ALPHA: Advanced Learning for Portfolio Handling Applications

Benjamin CORIAT[1] , **Eric BENHAMOU**[2,3]
[1]CentraleSupelec, [2]Ai for Alpha, [3] Dauphine PSL

Abstract

This paper presents a novel hierarchical framework for portfolio optimization, integrating lightweight Large Language Models (LLMs) with Deep Reinforcement Learning (DRL) to combine sentiment signals from financial news with traditional market indicators. Our three-tier architecture employs base RL agents to process hybrid data, meta-agents to aggregate their decisions, and a super-agent to merge decisions based on market data and sentiment analysis. Evaluated on data from 2018 to 2024, after training on 2000–2017, the framework achieves a 26% annualized return and a Sharpe ratio of 1.2, outperforming equal-weighted and S&P 500 benchmarks. Key contributions include scalable cross-modal integration, a hierarchical RL structure for enhanced stability, and open-source reproducibility thanks to Google Collab notebooks.

1 Introduction

The integration of Large Language Models (LLMs) and Reinforcement Learning (RL) offers a powerful approach to financial portfolio optimization, leveraging LLMs' ability to process unstructured data and RL's strength in sequential decision-making. Domain-specific LLMs like FinBERT [Araci, 2019] extract nuanced sentiment signals from financial news, capturing market sentiment and investor behavior critical for anticipating price movements [Tetlock, 2007]. Meanwhile, RL enables adaptive strategies in dynamic markets characterized by feedback loops and regime shifts [Jiang *et al.*, 2017].

Recent studies highlight the efficacy of LLM-RL hybrids, with sentiment-enhanced RL models outperforming traditional RL in single-stock trading and portfolio management. These models integrate qualitative signals from news with quantitative metrics, improving risk-adjusted returns [Unnikrishnan and others, 2024]. For instance, news-driven RL frameworks leverage textual cues to enhance decision-making, demonstrating the value of cross-modal integration [Xu and Zhou, 2018].

Despite these advances, many LLM-RL approaches rely on single-modal or flat architectures, which limit their ability to fully exploit textual and numerical data. Single-modal systems, using only price data or sentiment scores, struggle to capture the multidimensional nature of financial markets, leading to suboptimal decisions in volatile conditions [Li *et al.*, 2021]. Flat architectures, as seen in early RL trading systems [Deng *et al.*, 2016], lack scalability and interpretability for complex portfolios, often resulting in unstable policies or overfitting.

To overcome these limitations, we propose a hierarchical portfolio management framework combining Deep Reinforcement Learning (DRL) with lightweight, domain-specific LLMs like FinBERT [Araci, 2019]. The framework creates a hybrid observation space by integrating sentiment scores with traditional financial indicators. It employs a three-layer hierarchy: base RL agents process raw data, meta-agents aggregate base-level decisions, and a super-agent synthesizes cross-modal signals to optimize portfolio allocations across diverse market regimes.

The key contributions of this work are :

- **Cross-modal integration:** We seamlessly combine LLM-derived sentiment scores with structured financial data within a unified RL-driven portfolio optimization framework.
- **Hierarchical aggregation:** We introduce a novel three-layer architecture that hierarchically combines base agent decisions through meta-agents and a final super-agent, enabling adaptive decision-making across diverse market conditions.
- **Practical applicability:** Our approach showcases the effective deployment of lightweight LLMs in finance, offering a scalable and interpretable solution for latency-sensitive and transparency-critical applications.

The remainder of this paper is organized as follows. In Section 2, we establish the foundations of our work by reviewing the state of the art in Portfolio Optimization (PO), Reinforcement Learning (RL), and Natural Language Processing (NLP) within financial applications. Section 3 presents our overall framework architecture, detailing the NLP-driven and data-driven pipelines used to extract features and construct monthly observation vectors for RL agents. In Section 4, we describe the selected portfolio assets and outline the constraints imposed to reflect realistic investment scenarios. Section 5 introduces the individual RL agents and explains how actions, rewards, and training were implemented in our portfolio management environment. We then detail the hierarchical structure of our RL pipeline in Section 6, where base agents are aggregated via meta-agents trained to specialize on different data modalities. Building on this, Section 7 introduces the super-agent that synthesizes meta-agent outputs to produce final portfolio allocations. Section 8 gives key results. Finally, Section 9 concludes and gives hints for future research and enhancement.

2 Literature Review

2.1 Portfolio Optimization

Portfolio optimization has long been a cornerstone of financial management, with Harry Markowitz's Mean-Variance Optimization (MVO) framework serving as its foundation [Markowitz, 1952]. MVO revolutionized investment strategy by quantifying the trade-off between risk and return, proposing that investors should select portfolios that maximize expected return for a given level of risk, or minimize risk for a desired return.

