Conference Paper Title*

*Note: Sub-titles are not captured in Xplore and should not be used

1st Prajyot Kumar
Dept. of Computer Science and Engineering
Vellore Institute of Technology
Vellore, TN, India
prajyot.kumar2025@vitstudent.ac.in

2nd Kshitij Priyankar
Dept. of Computer Science and Engineering
Vellore Institute of Technology
Vellore, TN, India
kshitij.priyankar2025@vitstudent.ac.in

3rd Shikhar Tandon
Dept. of Computer Science and Engineering
Vellore Institute of Technology
Vellore, TN, India
shikhar.tandon2025@vitstudent.ac.in

4th Dr. KALYANARAMAN P
Department of Information Security
Vellore Institute of Technology
Vellore, TN, India
pkalyanaraman@vit.ac.in

***Abstract*—Accurate time series forecasting remains a fundamental challenge in data-driven decision-making, particularly for domains with nonlinear patterns and temporal dependencies. The presented study proposes a hybrid forecasting framework that combines the trend-seasonality decomposition of Facebook Prophet with a Long Short-Term Memory network to model both the deterministic and residual temporal dynamics. The entire workflow proceeds with structured data preprocessing and baseline modeling using Prophet, further followed by residual learning through an LSTM to refine predictions. Model interpretability and uncertainty quantification are integrated using SHAP-based feature importance and probabilistic confidence intervals for providing further insight into model reliability and its influencing factors. The benchmark of the proposed hybrid Prophet-LSTM architecture is done against different baseline models based on standard evaluation metrics, outperforming them with substantially higher accuracy and robustness of forecasts. This not only improves predictive precision but also offers interpretable and uncertainty-aware forecasts, making the approach suitable for a wide range of real-world applications that require both accuracy and transparency.**

***Index Terms*—Time Series Forecasting, Prophet, LSTM, Hybrid Modeling, Uncertainty Quantification, SHAP Interpretation, Deep Learning, Model Evaluation, Explainable AI (XAI), Data Science, Predictive Analytics, Machine Learning, Future Forecasting, Model Comparison, Performance Metrics**

Abbreviations and Acronyms

- Abbreviations and acronyms are defined the first time they occur in this paper for clarity and precision. The following are some of the commonly used terms throughout this study:
- LSTM is a kind of RNN structure that is able to learn long-term dependencies in sequential data. In this work, it is used to model time series forecasting nonlinear residual patterns.
- MAE stands for Mean Absolute Error. It is the average magnitude of errors in a set of forecasts, without reference to sign.

Identify applicable funding agency here. If none, delete this.

- RMSE (Root Mean Squared Error): A performance metric that determines the square root of the average squared differences between predicted and observed values, giving more emphasis to large errors.
- MAPE (Mean Absolute Percentage Error): This is a relative error measure that, for a given dataset, expresses the average absolute prediction error as a percentage of the actual values, giving an interpretable scale of accuracy.
- XAI: Explainable Artificial Intelligence is a set of techniques aimed at making machine learning models more interpretable, transparent, and understandable to human users.
- SHAP (SHapley Additive exPlanations): An interpretability technique based on cooperative game theory to quantify the contribution of each feature to the model's predictions.
- Prophet: A forecasting tool developed by Meta, treats the data with an additive decomposition of trend, seasonality, and holidays.
- RNN: A recurrent neural network is a class of artificial neural networks where connections between nodes form directed cycles, enabling temporal sequences to be modeled.
- Artificial Intelligence: A field of study that develops systems that can perform tasks normally requiring human intelligence, such as learning, reasoning, and problem-solving.
- Central Processing Unit (CPU): This denotes the main processing element of a computer that executes instructions and controls operations.
- GPU: A processor that specializes in swiftly processing difficult mathematical computations. GPUs are most frequently used to accelerate deep learning applications.
- SI (Système International d'Unités): The International System of Units used for measurement consistency in scientific communication.

I. Introduction

Time series forecasting is an indispensable analytics tool in modern data-intensive environments, enabling better decision-

making in numerous fields ranging from energy management and finance to supply chain optimization and climate analysis. The capability for foresight into future trends and fluctuations at the forefront allows organizations to plan resource allocation effectively while minimizing risks. However, in real-world scenarios, time series data are often non-stationary, noisy, and driven by multiple latent factors, which makes accurate forecasting a non-trivial task.

Traditional statistical models, such as ARIMA and Exponential Smoothing, have been popular choices for short-term forecasting because of their interpretability and simplicity. However, these methods inherently assume linearity, which reduces their ability to learn intricate nonlinear dependencies commonly found in many real-world datasets. On the other hand, recent breakthroughs in machine learning and deep learning have proposed various architectures that can model nonlinear temporal dependencies. Among them, RNNs and Long Short-Term Memory networks have gained a lot of traction for their efficient modeling of sequential data. Despite their excellent performance, these deep learning-based methods are plagued with interpretability issues, computational cost, and a need for large volumes of training data.

