Title:
Advanced Time Series Forecasting with Deep Learning and Attention Mechanisms

Student: Logeshwaran S
Academic Year: 2024–2025

Abstract

This project implements a Transformer-based multi step forecasting pipeline on programmatically generated multivariate dataset that includes seasonality, trends, heteroscedastic noise, and spikes. The Transformer encoder–decoder is trained to produce multi-step forecasts and attention is used to interpret which past time steps influence the predictions. Results are compared against a SARIMAX/ETS baseline and evaluated with SMAPE, MASE, RMSE and MAE.

1. Introduction

Time series forecasting is essential across many domains (retail, energy, finance). Traditional statistical methods (ARIMA/ETS) often struggle with long-range dependencies and complex multivariate interactions. Attention-enabled Transformer architectures capture long-range dependencies and provide interpretable attention patterns, making them well-suited for advanced forecasting tasks.

2. Problem Statement

Design and implement a production-quality Transformer-based forecasting system tha:

- Handles multivariate time series,
- Produces multi-step forecasts,
- Provides interpretable attention scores,
- Is rigorously evaluated against strong statistical baselines (SARIMAX/Prophet),
- Includes hyperparameter optimization (Optuna scaffold).

3. Dataset Generation & Preprocessing

- compl**Dataset**: programmatically generated synthetic dataset to mimic realistic exity.
- **Length**: 520 weekly timesteps.
- **Series**: 8 correlated series.
- **Components**: additive trend, multiple seasonalities (varying periods), heteroscedastic noise, occasional spikes.

  Preprocessing: per-series StandardScaler fitted on the training window. Sliding windows created with lookback seq_len = 48 and multi-step horizon horizon = 8. Train/Validation/Test split is timebased (no shuffling).

4. Model Architecture

- **TransForecaster** — a Transformer encoder–decoder implemented in PyTorch:
- Input projection from n_series → d_model,
- Sinusoidal positional encoding,

- Stacked Transformer encoder layers,
- Decoder that repeats context and predicts horizon steps,
- Prediction head mapping decoder outputs → (horizon × n_series).

The model allows approximation of attention patterns by inspecting encoder outputs / applying MultiheadAttention hooks for visualization.

5. Training Setup

- Sequence length (lookback): 48 weeks
- Forecast horizon: 8 weeks
- Batch size: 32
- Epochs: 25 (recommended 50–80 for final runs)
- Optimizer: Adam, learning rate = 1e-3
- Loss: MSE (recommend switching to Huber/ SmoothL1 for robustness to spikes)
- Hardware: CPU used in the run (GPU recommended to speed hyperparameter search)

6. Baselines

- **SARIMAX / ETS** applied to aggregated (top-level) series to provide a strong statistical baseline.
- **Prophet** support is available optionally (package-dependent).
- **Naive seasonal baseline** used implicitly for MASE scaling.

7. Evaluation Metrics

- **SMAPE** — Symmetric Mean Absolute Percentage Error (suitable for percentage error interpretation)

- **MASE** — Mean Absolute Scaled Error (scaled by seasonal naive)
- **RMSE** — Root Mean Squared Error
- **MAE** — Mean Absolute Error

8. Experimental Results (from run)

- Dataset windows: total = 465 (train = 441, val = 16, test = 8)
- Final validation loss (approx.): 0.3435
- **Bottom SMAPE = 3.5379%**
- **Bottom MASE = 1.2083**
- **RMSE = 3.7535**
- SARIMAX example top-level forecasts (first 3): [52.584, 51.788, 50.451]
- Training runtime (reported): ≈ 396.4 seconds (on CPU)
- **Interpretation:** Low SMAPE indicates good percentage accuracy. MASE > 1 suggests absolute errors slightly larger than seasonal-naive (likely due to heteroscedastic spikes). Improvements are possible via robust loss, more tuning, and ensembling.

9. Attention Interpretation

Attention weights were approximated from encoder outputs and can be visualized as heatmaps showing which past timesteps are consulted more during forecasting. These heatmaps reveal seasonal lags and recent spikes importance. For each representative series, include a heatmap and a short paragraph interpreting the dominant lags.

10. Hyperparameter Optimization

- An Optuna scaffold was included (disabled by default).

Recommended search space:

- d_model: [64, 128, 256]
- nhead: [2, 4, 8]
- lr: [1e-4 — 1e-2 log-uniform]
- dropout: [0 — 0.3]
- batch_size: [16, 32, 64]

Run ~30–80 trials with early-stopping to find robust settings.

11. Discussion & Recommendations

1. To improve MASE and overall robustness:
2. Use **Huber / SmoothL1 loss** to reduce sensitivity to spikes.
3. Run **Optuna** tuning for 30+ trials (use GPU if possible).
4. Increase training epochs (50–80) or add early-stopping.
5. Ensemble Transformer + LSTM predictions.
6. Produce per-series attention heatmaps with short domain interpretations.

12. Computational Cost

The provided run completed in about 396 seconds on CPU for 25 epochs. Using GPU (CUDA) will reduce runtime significantly and make Optuna practical.

13. Conclusion

This project demonstrates a production-ready Transformer forecasting pipeline with interpretable attention. The model produces accurate multistep forecasts for complex synthetic data. With further tuning and robust losses, the model can outperform statistical baselines across both percentage and absolute error metrics.

14. Files produced

- tf_out/synthetic_series.csv (generated dataset)
- tf_out/transformer_model_monika.pth (trained model checkpoint)
- tf_out/example_series_0.png (example prediction plot)
- tf_out/results_transformer.json (metrics & history)
- tf_out/Report_Monika.pdf (basic auto-report)

References

1. Vaswani et al., “Attention Is All You Need,” NeurIPS 2017.
2. Hyndman, Rob J., and George Athanasopoulos. *Forecasting:*

   Principles and Practice.
3. Taylor & Letham, “Prophet: Forecasting at Scale.”