This led to the concept of the efficient frontier, where optimal portfolios reside. However, MVO rests on assumptions such as Gaussian returns and static correlations, which rarely hold in real-world markets. Financial crises, notably Black Monday in 1987 and the 2008 global financial meltdown, exposed these limitations, as markets exhibited extreme volatility and non-linear behaviors that MVO failed to anticipate. These events underscored the need for more adaptive and dynamic approaches to portfolio management.

2.2 Reinforcement Learning in Finance

Reinforcement Learning (RL) has emerged as a powerful alternative for financial decision-making, particularly in dynamic and uncertain environments. Early pioneers like Moody and Saffell [Deng *et al.*, 2016] applied RL to trading, demonstrating its potential for sequential decision-making. More recently, [Jiang *et al.*, 2017] and [Liang *et al.*, 2018] introduced a deep RL framework tailored for portfolio management, leveraging the ability of RL agents to learn optimal policies through interaction with market environments. These algorithms enable RL agents to adapt dynamically to market conditions, learning from experience rather than relying on static assumptions, making them well-suited for portfolio optimization in very fast-evolving markets.

2.3 NLP in Financial Applications

Natural Language Processing (NLP) has revolutionized extracting insights from unstructured financial data. FinBERT, a BERT variant fine-tuned on financial texts, excels at classifying sentiment in news and social media into positive, neutral, or negative categories [Araci, 2019]. This sentiment analysis captures market trends and investor behavior beyond historical price data [Tetlock, 2007], enhancing predictive models for market movements.

Recent studies support compact domain-specific models like FinBERT in financial applications. [Lefort *et al.*, 2024] show that fine-tuned lightweight models, such as FinBERT and FinDRoBERTa, outperform large-scale models like GPT-3.5 and GPT-4 in financial sentiment classification, especially in zero-shot settings, making FinBERT a reliable, efficient choice for sentiment signal generation in RL frameworks.

A survey by [Li *et al.*, 2024] categorizes large language model applications in finance, including sentiment analysis and risk forecasting, highlighting trade-offs between domain-specific fine-tuning and general-purpose models.

Recent FinLLM challenge submissions demonstrate innovative LLM applications. Finance Wizard [Lee and Lay-Ki, 2024] fine-tuned a LLaMA3-based model for financial news summarization. L3iTC [Pontes *et al.*, 2024] used quantization and LoRA for efficient financial text classification. The CatMemo team [Cao *et al.*, 2024] improved cross-task generalization by integrating diverse financial datasets for LLM fine-tuning.

2.4 CompAI

Composite AI represents a paradigm shift, blending multiple AI techniques to create robust, context-aware systems. In the context of portfolio management, Composite AI integrates RL's decision-making capabilities with NLP's sentiment insights, forming a holistic approach that addresses both quantitative and qualitative market factors. The hierarchical structure proposed in this paper exemplifies Composite AI, leveraging specialized agents to process distinct data types and synthesizing their outputs for optimized portfolio allocations.

3 Methods

3.1 Architecture

Our portfolio optimization framework integrates reinforcement learning (RL) and natural language processing (NLP) with a three-tier hierarchical structure as described in Figure 1. Base agents, using Stable Baselines 3 algorithms, process monthly financial metrics from YahooFinance or sentiment scores from financial news via FinBERT, proposing portfolio weights in custom RL environments with a reward function balancing ROI, volatility, and drawdown. Meta-agents, built in PyTorch, refine outputs from data-driven and NLP-based base agents, while a super-agent combines these to produce final allocations. Trained on 2003–2017 data and backtested on 2018–2024, the system outperforms benchmarks, effectively blending quantitative and qualitative insights for modern investment strategies.

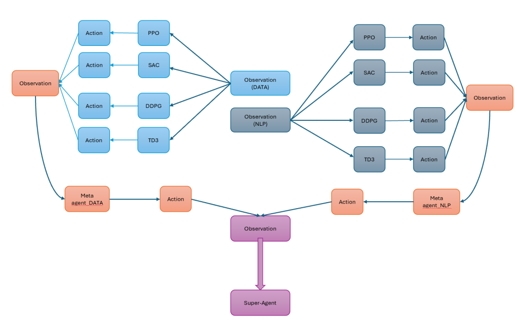

Figure 1: Summarized Architecture

3.2 Data-Driven Pipeline

Collecting Closing Prices

We gather daily adjusted closing prices for 14 financial asset from January 1, 2003, to December 31, 2024. This data is fetched using the `yfinance` Python library, which connects to Yahoo Finance. Adjusted closing prices are used because they adjust for events like stock splits and dividends, making them suitable for accurate financial analysis. The process involves specifying asset tickers (e.g., GSPC for S&P 500), setting the date range, and downloading the data into a structured format like a CSV file.

