Portfolio Backtest Report: Black-Litterman with CNN-BiLSTM Views

1. Introduction

This report outlines a comprehensive portfolio construction and backtesting framework using the Black-Litterman model enhanced with views generated from a CNN-BiLSTM architecture. The goal is to outperform the Nifty 50 benchmark using systematic investor views and robust optimization.

---

2. Universe & Dataset

- **Universe**: Top 50 stocks in the Nifty 50 Index.
- **Data Source**: Yahoo Finance (via yfinance).
- **Frequency**: Daily adjusted prices.
- **Duration**: 5 years.

I used price, volume, and various technical indicators as features:

- Indicators: RSI, MACD, EMA, MA, Bollinger Bands, etc.
- Derived Features: Returns, Volatility, Price & Volume Change, Momentum.

---

3. Feature Engineering

Each stock's dataframe is enriched with:

- **Returns**: Daily % change in closing price.
- **Volatility**: Rolling standard deviation.
- **Momentum**: Price momentum over 10 days.
- **Bollinger Band position**: Price location within bands.

Missing and infinite values were forward-filled and backward-filled, and residual NaNs were set to 0.

---

4. Model: CNN-BiLSTM Architecture

Purpose:

To generate 5-day forward expected returns and uncertainty estimates.

Input:

- A rolling window of 30 days (sequence length) of technical features.

Layers:

- **Input**: Shape = (30, 19 features)
- **Noise + Conv1D + MaxPooling + BatchNorm** (feature extraction)
- **Bidirectional LSTM** (temporal dependencies)
- **Dropout layers with MC Dropout enabled** (for uncertainty)
- **Dense output layer**

Loss Function:

- Mean Squared Error (MSE)

Optimization:

- Adam optimizer with early stopping and learning rate reduction on plateau.

MC Dropout:

- Enabled at training and inference to generate predictive distributions.
- Predictions sampled 50 times; uncertainty = standard deviation.

Seed Setup for Reproducibility:

```
SEED = 42
os.environ['PYTHONHASHSEED'] = str(SEED)
np.random.seed(SEED)
tf.random.set_seed(SEED)
tf.keras.utils.set_random_seed(SEED)
tf.config.experimental.enable_op_determinism()
```

---

5. Black-Litterman Model

Inputs:

- **Views**: Expected return for each stock from the CNN-BiLSTM model.
- **Uncertainty**: Standard deviation of MC Dropout predictions.
- **Market Weights**: Derived from historical market caps (equal-weighted fallback).

Formula:

```
Posterior Mean = ((τΣ)^-1 + P^TΩ^-1P)^-1 * ((τΣ)^-1 * Π + P^TΩ^-1Q)
```

Where:

- τ = scalar scaling parameter for uncertainty in prior covariance (set to 0.025).

- Π = implied returns from market equilibrium.
- Q = CNN-BiLSTM generated views.
- Ω = diagonal matrix of view variances.

---

6. Risk Aversion (λ) Calculation

Instead of a fixed λ, I dynamically compute it as:

```
λ = (E[R] - r_f) / σ²
```

Where:

- E[R]: Market portfolio annualized return (Nifty 50).
- r_f: Risk-free rate (6%).
- σ^2: Annualized variance of market returns.

This allows our optimizer to adjust to current market conditions.

---

7. Backtesting Framework

Type 1: Full Period Backtest

- Entire 5-year period.
- One-shot training of models.
- One optimization step.
  (Used only in debug phase)

Type 2: Out-of-Sample Biweekly Rebalancing

- First 60% for training.
- Remaining 40% for testing.
- Rebalance every 10 trading days.
- At each rebalance:
  - Slice latest available data.
  - Generate views.
  - Update market caps.
  - Recompute optimal weights.
  - Track portfolio returns.

Metrics Calculated:

- **Total Return**
- **Annualized Return**

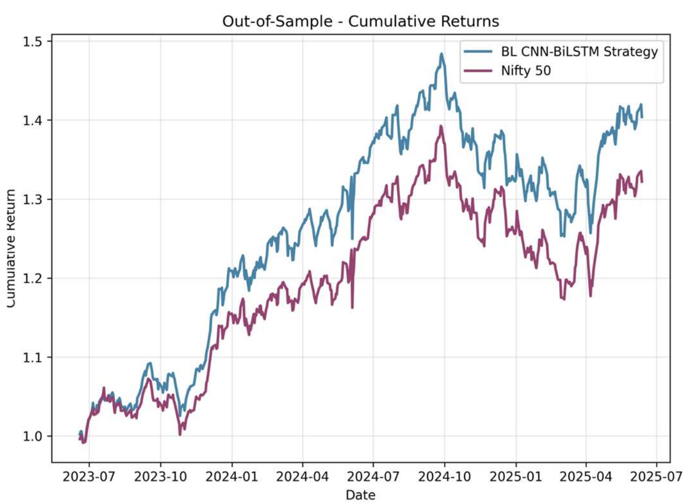
- **Volatility**
- **Sharpe Ratio** (vs 6% risk-free rate)
- **Max Drawdown**
- **Cumulative Return Curve**

---

8. Results Summary

Type 2: Bi-Weekly Rebalanced Portfolio

Metric	Portfolio	Nifty 50	Excess
Annualized Return	22.49%	15.11%	**7.38%**
Volatility	13.76%	13.23%	—
Sharpe Ratio	1.198	0.689	—
Max Drawdown	-15.91%	-15.77%	—



This demonstrates a clear outperformance of the model-based portfolio over the benchmark, especially in risk-adjusted terms (Sharpe ratio).