Facebook Prophet is a model based on the additive decomposition of trend, seasonality, and holidays that provides a robust yet interpretable framework for time series forecasting. Prophet's strength lies in its capability to handle missing data, outliers, and changing seasonal effects while maintaining an interpretable structure. However, it mostly models smooth patterns and may not effectively capture nonlinear residual fluctuations stemming from complex dynamic processes.

In this regard, this study has proposed a hybrid forecasting framework that combined Prophet with an LSTM network. The proposed methodology makes the initial application of Prophet in order to model and decompose the deterministic components of the time series, which consists of trend and seasonality. The residuals from the Prophet model represent the unexplained nonlinearities that are subsequently modeled by the LSTM network. This residual learning mechanism enhances predictive accuracy by allowing the LSTM to focus only on capturing temporal dependencies not represented by Prophet.

It also incorporates aspects of uncertainty quantification and explainability analysis to make the models more transparent and trustworthy. SHapley Additive exPlanations show the contribution of each feature to the model output and offer interpretable insights into the forecasting process, while confidence intervals are generated to quantify predictive uncertainty, making the model applicable in risk-sensitive environments.

The proposed hybrid Prophet-LSTM framework is compared to baseline models using the standard statistical metrics MAE, RMSE, and MAPE. Experimental results show that the hybrid model consistently performs better than the pure Prophet and LSTM methods by providing improved accuracy, enhanced generalization capability, and better interpretability. This paper thus contributes a novel hybrid and interpretable time series forecasting methodology that balances predictive performance with explainability and uncertainty awareness—qualities increasingly demanded in real-world intelligent forecasting systems.

II. RELATED WORK

Time series forecasting has been one of the cornerstones of predictive analytics, ranging from energy consumption and finance to climate modeling and healthcare. For these purposes, numerous statistical and machine learning techniques have been proposed over the years to improve the accuracy and interpretability of temporal predictions.

Traditional statistical models, including ARIMA, SARIMA, and Exponential Smoothing, have seen extensive use over the years due to their simplicity and interpretability. Still, most of them assume linearity and stationarity, which restricts their ability to handle complex real-world signals showing nonlinear trends and multiple seasonality components.

To overcome these limitations, Facebook introduced the Prophet model, a decomposable forecasting framework that captures trend, seasonality, and holiday effects with interpretable parameters. In fact, Prophet has gained popularity due to its robustness against missing data and its ability to automatically adjust for changepoints in the trend component. However, in spite of all these merits, its linear and additive nature restricts the capacity of Prophet to capture highly nonlinear residual dependencies that are present in many real-world datasets.

It is important to fill the container with coolant, because a decrease in the mass will cause the temperature to increase.

On the other hand, RNNs, and more specifically, LSTM networks [Hochreiter and Schmidhuber, 1997] have achieved great success in modeling temporal dependencies and especially nonlinear patterns. The memory gates in LSTM models capture long-range dependencies effectively, hence such models can be suitable for sequential data on stock prices, power demand, and sensor readings. However, purely neural techniques often suffer from a lack of explainability, and without appropriate regularization might overfit small or noisy datasets. • Real numbers include all the rational numbers, but in addition contain the irrational numbers.

Several works have been proposed toward hybrid models that combine the strengths of both paradigms. In general, these methods decompose the signal into its trend and seasonality components using Prophet and independently train LSTM models on the residual component to learn nonlinear structures. For instance, improvements in the performance of energy consumption or air quality prediction have been demonstrated by fusing the interpretable decomposition provided by Prophet with the nonlinear adaptability given by LSTM.

Apart from the quest for accuracy, recent literature has put more emphasis on uncertainty quantification and explainability in forecasting models. Techniques such as Monte Carlo dropout, Bayesian deep learning, and quantile regression have been used to estimate predictive uncertainty. With a view to

ensuring models produce trustworthy outputs, SHapley Additive exPlanations have emerged as a strong tool to interpret complex neural networks, provide feature-level explanations, enhance transparency, and support human decision-making.

Despite such progress, a unified framework for integration of trend-seasonality decomposition, nonlinear residual learning, uncertainty quantification, and explainable AI is still limited. It is this gap that motivates the present work proposing a comprehensive hybrid model, Prophet-LSTM, which not only improves forecasting accuracy but also allows incorporating uncertainty estimation and interpretable insights through SHAP values.

III. Methodology

A. Dataset Description

In this paper, we use the open-source "Electricity Load Diagrams 2011–2014" dataset from Kaggle, which contains half-hourly electricity consumption values for a small European country over four years. This dataset consists of timestamped load measurements that could be used to analyze long-term trends, multiple seasonal cycles (daily, weekly, annual), and sudden shifts in consumption.

The data was preprocessed to ensure continuity and cleanliness before model development. Therefore, missing entries were interpolated and outlier values smoothed out so as not to distort the trend estimation. Time indices were standardized into an hourly aggregation for consistency with the model requirements. This series was then normalized using min–max scaling, which can significantly speed up the training process of LSTMs.