Creating Monthly Observation Vectors

Using the daily closing prices, we create monthly observation vectors for the reinforcement learning (RL) agent. For each month, we compute

- **Sharpe Ratio**: Measures risk-adjusted return based on daily returns.

$$\text{Sharpe Ratio} = \frac{E[R_p - R_f]}{\sigma_p} \tag{1}$$

  where R_p is the mean daily return of the portfolio, R_f is the risk-free rate, and σ_p is the standard deviation of daily returns (volatility).

- **Sortino Ratio**: Focuses on downside risk, using the standard deviation of negative returns.

$$\text{Sortino Ratio} = \frac{E[R_p - R_f]}{\sigma_d} \tag{2}$$

  where σ_d is the downside deviation:

$$\sigma_d = \sqrt{\frac{1}{T} \sum_{t: R_t < 0} (R_t - 0)^2} \tag{3}$$

  with T as the number of days with negative returns R_t.

- **Maximum Drawdown (MDD)**: Largest peak-to-trough decline in portfolio value within the month.

$$\text{MDD} = \max_{t \in [1,T]} \left(\frac{\text{Peak}_t - \text{Trough}_t}{\text{Peak}_t} \right) \tag{4}$$

  where Peak_t is the highest portfolio value up to time t, and Trough_t is the lowest value after the peak.

- **Calmar Ratio**: Measures return relative to maximum loss.

$$\text{Calmar Ratio} = \frac{E[R_p]}{\text{MDD}} \tag{5}$$

  where MDD is the maximum drawdown.

- **Volatility**: Standard deviation of daily returns.

$$\sigma_p = \sqrt{\frac{1}{T-1} \sum_{t=1}^{T} (R_t - \bar{R})^2} \tag{6}$$

  where R_t is the daily return at time t, and $\bar{R}$ is the mean daily return over T days.

- **Correlation Matrix**

  The correlation matrix is computed from daily returns across N assets. The Pearson correlation coefficient between assets i and j is:

$$\rho_{i,j} = \frac{\text{Cov}(R_i, R_j)}{\sigma_i \sigma_j} \tag{7}$$

  where the covariance is:

$$\text{Cov}(R_i, R_j) = \frac{1}{T-1} \sum_{t=1}^{T} (R_{i,t} - \bar{R}_i)(R_{j,t} - \bar{R}_j) \tag{8}$$

  with $R_{i,t}, R_{j,t}$ as daily returns of assets i and j, and $\bar{R}_i, \bar{R}_j$ as their mean returns.

  The correlation matrix C is an $N \times N$ symmetric matrix with $C_{i,j} = \rho_{i,j}$, $C_{i,i} = 1$, and $C_{i,j} = C_{j,i}$. It is flattened into a vector by taking the upper triangular elements (excluding the diagonal), yielding $\frac{N(N-1)}{2}$ unique correlations.

3.3 NLP-Driven Pipeline

How to Aboard the Time Specific Data Collection Issue?

To collect news articles matching each month from 2003 to 2024, we use Google News with date filters. For each of the 14 assets, we define search terms (e.g., "S&P 500", "SPX") and scrape articles published within each month. These articles are processed with FinBERT, a model that analyzes financial sentiment, to produce monthly sentiment scores. The pseudo code given in Algorithm 1 outlines this process:

Algorithm 1 News Scraping and Sentiment Analysis

Input: Assets and keywords
Output: Monthly sentiment scores
for each asset **do**
    Define terms (e.g., "S&P 500" = {"SP 500", "SPX"})
end for
for each term **do**
    for each month in 2003–2024 **do**
        Generate Google News URL with date filter
        Scrape the 10 first article for each links
    end for
end for
for each article **do**
    Extract text
    Compute sentiment with FinBERT
    Compute asset sentiment score $S_t = \frac{\sum (P_{\text{positive}} - P_{\text{negative}})}{N}$
end for
Store scores by month and asset

NLP Driven Observation Vectors

The NLP-driven observation vector for each month combines:

- **Volatility Vector**: Standard deviation of daily returns.
- **Sentiment Score Vector**: Derived from that month's news.

We chose to stress the importance of volatility as it gives the agent an extra leg to stand on. The volatility of the market is a strong indicator and it often indicates the precision of trends (trends will be simpler to identify in a low volatility market).

