Technical Report: Hybrid Deep Learning Model for Financial Time Series Forecasting

1. Introduction

This project implements a hybrid deep learning model combining Long Short-Term Memory (LSTM) networks and Transformer blocks to forecast financial time series data. The objective is to predict future returns of a basket of financial assets based on their historical weekly performance. The system aims to support portfolio optimization, signal generation, and quantitative trading strategies.

2. Data Pipeline

Historical financial data is fetched using the Yahoo Finance API via the `yfinance` package. The selected assets include ETFs like SPY, QQQ, IWM, EFA, EEM, TLT, etc. Data is cleaned, normalized using `StandardScaler`, and transformed into a supervised format using a sliding window technique (e.g., 30 weeks of input to predict the next week).

This pipeline supports incremental updates, saving previously fetched data and appending new observations weekly.

3. Modeling Approach

The core modeling architecture is a hybrid of:
- LSTM Layers: Capture temporal dependencies and sequential trends.
- Transformer Block: Utilize multi-head self-attention to understand complex patterns, relationships, and global context across sequences.

Model Architecture Includes:

- Input layer: Accepts a 3D tensor `(batch_size, time_steps=30, features=10)`
- One LSTM layer with drop out regularsation
- Transformer encoder block with:
  - Multi-head attention
  - Layer normalization
  - Residual connections
  - Feed-forward dense layers
- Output: Dense layer for multi-output regression of next-period returns

4. Model Training

The model is compiled with the Mean Squared Error (MSE) loss and optimized using the Adam optimizer. Training includes:

- `ModelCheckpoint` to save the best-performing model
- `EarlyStopping` to prevent overfitting

Recommended to use following based upon inputs from SME
- `DirectionalAccuracyCallback` to measure the financial significance of predictions (i.e., correctly predicting the direction of asset returns)

5. Hyperparameter Tuning

`Keras-Tuner` is employed to automatically tune:
- Number of LSTM units
- Number of attention heads
- Dropout rates
- Learning rate

The search is guided by validation loss and directional accuracy, exploring various architectures for the best generalization performance. Each model with the performance result during training is being logged for error analysis and more targeted tunings. Refer Para 9, 10 below for further enhancements.

6. Automation with Airflow

The pipeline is automated using Apache Airflow, with two primary DAGs:
- Training DAG (`train_ai_finance_model`): Executes weekly to retrain the model using updated data in offline setting without hampering the prediction setup.
- Prediction pipeline (`predict_latest`): Executes to generate forecasts using the latest model once UI is triggered (Online setup). This can further be made off-line by creating DAG that runs daily and Fast API can be setup for display based upon the last predicted outputs.

The DAG consists of tasks like:
- Data loading
- Preprocessing
- Model loading or training
- Forecast generation
- Metric logging

7. Streamlit Interface

A lightweight `Streamlit` Basic UI dashboard allows users to:
- Trigger Prediction pipeline manually
- Monitor portfolio performance on dashboard on selected metric.
- Display recent performance (optional)

This Basic UI helps non-technical users or analysts interact with the forecasting engine.

8. Evaluation Metrics

Model performance is evaluated using both statistical and financial metrics:

- Directional Accuracy: Percentage of predictions that correctly forecast the direction (up/down) of return.
- Cumulative Return: Tracks compounded investment return over time.
- Annualized Return & Volatility: Standard financial performance indicators.

- Sharpe Ratio: Risk-adjusted return metric.
- Max Drawdown: Maximum observed loss from a historical peak.

These metrics ensure predictions are both statistically valid and financially meaningful.

9. Future Scope and Recommendations

This implementation serves as a boilerplate setup for a larger, production-grade forecasting system. The current structure offers modularity, basic automation, and decent performance. However, there is significant room for further exploration:

- Advanced Hyperparameter Search: The tuning can be extended to include attention dimensions, learning rate schedulers, layer stacking, and residual depths.
- Data Preprocessing: While percentage change (`pct_change`) helps mitigate some of the issues of non-stationarity, full preprocessing should be informed by statistical tests like ADF (Augmented Dickey-Fuller) to assess stationarity.
    - Based on ADF results, dynamic or asset-specific window sizes can be determined.
    - Additional transformations (e.g., differencing, log returns) could further stabilize the series.

- Cross-Asset Interaction Modeling: Incorporate correlations and lead-lag patterns between asset pairs.

- Model Enhancements:
    - Attention-on-attention mechanisms
    - Temporal convolution layers
    - Transformer-only variant (e.g., Informer or TFT)

- Backtesting Integration: Link forecasts with trading strategies (e.g., long-short positions) for simulated returns.

- Explainability: Implement techniques like SHAP values or attention visualizations to interpret feature influence.

Highly Recommended Next Steps:
- Perform full stationarity analysis using ADF and define preprocessing strategies accordingly.
- Extend Keras-Tuner to optimize over architecture depth, dropout, and embedding dimensions.
- Evaluate with rolling window validation instead of static train/test splits.
- Introduce external features like macroeconomic indicators or sentiment scores.

10. Additional Observations and Constraints

With the increase in model complexity—particularly the inclusion of both LSTM and Transformer components—the computational demand grew considerably. Due to practical resource constraints, full-scale precision hyperparameter tuning could not be comprehensively explored. The tuning process was restricted to only **95 combinations of various hyperparameter configurations**, and each configuration was trained for **35 epochs**.

Despite this limitation, the best-performing model within this constrained search space has been referenced in this report. It is important to note that **both training and**

validation losses were generally on a convergence trajectory, suggesting that the models were trending toward optimal performance. It is likely that extended training duration or a broader hyperparameter search space could have helped the models approach a **global minima**, potentially yielding more accurate and robust predictions.
This aspect opens a clear path for future enhancement where more computational resources can facilitate:

- Extended epoch training (with scheduled learning rate decay)
- Larger and more granular hyperparameter search spaces
- Advanced scheduling or distributed training setups for parallel experimentation

Such improvements would be particularly beneficial in production-grade financial modeling scenarios where even marginal accuracy gains can yield significant financial returns.

12. Conclusion

This project successfully demonstrates a modular, hybrid deep learning approach to financial time series forecasting. By combining the strengths of LSTMs and Transformers, the system captures both temporal and contextual patterns. Integration with Airflow ensures automation, while Streamlit provides accessibility. With proper tuning and incremental updates, the model holds promise for use in real-world quantitative finance and trading systems.