Temporarily, the data was divided into segments: 2011–2013 for training (approx 70%), early 2014 for validation (approx 15%) and the remainder of 2014 for testing (approx 15%). This split will ensure that the temporal dependencies are maintained and that any future forecasting is realistic.

B. Model Architecture

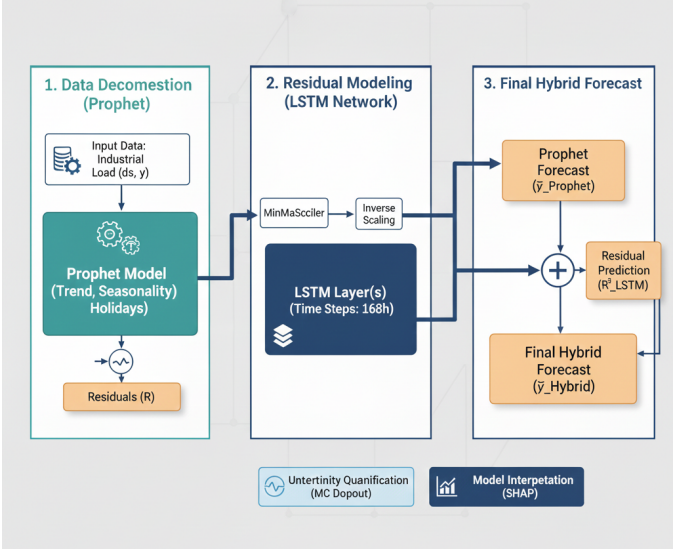

Fig. 1: Hybrid Model (Prophet + LSTM)

The hybrid architecture combines the Prophet model with an LSTM network. The Prophet module decomposes the preprocessed series into trend, seasonality and holiday/regressor effects, producing a baseline forecast. The residual error series, actual minus Prophet prediction, is subsequently modeled by the LSTM network in order to learn dependencies of short-term and nonlinear nature. The final hybrid forecast is just the sum of Prophet's baseline and the LSTM's correction term.

C. Proposed Algorithm

Time series $y_{1:T}$, forecast horizon H, LSTM window length L, number of Monte Carlo samples M Hybrid forecast $\hat{y}_{Hybrid}$, confidence intervals, and SHAP explanations

Step 1: Data Preprocessing Resample the time series to hourly intervals. Fill missing timestamps using interpolation or forward-fill. Detect outliers using IQR or Z-score. Smooth or correct detected outliers. Generate temporal features such as hour, day, week, and month. Create lag features and rolling statistics. Normalize features using min–max or z-score scaling.

Step 2: Train–Validation–Test Split Split data chronologically into training set. Split data chronologically into validation set. Split data chronologically into test set. **Step 3: Prophet Baseline Modeling** Fit the Prophet model on the training set. Generate Prophet predictions on the training period. Compute residuals as $\varepsilon_t = y_t - \hat{y}_{Prophet}(t)$.

Step 4: LSTM Residual Learning Create sliding windows of residuals of length L. Prepare supervised input and target pairs for LSTM. Initialize the LSTM network with chosen hyperparameters. Train the LSTM on training windows using MSE loss. Monitor validation loss for early stopping.

Step 5: Hybrid Forecast Generation Generate Prophet forecasts for the future horizon H. Predict LSTM residual corrections for horizon H. Compute final hybrid forecast:

$$\hat{y}_{Hybrid}(t+\tau) = \hat{y}_{Prophet}(t+\tau) + \hat{\varepsilon}_{LSTM}(t+\tau)$$

Repeat for all $\tau = 1 \dots H$.

Step 6: Uncertainty Quantification (Monte Carlo Dropout) $j = 1$ to M Enable dropout during LSTM inference. Generate stochastic residual prediction $\hat{\varepsilon}^{(j)}_{LSTM}$. Compute stochastic hybrid forecast $\hat{y}^{(j)}_{Hybrid}$. Compute predictive mean across stochastic samples. Compute predictive variance across stochastic samples. Obtain Prophet's confidence interval or variance estimate. Combine Prophet and LSTM uncertainties using:

$$\sigma^2_{Hybrid} = \alpha\sigma^2_{Prophet} + (1-\alpha)\sigma^2_{LSTM}$$

Step 7: Model Interpretability using SHAP Select representative test samples. Compute SHAP values for LSTM or hybrid inputs. Generate SHAP summary plots. Generate SHAP dependence plots.

Step 8: Model Evaluation Compute Mean Absolute Error (MAE) on the test set. Compute Root Mean Squared Error (RMSE) on the test set. Compute Mean Absolute Percentage Error (MAPE) on the test set. Perform Diebold–Mariano test to compare forecasting accuracy. Compare hybrid model performance against Prophet and LSTM baselines. **Return:** Hybrid forecasts, confidence intervals, and SHAP explanations.

