM for large dataset buffering.
  2) High-speed NVMe SSDs to store historical time-series and sentiment data for quick streaming to GPUs.
- **Scalability Considerations**:
  1) Perform *distributed data loading* across nodes to manage large-scale training sets.
  2) *Mixed-precision* or BF16 training for speed-ups on A100 Tensor Cores.
  3) If the hypernetwork or the state-space model grows very large, investigate model-parallel strategies across multiple GPUs.
- **Real-Time Inference**:
  1) Maintain a streamlined inference service on a dedicated GPU or CPU for sub-second predictions.
  2) Off-chain services will handle data pre-processing (e.g., sentiment or on-chain analytics) before forwarding features to the inference engine.

VI. Presentation Slide Highlights

We distill the key points for future slides or educational materials:

- **Motivation Slide**: Outline limitations of static RL algorithms, significance of meta-learning, and the need for large parameter shifts in finance.
- **Architecture Slide**: Show the full end-to-end system (Fig. 1), emphasizing data ingestion, meta-learning pipeline, execution, and feedback.
- **Algorithm Slide**: Summarize Algorithm IV-C, highlighting the synergy between Mamba's model-based approach and hypernetwork generation of parameters for rapid adaptation.

- **Implementation Slide (HPC)**: Provide the HPC architecture for training (multi-GPU, distributed data, memory requirements). Mention how to integrate large-scale historical data streams (e.g., multiple years of crypto or equity data).
- **Risk Management Slide**: Propose additional modules for drawdown control, hedging, and adversarial robustness in manipulated or volatile markets.

VII. Conclusion and Future Directions

We have introduced **HyperMamba**, a hypernetwork-enhanced, model-based meta-RL system inspired by *HyperMAML*, *Mamba*, and conceptual innovations from *CryptoMamba*. By integrating state-space elements (SSMs), advanced data ingestion, and a pathway to on-chain execution, HyperMamba is designed to adapt quickly in non-stationary financial environments.

No live or simulation-based experiments have yet been performed, so we cannot make claims about HyperMamba's practical performance. However, the methodology outlined herein—including high-performance computing setups, risk management integration, and multi-task data segmentation—provides a robust roadmap for future empirical validation. In particular:

- **Realistic Backtesting**: We plan to run large-scale backtests on multi-year datasets (crypto and traditional) to measure adaptation speed, risk-adjusted returns, and drawdown management.
- **Live Paper Trading**: Before committing real capital, a live simulation environment will test the model in real-time data conditions, ensuring HPC-based inference can keep pace with market updates.
- **Solana On-Chain Prototype**: We will prototype decentralized execution for selected crypto-assets, exploring how fast the system can adapt in DeFi markets and how risk controls can be enforced via smart contracts.
- **Extended Alternative Data**: Future expansions include advanced sentiment analysis (e.g., Twitter, Reddit), on-chain analytics for whales' movements, and real-time macroeconomic updates for multi-asset strategies.

Ultimately, **HyperMamba** aims to push the boundaries of autonomous trading by melding cutting-edge meta-learning, robust HPC architectures, and risk-aware on-chain execution. Though the framework remains untested in practice, the theoretical foundations and design considerations presented here lay the groundwork for a next-generation adaptive trading agent capable of thriving in complex, fast-evolving markets.

References

[1] M. Przewięźlikowski, P. Przybysz, J. Tabor, M. Zięba, and P. Spurek, "HyperMAML: Few-Shot Adaptation of Deep Models with Hypernetworks," *arXiv preprint arXiv:2205.15745*, 2022.

[2] M. S. Sepehri, A. Mehradfar, M. Soltanolkotabi, and S. Avestimehr, "CryptoMamba: Leveraging State Space Models for Accurate Bitcoin Price Prediction," *arXiv preprint arXiv:2501.01010*, 2025.

[3] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.