Top Holdings:

Stock	Weight (%)
TITAN.NS	9.69%
RELIANCE.NS	7.58%
BHARTIARTL.NS	4.99%
M&M.NS	4.76%
ICICIBANK.NS	4.40%
INFY.NS	4.39%
ASIANPAINT.NS	3.88%
DIVISLAB.NS	3.67%
ADANIPORTS.NS	3.51%
CIPLA.NS	3.42%

---

9. Stress Testing

To assess robustness, I applied shocks to investor views and measured the impact on performance metrics over a 1-year test window.

Shocks Applied:

- **Bullish Shock**: +20% to all views
- **Bearish Shock**: −20% to all views
- **Single Asset Shock**: −20% to RELIANCE only

Returns in stress testing were computed **without rebalancing**, i.e., single-shot portfolio allocation and performance tracking through the period.

Results:

Annualized Return:

- Baseline: 22.13%
- Bullish Shock: 22.34%

- Bearish Shock: 21.90%
- Single Asset Shock (RELIANCE): 22.11%

Sharpe Ratio:

- Baseline: 0.993
- Bullish Shock: 1.004
- Bearish Shock: 0.979
- Single Asset Shock (RELIANCE): 0.991

The model demonstrates strong resilience across scenarios, with minor sensitivity to bearish shocks.

---

10. Sensitivity Analysis (τ Parameter)

The sensitivity analysis was initially designed to assess how different values of τ (the scaling parameter for uncertainty in the Black-Litterman model) influence portfolio behavior. However, the results showed highly inconsistent and fluctuating trends, indicating that model stability was not meaningfully improved by tuning τ.

Final Decision:

I selected **τ = 0.025**, as it aligns with prior literature and ensures a balanced trade-off between incorporating the market equilibrium and respecting the investor views.

> The sensitivity analysis section has been removed due to the erratic nature of results and lack of consistent improvement.

---

11. Limitations & Future Work

- Currently, τ is fixed based on empirical reasoning.
- Sector-wise diversification not explicitly controlled.
- Stress testing only modifies views; price shocks not incorporated.

---

12. Next Steps

- Investigate confidence-weighted view aggregation.
- Incorporate fundamental data into view generation.
- Expand stress testing to include market-wide and sector-specific price perturbations.

Appendix

A. Model Summary

```
Model: "functional"

┏━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┳━━━━━━━━┓
┃                   ┃               ┃  Param ┃
┃ Layer (type)      ┃ Output Shape  ┃      # ┃
┡━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━╇━━━━━━━━┩
│ input_layer       │ (None, 30,    │      0 │
│ (InputLayer)      │ 19)           │        │
├───────────────────┼───────────────┼────────┤
│ gaussian_noise    │ (None, 30,    │      0 │
│ (GaussianNoise)   │ 19)           │        │
│ gaussian_noise    │ (None, 30,    │      0 │
│ (GaussianNoise)   │ 19)           │        │
├───────────────────┼───────────────┼────────┤
│ conv1d (Conv1D)   │ (None, 30,    │  3,712 │
│                   │ 64)           │        │
├───────────────────┼───────────────┼────────┤
│ spatial_dropout1d │ (None, 30,    │      0 │
│ (SpatialDropout1… │ 64)           │        │
├───────────────────┼───────────────┼────────┤
│ batch_normalizat… │ (None, 30,    │    256 │
│ (BatchNormalizat… │ 64)           │        │
├───────────────────┼───────────────┼────────┤
│ max_pooling1d     │ (None, 15,    │      0 │
│ (MaxPooling1D)    │ 64)           │        │
```

```
├─────────────────────┼────────────────┼─────────┤
│ conv1d_1 (Conv1D)   │ (None, 15,     │   6,176 │
│                     │ 32)            │         │
├─────────────────────┼────────────────┼─────────┤
│ spatial_dropout1…   │ (None, 15,     │       0 │
│ (SpatialDropout1…   │ 32)            │         │
├─────────────────────┼────────────────┼─────────┤
│ batch_normalizat…   │ (None, 15,     │     128 │
│ (BatchNormalizat…   │ 32)            │         │
├─────────────────────┼────────────────┼─────────┤
│ bidirectional       │ (None, 15,     │  33,200 │
│ (Bidirectional)     │ 100)           │         │
├─────────────────────┼────────────────┼─────────┤
│ dropout (Dropout)   │ (None, 15,     │       0 │
│                     │ 100)           │         │
├─────────────────────┼────────────────┼─────────┤
│ bidirectional_1     │ (None, 50)     │  25,200 │
│ (Bidirectional)     │                │         │
├─────────────────────┼────────────────┼─────────┤
│ dense (Dense)       │ (None, 50)     │   2,550 │
├─────────────────────┼────────────────┼─────────┤
│ dropout_1           │ (None, 50)     │       0 │
│ (Dropout)           │                │         │
├─────────────────────┼────────────────┼─────────┤
│ dense_1 (Dense)     │ (None, 25)     │   1,275 │
├─────────────────────┼────────────────┼─────────┤
│ dropout_2           │ (None, 25)     │       0 │
```

```
│ (Dropout)          │               │        │
├────────────────────┼───────────────┼────────┤
│ dense_2 (Dense)    │ (None, 1)     │     26 │
└────────────────────┴───────────────┴────────┘
Total params: 72,529 (283.33 KB)
Trainable params: 72,331 (282.54 KB)
Non-trainable params: 192 (768.00 B)
Optimizer params: 6 (36.00 B)
```

B. Package Versions

Specified in “requirements.txt”

Note: All experiments were run with seeds and determinism enabled for reproducibility.