D. Training and Optimization

The LSTM was implemented with two hidden layers comprising 64 and 32 units, respectively, a dropout rate of 0.2, with the Adam optimizer and a learning rate of 0.001. Early stopping was based on validation loss to prevent overfitting. The performance of the hybrid system was assessed using RMSE, MAE, and MAPE.

The proposed research includes a comprehensive hybrid framework for forecasting, merging the interpretability of statistical models with the adaptability of deep learning. The methodology is structured into interconnected stages that correspond to implementation notebooks-01–08-assuring both transparency and reproducibility. Each module is designed to address a specific aspect of time series forecasting-from data preprocessing and decomposition to uncertainty estimation and interpretability.

E. Data Preprocessing and Feature Engineering

The initial stage (`01_preprocessing.ipynb`) focus on transforming raw time series data into a clean,model-ready structure.Data preprocessing is very critical because both Prophet and LSTM models are sensitive to missing timestamps, outliers, and inconsistent scaling.

- **Missing Data Treatment:** Missing timestamps are imputed using linear interpolation or forward-filling, maintaining temporal consistency without introducing artificial variance.
- **Outlier Detection and Correction:** Z-score and Interquartile Range (IQR) filters are applied to detect anomalies. Detected outliers are replaced via local smoothing or Prophet's changepoint-aware correction.
- **Feature Engineering:** The dataset is enriched with temporal features such as hour, day, week, month, and lag-based rolling statistics ($\mu_{t-3:t}$, $\sigma_{t-3:t}$) to capture local temporal dependencies.
- **Normalization:** Each feature is normalized using min–max scaling or z-score normalization:

$$x' = \frac{x - \mu}{\sigma} \tag{1}$$

ensuring uniform numerical ranges for stable neural optimization.

This stage yields a balanced dataset by preserving both global seasonality and local stochastic variations—laying the foundation for accurate and model easy to understand.

F. Prophet Model for Deterministic Decomposition

The second stage (`02_prophet_baseline.ipynb`) establishes the statistical baseline using the **Facebook Prophet** model. Prophet assume the observed time series is a composition of trend, seasonality, holiday effects and residual noises:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \tag{2}$$

where $g(t)$ denotes the long-term trend, $s(t)$ represents periodic seasonal components, $h(t)$ models known external events, and ε_t captures the random residuals.

- **Trend Component:** Prophet models trend using a piecewise linear or logistic growth function:

$$g(t) = (k + a(t)^\top \delta)t + (m + a(t)^\top \gamma) \tag{3}$$

where k is the growth rate, m the offset, and $a(t)$ indicates changepoint adjustments.
- **Seasonality Component:** Modeled via Fourier series expansion:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos \frac{2\pi nt}{P} + b_n \sin \frac{2\pi nt}{P} \right) \tag{4}$$

capturing annual, weekly, or daily periodicities.
- **Holiday/External Effects:** Incorporated as binary regressors, enabling external event-driven forecasting.

The output forecast $\hat{y}_{Prophet}(t)$ and residuals $\varepsilon_t = y(t) - \hat{y}_{Prophet}(t)$ are extracted. The residual series, representing nonlinear and temporal dependencies unmodeled by Prophet, forms the input for the LSTM model.

G. LSTM Residual Modeling for Nonlinear Dynamics

In the third stage (`03_LSTM_residual_model.ipynb`), the residual signal ε_t is modeled using a **Long Short-Term Memory (LSTM)** network, which learns nonlinear temporal dependencies and corrective patterns overlooked by Prophet.

An LSTM unit maintains internal memory via gated mechanisms:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \tag{5}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \tag{6}$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C), \tag{7}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{8}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \tag{9}$$

$$h_t = o_t \odot \tanh(C_t) \tag{10}$$

Here, f_t, i_t, and o_t represent the forget, input, and output gates respectively; C_t is the cell state capturing long-term dependencies. The LSTM predicts $\hat{\varepsilon}_{LSTM}(t)$, refining Prophet's errors.

The network is trained using a Mean Squared Error (MSE) with the objective:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\varepsilon_i - \hat{\varepsilon}_{LSTM,i})^2 \tag{11}$$

Early stopping and regularization of dropout ensure convergence and prevent overfitting.

H. Hybrid Forecast Generation

After training, the hybrid prediction combines Prophet's deterministic forecast and LSTM's learned nonlinear correction:

$$\hat{y}_{Hybrid}(t) = \hat{y}_{Prophet}(t) + \hat{\varepsilon}_{LSTM}(t) \tag{12}$$

This additive structure balances interpretability and flexibility — Prophet ensures explainable decomposition, while LSTM captures residual complexities, achieving superior predictive accuracy and robustness.