Data handling

Data quality is paramount in financial modeling, and rigorous preprocessing ensures that Reinforcement Learning

(RL) agents receive clean, standardized inputs. For price data, missing values—often due to non-trading days—are addressed using forward-filling, backward filling, or linear interpolations, as financial prices typically change gradually. This method preserves the continuity of market trends by minimizing disruptions in the time series. Prices are then normalized to all be 1 at first open, which is essential for comparing assets with vastly different price magnitudes. Without normalization, RL agents might inadvertently overweight higher-priced assets, skewing portfolio allocations.
For sentiment data, monthly aggregation of sentiment scores normalizes volume disparities across assets, as some indices receive far more media coverage than others. This ensures that sentiment inputs are consistent and comparable, preventing bias toward heavily covered assets.

3.4 Reproducibility with Google Collab

To ensure full transparency and enable further research, all experiments presented in this paper are reproducible via three Google Colab notebooks, each addressing a different part of the pipeline:

- **Data Pipeline and Sentiment Extraction:** The first notebook[1] provides a detailed, end-to-end pipeline for financial data collection and sentiment score generation. It scrapes financial news, applies FinBERT to extract sentiment at the asset level, and exports formatted sentiment scores for downstream use.
- **Fast Simulated RL Run (Sentiment Precomputed):** The second notebook[2] reproduces the reinforcement learning training pipeline using simulated sentiment data. This allows users to quickly test model dynamics, training cycles, and agent behavior with minimal compute (typically under 30 minutes).
- **Full Pipeline with Training:** The third notebook[3] combines the data scraping, sentiment extraction, and RL training into one integrated workflow. While comprehensive, this notebook is compute-intensive and requires approximately 8 hours of runtime in a typical Colab Pro environment.

This modular design offers both accessibility for quick experimentation and full reproducibility of the long-term training benchmarks presented in the paper.

4 Financial Instruments

Our portfolio consists both of equities and commodities, selected to ensure diversification across asset classes, regions, and economic drivers. Stock indices capture broad market dynamics and offer lower idiosyncratic risk, while commodities reflect real-world supply-demand conditions, providing uncorrelated signals.

[1]https://colab.research.google.com/drive/1DLQIooP7kNYHztQ7tHu5eO9NPNDPxIrY?usp=sharing
[2]https://colab.research.google.com/drive/1FPX9_8z0X39Pg3tf1bSvoByEWbbQ_juF?usp=sharing
[3]https://colab.research.google.com/drive/1SbKGmPLjF2DAKkNwEYdfc_2lS2KWMySi?usp=sharing

4.1 List of Assets

To ensure sufficient market coverage and data diversity, the portfolio includes both equities and commodities spanning multiple geographic regions and economic sectors. Stock indices serve as proxies for macroeconomic conditions across developed and emerging markets, while commodities provide exposure to real asset dynamics and serve as potential hedges during equity downturns. This combination supports the training of reinforcement learning agents on heterogeneous data sources and enhances the model's ability to generalize across financial regimes.

Table 1 summarizes the selected instruments along with their corresponding tickers and asset class labels. These assets were chosen based on liquidity, historical availability, and relevance in global financial markets.

Ticker	Asset	Asset Class
GSPC	S&P 500 Index	Equities
IXIC	NASDAQ Composite	Equities
DJI	Dow Jones Industrial Average	Equities
FCHI	CAC 40 (France)	Equities
FTSE	FTSE 100 (UK)	Equities
STOXX50E	EuroStoxx 50	Equities
HSI	Hang Seng Index (Hong Kong)	Equities
000001.SS	Shanghai Composite (China)	Equities
BSESN	BSE Sensex (India)	Equities
NSEI	Nifty 50 (India)	Equities
KS11	KOSPI (South Korea)	Equities
GC=F	Gold	Commodities
SI=F	Silver	Commodities
CL=F	WTI Crude Oil Futures	Commodities

Table 1: Complete list of financial instruments used in the portfolio, grouped by asset class.

4.2 Portfolio Constraints and Rules

Our experiment imposes strict rules to mimic realistic investment scenarios.

- **Long-Only**: We only buy assets, not sell them short. Short-selling—borrowing an asset to sell, then repurchasing it later—adds complexity and risk (e.g. unlimited losses if prices soar). A long-only approach keeps things simpler and safer, aligning with conservative strategies.
- **No Leverage**: We invest only the capital we have, without borrowing. Leverage amplifies gains but also losses—borrowing \$50,000 to add to a \$100,000 portfolio could double profits or wipe out the initial stake. Avoiding leverage caps downside risk.
- **Monthly Rebalancing**: Every month, the RL agent reassigns weights to the 14 assets based on its policy. For example, if gold surges, it might increase gold's share from 7% to 10%. This cadence balances adaptability with practicality, as frequent trading incurs costs (excluded here for simplicity).
- **Equal Initial Weights**: At the outset, each asset gets roughly 7.14% of the portfolio. This neutral start lets the RL agent shape the portfolio without inherited biases.

