Portfolio Optimization in NEPSE Using Deep Learning and Combinatorial Constraints

Abhinaash Tiwari [030152-21]
at22031721@student.ku.edu.np
9865471375

September 15, 2025

1. Introduction

Portfolio optimization is a fundamental problem in financial mathematics, seeking to maximize returns while minimizing risk under a set of constraints. Traditional approaches, such as mean-variance optimization, often face limitations when applied to real-world markets due to difficulties in accurately predicting returns and handling discrete investment restrictions.

To address these challenges, this project integrates deep learning (DL) to predict asset returns and volatility with combinatorial optimization techniques that account for practical constraints, such as limiting the number of selected assets or ensuring sectoral diversification.

This approach is particularly relevant in the context of the Nepal Stock Exchange (NEPSE), where issues such as sector concentration, liquidity restrictions, and limited diversification opportunities make portfolio construction especially challenging. By combining predictive modeling with discrete optimization, the proposed framework provides a more realistic and robust approach to portfolio selection, offering valuable insights to investors in emerging markets like Nepal.

2. Objectives

- Understand and implement the techniques of Deep Learning, Combinatorial Constraints, Portfolio Optimization, and Decision Theory.
- Develop a deep neural network to predict asset returns and volatility, providing a foundation for financial market forecasting.
- Implement an optimization algorithm to construct a diversified portfolio that incorporates combinatorial constraints.
- Evaluate the final portfolio performance using metrics such as the Sharpe ratio and maximum drawdown.
- Analyze the trade-off between portfolio risk, return, and constraints to understand how strategic choices impact performance.

3. Problem Statement

In this work, we hope to develop a portfolio optimization framework tailored to NEPSE that integrates deep learning models with combinatorial constraints. Unlike traditional approaches, such as mean-variance optimization, which rely heavily on forecasting expected returns through econometric models, our method leverages deep learning to directly learn allocation strategies that maximize portfolio performance.

Previous studies have shown that return forecasting alone does not necessarily translate to optimal portfolio outcomes, as minimizing prediction error is not equivalent to maximizing investment reward. To address this, our framework incorporates predictive models (e.g., LSTM, Random Forest) for returns and volatility, and then formulates a combinatorial optimization problem that directly maximizes risk-adjusted performance metrics such as the Sharpe ratio.

By embedding discrete constraints such as limiting the number of selected assets or enforcing sector diversification, our model captures practical investment considerations in the Nepalese market while extracting salient patterns from multi-asset features to generate optimal portfolio weights.

4. Hypotheses

Main Hypothesis (H1):

- Deep learning models predict stock returns and volatility in NEPSE more accurately than traditional models, leading to better portfolio optimization outcomes.
- **Null Hypothesis ($H0_1$):** Deep learning models do not significantly improve prediction accuracy or portfolio outcomes compared to traditional models.

Supporting Hypotheses:

- **H2:** Portfolios optimized with deep learning–based predictions achieve higher risk-adjusted returns (e.g., Sharpe ratio) compared to traditional mean-variance portfolios.
- **$H0_2$:** There is no significant difference in risk-adjusted returns between deep learning–based portfolios and traditional mean-variance portfolios.
- **H3:** Introducing combinatorial constraints (such as limiting the number of assets or enforcing sector diversification) improves portfolio stability and reduces concentration risk without significantly lowering returns.
- **$H0_3$:** Combinatorial constraints do not improve portfolio stability and may reduce returns.
- **H4:** The combination of deep learning predictions and combinatorial optimization results in portfolios that outperform equal-weighted and index-based benchmarks.
- **$H0_4$:** Portfolios using deep learning with combinatorial optimization do not significantly outperform equal-weighted or index-based benchmarks.

5. Methodology

1. **Data Collection**

   - Select an asset universe of approximately 15 NEPSE-listed stocks from different sectors, ensuring sufficient liquidity.
   - Collect historical daily (or weekly) OHLCV data over a long period (e.g., 5–10 years), plus sector indices and relevant macroeconomic variables for augmentation.

2. **Data Preprocessing**

   - Handle missing values, adjust for splits and dividends.
   - Compute returns (simple or log), normalize or standardize features.
   - Check and, if necessary, transform for stationarity.
   - Split data by time: training (first 70%), validation (15%), test (last 15%).

3. **Feature Engineering**

   - Compute historical volatilities (rolling windows, e.g., 20, 60, 120 days).
   - Generate technical indicators (e.g., SMA, EMA, MACD, RSI).
   - Compute liquidity features (volume, turnover).
   - Include macroeconomic indicators if available.
   - Perform feature selection or dimensionality reduction (e.g., correlation thresholding or PCA) to avoid overfitting.

4. **Predictive Modeling**

   - Train several models: LSTM, Random Forest, XGBoost, and simple benchmarks.
   - Define input-output structure: sliding windows of past n periods to predict future returns/volatility over k periods.
   - Use validation set for hyperparameter tuning.
   - Apply techniques to avoid overfitting (early stopping, dropout, regularization).
   - Evaluate predictive performance using RMSE, MAE, and R^2.

5. **Portfolio Optimization with Combinatorial Constraints**

   - Formulate the optimization problem: maximize risk-adjusted return (e.g., Sharpe ratio) with continuous weights and binary selection variables.
   - Constraints include:
     - Maximum number of assets selected ($k = 5, 10, 15$).
     - Minimum/maximum weight per asset.
     - Sector diversification (e.g., no sector exceeds some percentage).
     - Fully invested (sum of weights = 1), non-negative weights if no short selling.

- Use a solver supporting Mixed-Integer Quadratic Programming (MIQP) via CVXPY or Pyomo with appropriate backends (GLPK, CBC, Gurobi, etc.).

6. **Evaluation and Backtesting**
   - Test predictive models and portfolio performance on the out-of-sample dataset.
   - Compute portfolio metrics: annualized return, volatility, Sharpe ratio, maximum drawdown, Sortino ratio.
   - Compare to benchmarks: equal-weighted portfolio, traditional mean-variance optimization, NEPSE index.
7. **Sensitivity and Robustness Analysis**
   - Vary k to analyze diversification-return trade-offs.
   - Explore different model architectures and hyperparameters.
   - Perform rolling window backtests across different periods.
   - Conduct statistical tests (e.g., paired t-test, Wilcoxon signed-rank) for significance.

6. Expected Outcomes

- A Python program that predicts asset returns using DL and optimizes portfolios with combinatorial constraints.
- Results demonstrating the trade-off between risk, return, and constraint complexity.
- A comparison of DL-based portfolio optimization with traditional methods, highlighting improvements in predictive accuracy and performance.