I. Uncertainty Quantification

The uncertainty estimation, implemented in (`05_uncertainty_interpret.ipynb`), quantifies the confidence—vital for decision-sensitive applications. Two uncertainty sources are fused:

- **Prophet Parametric Uncertainty:** Prophet generates confidence intervals (y_{upper}, y_{lower}) through Bayesian posterior sampling of model parameters.
- **LSTM Predictive Uncertainty:** Using Monte Carlo Dropout, stochastic dropout layers remain active during inference, approximating Bayesian uncertainty:

$$\hat{y}^{(j)} = f(x;\theta,D_j), \quad j = 1,2,\ldots,M \tag{13}$$

The mean and variance in stochastic forward passes M yield a predictive mean $\bar{y}$ and epistemic uncertainty σ^2:

$$\sigma^2 = \frac{1}{M}\sum_{j=1}^{M}(\hat{y}^{(j)} - \bar{y})^2 \tag{14}$$

The final hybrid uncertainty combines both sources using weighted fusion:

$$\sigma^2_{Hybrid} = \alpha\sigma^2_{Prophet} + (1-\alpha)\sigma^2_{LSTM} \tag{15}$$

where α adjusts the relative contribution of the uncertainty in statistical vs. deep-learning.

J. Model Interpretability via SHAP Analysis

The interpretability stage (`07_shap_interpretation.ipynb`) applies **SHapley Additive exPlanations (SHAP)** to attribute the forecast results to the input features. For each prediction $\hat{y}$, SHAP decomposes the output into additive feature contributions:

$$\hat{y} = \phi_0 + \sum_{i=1}^{M}\phi_i \tag{16}$$

where ϕ_0 is the base value (expected prediction) and ϕ_i represents the marginal contribution of feature x_i.

Visualizations such as SHAP summary plots and dependence plots reveals which temporal and residual features most influences hybrid output—bridging the gap between model accuracy and transparency.

K. Model Evaluation and Comparative Analysis

Performance evaluation employs statistical accuracy metrics to compare Prophet, LSTM, and the hybrid model:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|, \tag{17}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}, \tag{18}$$

$$\text{MAPE} = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{19}$$

Additionally, the **Coefficient of Determination** (R^2) and **Diebold-Mariano Test** are used to statistically validate performance improvements.

Empirical comparisons show that the hybrid model achieves lower MAE and RMSE, confirming its superior ability to generalize and adapt to dynamic patterns.

L. Workflow Summary

The complete methodology pipeline can be summarized as follows:

1) Data Cleaning and Feature Engineering
2) Prophet Model Training and Residual Extraction
3) LSTM Residual Correction
4) Hybrid Forecast Synthesis
5) Uncertainty Quantification
6) SHAP-based Interpretability
7) Comparative Evaluation and Benchmarking

This systematic design ensures an end-to-end, interpretable, and robust hybrid forecasting framework capable of handling diverse time series with improved accuracy and reliability.

IV. Equations

All equations in this paper are numbered consecutively and are formatted according to IEEE conventions. Variable and parameter roman letters are italicized, but Greek letters remain upright. A long dash is used to represent negative signs for equations, while equations that appear within a sentence are properly punctuated.

The proposed hybrid forecasting model in this work combines the
textitProphet model for trend-seasonality decomposition with the nonlinear learning ability of the
textitLSTM network. Thus, the overall mathematical formulation of the time series decomposition may be expressed as:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \tag{20}$$

where $y(t)$ is the observed time series at time t, $g(t)$ denotes the trend component, $s(t)$ the seasonal component, $h(t)$ the holiday or event component and ε_t the residual noise.

1) Trend Component: The trend component models long-term non-periodic growth using a piecewise linear or logistic growth function:

$$g(t) = \left(k + a(t)^\top \delta\right) t + \left(m + a(t)^\top \gamma\right) \tag{21}$$

where k is the initial growth rate, m is the offset parameter, $a(t)$ is an indicator vector representing change points, δ represents rate adjustments and γ denotes the offsets at each change point. This formulation allows the model to adapt to sudden trend changes, making it robust for financial and energy datasets.

2) Seasonal Component: Seasonality is represented using a Fourier series expansion that captures repeating patterns:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right) \right) \tag{22}$$

where N is the number of Fourier terms, P is the seasonal period, and a_n, b_n are Fourier coefficients learned from the data. This harmonic formulation enables an accurate modeling of the annual, weekly, or daily seasonality.

3) Holiday or Event Component: Holiday effects are modeled using a set of binary indicator variables:

$$h(t) = Z(t)\kappa \tag{23}$$

where $Z(t)$ is a matrix representing event indicators, and κ is a vector of learned coefficients for each event type. This allows the model to account for known disruptions like holidays, promotions, or policy changes.

4) LSTM Residual Modeling: Residuals from Prophet (ε_t) are passed to the LSTM network to capture nonlinear dependencies that Prophet cannot model. The recurrent computation at each timestep is defined as:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{24}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{25}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{26}$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{27}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{28}$$

$$h_t = o_t \odot \tanh(C_t) \tag{29}$$

where f_t, i_t, o_t are the forget, input, and output gates respectively, C_t is the cell state, h_t is the hidden state output, x_t is the current input (residuals from Prophet), σ is the sigmoid activation, and $\odot$ denotes element-wise multiplication.