These constraints ground the experiment in real-world norms, ensuring that AI decisions are practical and interpretable. To change those, it is possible to use the codes provided in Section 3.4 and changing or taking out parameters (for leverage, take out the normalization step)

4.3 Benchmarks for Performance Evaluation

We measure our RL approach against two standards:

- **Equal-Weighted Portfolio**: Each of the 14 assets gets 7.14%, this gives an idea of the performance gains of the strategy compared to a simple buy and hold.
- **S&P 500 (GSPC)**: The most commonly used financial benchmark in Portfolio Management. tracking U.S. market performance.

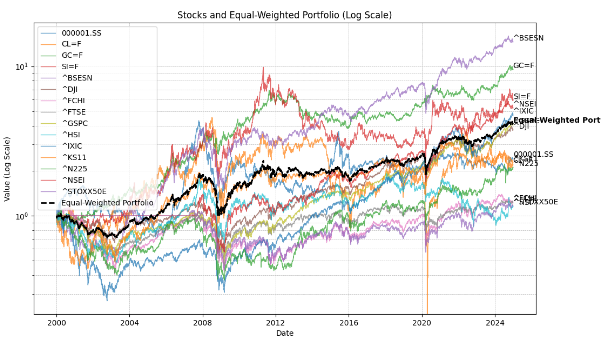

Figure 2: Log evolution of Normalized Asset Prices vs Normalized Equal weights (2003-2025)

We choose to model log evolution to get a grasp of a strongly varying financial context as provided in Figure 2. It would be hard to get a good idea of what is happening if using linear scales as markets change very strongly and very fast.

5 Stable Baselines 3 Agents and Environment Setup

We chose to use Stable Baselines 3 (SB3) [Raffin *et al.*, 2021], a widely adopted Python library that implements state-of-the-art reinforcement learning (RL) algorithms on top of OpenAI Gym environments. Its modular design, ease of integration, and support for policies make it well-suited for financial applications where agents must learn sequential allocation decisions.

5.1 Action

The action space is continuous, representing the portfolio weights for each asset. These weights must sum to 1 and be non-negative (no leverage, no short-selling), aligning with standard portfolio constraints. A continuous action space allows for precise adjustments, unlike discrete actions which would limit flexibility in allocation.

5.2 Reward Function

The reward function guides the agent's learning by balancing multiple objectives:

- **Return on Investment (ROI)**: Encourages higher portfolio returns.
- **Penalties**: For high volatility and large drawdowns, discouraging excessive risk.

We chose to attribute relative importance to each by a linear combination:*

$$Reward = \alpha_1 * ROI - \alpha_2 * MDD - \alpha_3 * \sigma$$

with the α_i's some real values defined depending on investor needs. For the results presented below, we used values varying between 0.5 and 2 (giving a relative but still consistent importance to each component, and severely punishing MDD).

5.3 Overview of Agents

We employ four well-established reinforcement learning algorithms tailored for continuous control in financial environments: Proximal Policy Optimization (PPO) [Schulman *et al.*, 2017], Soft Actor-Critic (SAC) [Haarnoja *et al.*, 2018], Deep Deterministic Policy Gradient (DDPG) [Lillicrap *et al.*, 2015], and Twin Delayed DDPG (TD3) [Fujimoto *et al.*, 2018]. PPO offers stable on-policy learning via clipped updates, while SAC encourages exploration through entropy maximization in off-policy settings. DDPG leverages deterministic policies for fine-grained action selection, and TD3 improves upon DDPG by mitigating overestimation bias with dual critics and delayed updates. (See appendix for detail)

5.4 Backtesting

Backtesting evaluates the RL agents on historical data to assess their performance. We test the agents on both the training period (2003–2017) and unseen data (2018–2024) to measure their ability to generalize beyond the training set.

5.5 Seeds

To ensure reproducibility, we use fixed consecutive seeds for all experiments. Seeds control the randomness in the environment and algorithms, allowing consistent results across runs.