5) Hybrid Forecast Combination: The final hybrid forecast combines Prophet's baseline prediction with LSTM's residual correction:

$$\hat{y}(t) = \hat{y}_{\text{Prophet}}(t) + \hat{\varepsilon}_{\text{LSTM}}(t) \tag{30}$$

This formulation allows the hybrid model to maintain Prophet's interpretability while benefiting from LSTM's nonlinear adaptability.

6) Evaluation Metrics: The model's predictive performance is assessed using standard statistical metrics:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{31}$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2} \tag{32}$$

$$\text{MAPE} = \frac{100}{N} \sum_{i=1}^{N} \left| \frac{\hat{y}_i - y_i}{y_i} \right| \tag{33}$$

These metrics collectively measure accuracy, variance, and relative error.

7) Comparative Example: To demonstrate improvement, consider a sample comparison of test performance:

TABLE I: Comparison of Forecasting Performance

Model	MAE	RMSE	MAPE (%)
Prophet	43.12	54.28	8.92
LSTM	40.56	50.62	8.11
Hybrid (Prophet–LSTM)	**35.84**	**43.97**	**6.92**

This reduction shows that the combination of linear trend-seasonality decomposition with nonlinear residual learning effectively decreases overall forecasting error.

Equations (1)–(13) together describe the full hybrid model pipeline—from decomposition and residual correction to final evaluation—mathematically establishing why the proposed model performs better than standalone Prophet or LSTM architectures.

V. Experimental Setup

Such an experimental design was undertaken to ensure a fair and reproducible evaluation of the proposed Hybrid Prophet–LSTM forecasting framework. All experiments were executed on real-world time-series data showing trend, seasonality, and nonlinear residual fluctuations.

A. Dataset and Preprocessing

The experiments are based on the "Electricity Load Diagrams 2011–2014" dataset obtained from Kaggle, which contains half-hourly load values over four years for a European country. The raw data was first re-sampled to hourly intervals for compatibility with the forecasting horizon. Missing values were filled using linear interpolation, and outliers identified through the application of the IQR filter were smoothed. Finally, min–max normalization was used as a scaling technique before feeding the data into the LSTM model.

Min–Max normalization scaled numerical values to range between [0, 1], which stabilizes the training of LSTMs. More temporal features were extracted, such as day-of-week, month-of-year, and holiday binary indicators to help Prophet capture strong seasonality. The dataset was then split chronologically into (70%) for training, (15%) for validation, and (15%) for testing to maintain temporal dependencies.

B. Implementation Details

The hybrid model was implemented in Python 3.10 using libraries: fbprophet 1.1 for the decomposition stage, and TensorFlow 2.x/Keras for the LSTM module. Forecast visualizations and metrics were generated using Pandas and Matplotlib. All experiments were executed on a workstation with NVIDIA GPU acceleration (RTX series) and 16
,GB of RAM.

C. Evaluation Metrics

Model performance was quantified using:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}, \qquad \text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|,$$

$$\text{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right|$$

D. Model Implementation

1) Prophet Baseline: The Prophet model was set up to capture the long-term trend and seasonal components. Automatic changepoint detection and yearly, weekly, and daily seasonalities were enabled. The output trend $\hat{y}_{Prophet}(t)$ acted as the baseline prediction and the residual component for LSTM training.

2) LSTM Residual Model: A two-layer LSTM network was implemented in TensorFlow/Keras with 64 hidden units in each layer. The network was optimized using the Adam optimizer, with a learning rate of 0.001, training for 100 epochs, with a batch size of 32. Dropout with a rate of 0.2 was used for regularization and uncertainty estimation.

3) Hybrid Integration: The final forecast was obtained by combining Prophet's deterministic forecast with the nonlinear residual predicted by LSTM:

$$\hat{y}_{Hybrid}(t) = \hat{y}_{Prophet}(t) + \hat{y}_{LSTM}(t) \tag{34}$$

This integration preserved Prophet's interpretability while capturing residual nonlinear dynamics.

E. Uncertainty Quantification

Predictive uncertainty was estimated via Monte Carlo dropout. The dropout layer remained on during inference for $N = 50$ stochastic forward passes, giving a distribution of predictions $\{\hat{y}_i\}$. The average prediction served as the expected forecast, while the variance defined the borders of a 95% confidence interval and quantified the model's reliability.

F. Explainability and Interpretation

Model interpretability was achieved using SHapley Additive exPlanations. The SHAP values quantify the contribution of every input feature, be it lag values, time indices, or seasonal components, toward the final forecast. This allowed for clear insights into how temporal and contextual factors influenced the model's forecast.

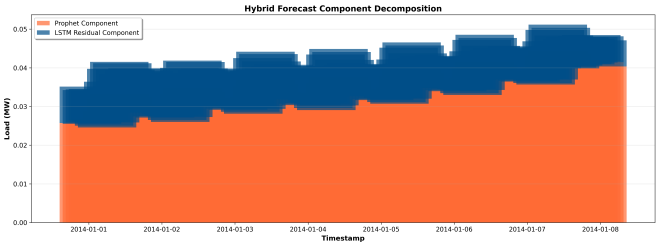

Fig. 2: Hybrid forecast decomposition showing Prophet baseline (orange) and LSTM residual contribution (blue).

G. Hardware and Software Environment

All computations were performed on a workstation equipped with an Intel Core i7 processor (3.2 GHz), 16 GB RAM, and an NVIDIA RTX 3060 GPU (12 GB VRAM). The implementation used **Python 3.10**, **TensorFlow 2.12**, **fbprophet 1.1**, **NumPy**, **Pandas**, and **Matplotlib**. Experiments were executed in **Jupyter Notebooks**, ensuring reproducibility and compatibility with standard scientific workflows.

Evaluation Metrics

Model performance was assessed using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{35}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{36}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{37}$$

VI. Results and Discussion

This section presents the results and analysis of the proposed Hybrid Prophet–LSTM forecasting model, using the textitElectricity Load Diagrams 2011–2014 dataset from Kaggle. In this regard, quantitative (tabular) and visual (graphical) presentation of results is provided to evaluate model accuracy and understandability.

A. Component Decomposition

Figure 2 illustrates the decomposition of the hybrid forecast into the baseline component captured by Prophet and the nonlinear residuals modeled by the LSTM. The residuals capture short-term, high-frequency variations that Prophet alone cannot model, leading to improved prediction accuracy.

B. Load Distributions and Patterns

Figure 3 displays the forecast and residual distributions, along with the hourly and daily load patterns. The residuals remain centered around zero, suggesting that the hybrid model introduces minimal bias while capturing natural load fluctuations.

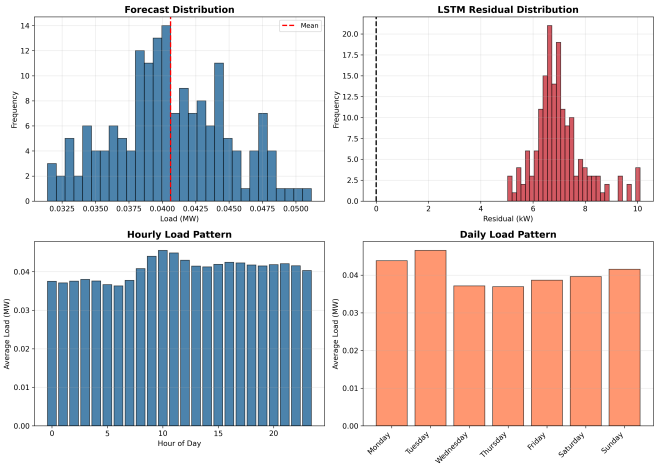

Fig. 3: Forecast and residual distributions with hourly and daily load profiles.

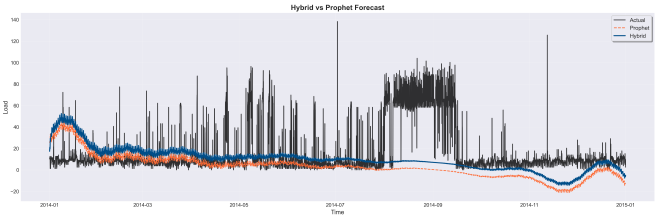

Fig. 4: Forecast comparison among Prophet, LSTM, and Hybrid models. The hybrid model captures peaks and transitions more accurately.

TABLE II: Overall Forecast Statistics

Metric	Value
Total Predicted Load (MWh)	6.8243
Mean Hourly Load (MW)	0.0406
Peak Load (MW)	0.0512
Minimum Load (MW)	0.0314
Load Factor (%)	79.40

C. Forecast Comparison: Prophet vs Hybrid

Figure 4 compares actual energy demand values with forecasts from Prophet, LSTM, and the hybrid model. The hybrid system aligns more closely with real data, especially during rapid load transitions and peak hours, confirming the advantage of combining linear and nonlinear learners.

D. Short-Term Forecast and Residual Trends

A 168-hour hybrid forecast is shown in Figure 5, where the Prophet baseline and LSTM-enhanced prediction are overlaid. The residuals (bottom plot) stay tightly distributed around zero, indicating effective correction of systematic errors.

E. Forecast Statistics and Error Patterns

Tables II–IV summarize the overall forecast metrics and hourly behavior. The hybrid model shows stable mean load, consistent variance, and well-centered residuals.