6 Hierarchy Structure

6.1 Why Use Hierarchy Structures in AI?

Hierarchical Reinforcement Learning (HRL) enhances portfolio optimization by decomposing decision-making into manageable components, improving interpretability, scalability, and stability [Sutton *et al.*, 1999]. Base agents specialize in quantitative financial metrics or qualitative NLP-derived sentiment scores [Li *et al.*, 2021], enabling clear, traceable decisions. Meta-agents aggregate these outputs into cohesive strategies [Kulkarni *et al.*, 2016], ensuring transparency and ease of adjustment. This structure scales efficiently for larger portfolios or additional data types without excessive computational complexity, while meta-agents stabilize decisions by smoothing erratic actions, reducing portfolio volatility in dynamic financial markets [Jiang *et al.*, 2017].

6.2 Environment Setup

Two distinct hierarchies are established within the HRL framework. The first hierarchy consists of Natural Language Processing (NLP)-based agents, while the second is data-driven agents. By separating these tasks, the framework ensures that each base agent specializes in a specific data modality, producing traceable and interpretable recommendations.

A naive approach to combining base agent outputs might involve computing a weighted average of their recommendations for asset allocations in a given month. However, such statistical methods fail to account for the strengths and weaknesses of individual agents, limiting their ability to adapt to complex market conditions. Weighted averages or similar numerical methods lack the capacity to learn dynamically from agent performance, reducing their effectiveness in volatile financial environments. This limitation underscores the need for a more sophisticated aggregation strategy that can learn optimal policies over time [Jiang *et al.*, 2017].

To address this, we design a custom reinforcement learning environment implemented in PyTorch, where a meta-agent receives an observation vector formed by concatenating the proposed action vectors from each base agent across different seeds and layouts (NLP-based or data-driven). Each action vector represents a recommended weight allocation for assets in the portfolio. The meta-agent processes this observation vector and outputs a final action vector, which is a weight allocation ensuring that the portfolio. This hierarchical structure allows the meta-agent to learn how to weigh the contributions of base agents dynamically, improving decision-making in dynamic financial markets [Kulkarni *et al.*, 2016].

Both base and meta-agents are trained on historical financial data spanning 2003 to 2017, a period that includes diverse market conditions such as the 2008 financial crisis [Brunnermeier, 2009]. The training process enables agents to learn optimal policies through interaction with the environment. The performance of the HRL framework is evaluated on a separate testing period, ensuring robustness and generalizability. [Jiang *et al.*, 2017].

The meta-agent is modeled as a three-layer fully connected neural network with compounded ReLU activations and a final softmax output layer:

$$f_\theta(X_t) = \text{Softmax}\left(A_3\left(\phi\left(A_2\left(\phi\left(A_1\left(X_t\right)\right)\right)\right)\right)\right) \quad (9)$$

where:

- X_t is the observation vector at time t, aggregating actions from base agents,
- $A_i(x) = W_i x + b_i$ for $i = 1, 2, 3$ are affine transformations (weights and biases),
- $\phi(x) = \text{ReLU}(x) = \max(0, x)$ is the activation function,
- The final softmax layer ensures the output forms a valid allocation (i.e., non-negative weights summing to 1).

This compact architecture, inspired by deep reinforcement learning [Mnih and others, 2015], enables the meta-agent to learn flexible mappings from base agent outputs to portfolio allocations, while maintaining both structure and interpretability.

7 Final Super Agent

In this section, we introduce the final super agent, which serves as the top-level decision-maker in our HRL structure. The super agent aggregates insights from lower-level meta-agents to determine the optimal portfolio allocation as explained in Algorithm 2

Algorithm 2 Training Super-Agent using PyTorch

```
Require: Trained base RL agents {A_metadata, A_metaNLP}, training dataset D_train, learning rate α, epochs E
Ensure: Trained Meta-agent
Initialize PyTorch neural network f_θ with random weights
Define loss function L(θ) = (1/B) Σ_{i=1}^{B} ||f_θ(X_i) − w_i*||²
Define optimizer Adam(θ, α)
Collect training data:
for each time step t in D_train do
    Compute base agent decisions w_t^(i) = A_i(X_t) for all agents
    Simulate future portfolio value for each w_t^(i) over H steps (lookahead reward)
    Select the best action w_t* = arg max_{w_t^(i)} Σ_{j=t}^{t+H} R_j
    Store training sample (X_t, w_t*)
end for
for each epoch e in {1, ..., E} do
    Shuffle training data
    for each batch B in training set do
        Compute predictions ŵ_B = f_θ(X_B)
        Compute loss L(θ)
        Update model: θ ← θ − α∇_θ L(θ)
    end for
end for
Return trained model f_θ
```

Aggregation and Observation Vectors

The observation vector for the super agent consists of the portfolio weights proposed by the meta-agents. Specifically, it includes:

- The weights suggested by the data-driven meta-agent, which focuses on quantitative metrics.
- The weights suggested by the NLP-based meta-agent, which incorporates sentiment analysis.

This observation vector allows the super agent to ”see” the recommendations from both perspectives, enabling it to make a well-rounded decision by balancing numerical data and market sentiment.

We use the same structure as the meta-agents for this agent. The only changing variable is the input, which is now the concatenated action vectors of the two meta agents. This structure stongly mimics a common way in financial markets, comparing market sentiment to current state and finding disrespencies is what gives financial actors an edge. We can see the data based meta agent as a market analyser and the NLP based one as a conviction giver. This gives a direction from which traders can benefit.

8 Summary of results

Table 2 gives the reader an overview of the final results. As presented below, all meta agents beat benchmarks over the testing period and the super-agent seems to be implement a very strong strategy.

Agent/Benchmark	ROI (%)	Sharpe	Volatility (%)
Equal-Weights	7.5	0.57	13.3
S&P 500	13.2	0.63	19.7
Meta-Agent (Metrics)	14.7	0.8	16.0
Meta-Agent (NLP)	20.5	1.2	16.0
Super-Agent	26.0	1.2	20.0

Table 2: Performance of super-agent vs. benchmarks and meta-agents (2018–2024).

Table 3 provides a granular analysis of all agents, note that the results for base agents are the median out of the 5 seeds tested.

Agent	Annualized ROI (%)	Annualized Sharpe	Annualized Volatility (%)
Equal-Weights Portfolio	7.5	0.57	13.3
S&P 500	13.2	0.63	19.7
$\text{PPO}_{\text{metrics}}$	12.9	0.6	18.0
$\text{SAC}_{\text{metrics}}$	9.4	0.6	10.4
$\text{TD3}_{\text{metrics}}$	16.5	0.8	21.3
$\text{DDPG}_{\text{metrics}}$	10.9	0.5	18.4
$\text{Meta-Agent}_{\text{metrics}}$	14.7	0.8	16.0
PPO_{NLP}	14.8	1.0	13.4
SAC_{NLP}	9.1	0.9	10.0
TD3_{NLP}	17.5	0.8	19.2
DDPG_{NLP}	12.9	0.7	18.0
$\text{Meta-Agent}_{\text{NLP}}$	20.5	1.2	16.0
Super-Agent	26.0	1.2	20.0

Table 3: Analysis of Results for Agents and Benchmarks.

Comparison with State-of-the-Art RL Strategies

To contextualize our framework's performance, Table 4 compares our meta-agents and super-agent with recent RL-based portfolio optimization strategies from academic literature. We compare our results to the 2024 study [Espiga-Fernández *et al.*, 2024], and against the deep RL framework [Jiang *et al.*, 2017]. Closely competing with CNN-RL (22.0% ROI, 1.3 Sharpe), our super agent seems to have surpassed the current state of the art. Furthermore, the consistent superiority of NLP augmented agents goes to confirm the results of [Xu and Zhou, 2018].

The super-agent's ROI of 26.0% demonstrates the effectiveness of the hierarchical approach, integrating quantitative metrics and sentiment analysis via NLP to outperform benchmarks and individual meta-agents. The strong performance of NLP-based agents, particularly TD3_{NLP} and $\text{Meta-Agent}_{\text{NLP}}$, underscores the value of sentiment-driven decision-making.

9 Conclusion and Future Directions

This paper introduces an innovative hierarchical reinforcement learning (RL) framework for portfolio optimization, integrating structured financial indicators with sentiment signals extracted from financial news using lightweight, domain-specific large language models (LLMs) such as FinBERT. The framework leverages a three-tier multi-agent architecture—comprising base agents that process hybrid data, meta-agents that aggregate these decisions, and a super-agent that synthesizes final portfolio allocations—enabling adaptive, interpretable, and robust decision-making in dynamic market environments.

Strategy	Annualized ROI (%)	Sharpe Ratio	Volatility (%)
Meta-Agent (Metrics)	14.7	0.8	16.0
Meta-Agent (NLP)	20.5	1.2	16.0
Super-Agent	26.0	1.2	20.0
DQN [Espiga-Fernández *et al.*, 2024]	26	0.8	38
DDPG [Espiga-Fernández *et al.*, 2024]	20.0	0.7	37
PPO [Espiga-Fernández *et al.*, 2024]	19	0.8	25
CNN-RL [Jiang *et al.*, 2017]	22.0	1.3	19.5
RNN-RL [Jiang *et al.*, 2017]	19.5	1.1	18.5
LSTM-RL [Jiang *et al.*, 2017]	21.0	1.2	19.0

Table 4: Comparison of meta-agents and super-agent with state-of-the-art RL-based portfolio optimization strategies.

However, the current implementation has limitations. It assumes synchronously available data inputs, which may not align with real-world asynchronous market conditions. Transaction costs are excluded, potentially overestimating practical returns, and the system has not been tested under adversarial or extreme market scenarios. Additionally, sentiment signals derived from financial news, while beneficial, may introduce noise or biases reflective of media perspectives, which could affect decision accuracy.

To overcome these shortcomings, future research will pursue several enhancements:

- Asynchronous Data Integration: Incorporating real-time and asynchronous data streams to better reflect market dynamics.
- Transaction Cost and Stress Testing: Adding transaction cost modeling and evaluating performance under adversarial conditions to improve real-world applicability.
- Expanded Text Corpus: Broadening the sentiment analysis by including diverse sources such as earnings calls, regulatory filings, and social media.
- Larger LLMs Exploration: Comparing the efficacy of lightweight, domain-specific LLMs against larger, general-purpose models (e.g., GPT, Claude, LLaMA) to assess scalability and performance trade-offs.
- Possibility of strategy developments using other financial tools (End of month expiring options, Futures, Perpetuals, etc)

These advances aim to refine the robustness and generalizability of the framework, making it more suitable for practical deployment.

References

[Araci, 2019] Dilan Araci. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*, 2019.

[Brunnermeier, 2009] Markus K. Brunnermeier. Deciphering the liquidity and credit crunch 2007–2008. *Journal of Economic Perspectives*, 23(1):77–100, 2009.

[Cao *et al.*, 2024] Yupeng Cao, Zhiyuan Yao, Zhi Chen, and Zhiyang Deng. Catmemo at the finllm challenge task: Fine-tuning large language models using data fusion in financial applications. *arXiv preprint arXiv:2407.01953*, 2024.

[Deng *et al.*, 2016] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):653–664, 2016.

[Espiga-Fernández *et al.*, 2024] Francisco Espiga-Fernández, Álvaro García-Sánchez, and Joaquín Ordieres-Meré. A systematic approach to portfolio optimization: A comparative study of reinforcement learning agents, market signals, and investment horizons. *Algorithms*, 17(12):570, 2024.

[Fujimoto *et al.*, 2018] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[Jiang *et al.*, 2017] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.

[Kulkarni *et al.*, 2016] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in Neural Information Processing Systems*, 29:3325–3333, 2016.

[Lee and Lay-Ki, 2024] Meisin Lee and Soon Lay-Ki. 'finance wizard'at the finllm challenge task: Financial text summarization. *arXiv preprint arXiv:2408.03762*, 2024.

[Lefort *et al.*, 2024] Baptiste Lefort, Eric Benhamou, Jean-Jacques Ohana, David Saltiel, and Beatrice Guez. Optimizing performance: How compact models match or exceed gpt's classification capabilities through fine-tuning. *arXiv preprint arXiv:2405.12345*, 2024.

[Li *et al.*, 2021] Feng Li, Jian Jiang, and Ming Xu. Sentiment analysis and its impact on financial markets: A comprehensive review. *Review of Quantitative Finance and Accounting*, 56(2):345–372, 2021.

[Li *et al.*, 2024] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large language models in finance: A survey. *arXiv preprint arXiv:2311.10723*, 2024.

[Liang *et al.*, 2018] Zhipeng Liang, Yun Chen, Yaoxing Zhu, Jun Jiang, and Zhen Li. Adversarial deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940*, 2018.

[Lillicrap *et al.*, 2015] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[Markowitz, 1952] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[Mnih and others, 2015] Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Pontes *et al.*, 2024] Elvys Linhares Pontes, Carlos-Emiliano González-Gallardo, Mohamed Benjannet, Caryn Qu, and Antoine Doucet. L3itc at the finllm challenge task: Quantization for financial text classification and summarization. *arXiv preprint arXiv:2408.03033*, 2024.

[Raffin *et al.*, 2021] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Sutton *et al.*, 1999] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[Tetlock, 2007] Paul C. Tetlock. Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, 62(3):1139–1168, 2007.

[Unnikrishnan and others, 2024] Ananya Unnikrishnan et al. Financial news-driven llm reinforcement learning for portfolio management. *arXiv preprint arXiv:2411.11059*, 2024.

[Xu and Zhou, 2018] Yue Xu and Samuel Zhou. News-driven reinforcement learning for algorithmic trading. *arXiv preprint arXiv:1807.05589*, 2018.