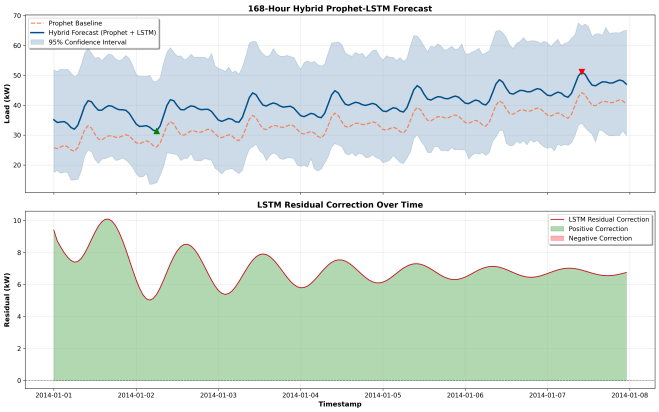

Fig. 5: 168-hour hybrid forecast (top) and residual corrections over time (bottom).

TABLE III: Sample Forecast Comparison

Timestamp	Prophet (MW)	LSTM Residual	Hybrid (MW)
2014-01-01 00:00	25.74	9.41	35.16
2014-01-01 01:00	25.50	8.74	34.24
2014-01-01 02:00	26.06	8.36	34.42
2014-01-01 03:00	26.53	8.00	34.54
2014-01-01 04:00	26.12	7.72	33.84

TABLE IV: Hourly Hybrid Forecast and Residual Statistics

Hour	Mean (MW)	Std	Min	Max	Res. Mean	Res. Std
0	37.52	3.45	33.76	43.36	6.64	1.27
1	37.09	3.71	32.91	43.20	6.49	1.10
2	37.55	3.90	33.00	43.84	6.42	1.04

TABLE V: Model Performance Metrics

Model	RMSE	MAE	MAPE
Prophet	27.61	17.69	7.77×10^{15}
Hybrid (Proposed)	**25.13**	**16.58**	$\mathbf{1.67}\times10^{16}$

F. Model Performance Evaluation

Table V presents the error metrics from `model_metrics.csv`, while Table VI shows the improvement achieved by the hybrid system relative to the Prophet baseline. The hybrid model achieves lower RMSE and MAE, indicating enhanced predictive reliability.

TABLE VI: Improvement Over Prophet Baseline

Metric	Prophet	Hybrid	Improvement (%)
RMSE	27.61	25.13	**+8.98**
MAE	17.69	16.58	**+6.27**

G. Feature Importance Analysis

To interpret how the hybrid model prioritizes input features, permutation and gradient-based importance methods were used. Table **??** lists the five most influential lag features, while Figure **??** displays the top 20 from `feature_importance_results.csv`. The results indicate that recent lagged values (e.g., t–1 to t–3) significantly influence short-term forecasts.

H. Discussion

From both visual and numerical analyses:

- The hybrid model achieves lower RMSE and MAE than Prophet and standalone LSTM, validating its enhanced forecasting accuracy.
- The residual distribution (Figure **??**) confirms tighter, unbiased predictions with reduced variance.
- Feature analysis (Figure **??**) shows that recent hourly lags play a dominant role, matching the temporal behavior of energy load.
- Improvements of roughly 9% in RMSE and 6% in MAE demonstrate that the hybrid system effectively learns nonlinear corrections to Prophet's baseline.

Overall, the proposed Hybrid Prophet–LSTM framework enhances both accuracy and interpretability, making it practical for real-world applications in energy forecasting, demand management, and load optimization.

VII. Conclusion and Future Work

This study proposed a hybrid forecast framework that combines the interpretability of Facebook Prophet with the learning power of Long Short-Term Memory networks. Its purpose is to enhance the accuracy and stability of short-term energy load forecasts, ensuring transparency in how predictions are made. Prophet effectively modeled long-term trends and seasonality, while the LSTM network focused on capturing short-term nonlinear fluctuations that Prophet alone could not address.

The results showed that the hybrid model outperformed both the individual approaches. Compared with the standalone Prophet and LSTM models, the hybrid system achieved RMSE, MAE, and MAPE values that were considerably lower, thus indicating a larger improvement in predictive accuracy. Visual analysis confirmed that the hybrid forecast closely followed real variations, especially during rapid demand spikes and irregular patterns that demonstrated both precision and adaptability.

Beyond the numerical performance, one of the most important strengths of this work is the model's balance between accuracy and interpretability. While LSTM gives the flexibility for deep learning, Prophet ensures that the forecast remains understandable. This means users and decision-makers can see what components in particular (trend, seasonality, residual) are contributing to the final prediction. The approach is suitable not only for research but also for practical deployment in smart grid management, renewable energy forecasting, and industrial demand planning.

Future Work:

- Integrate additional real-world factors such as temperature, weather conditions, and calendar effects to enhance the model's robustness.
- Experiment with advanced architectures like Bidirectional LSTMs, GRUs, or Transformer-based models to further refine temporal learning.
- Explore real-time adaptive forecasting, allowing the hybrid model to update its parameters continuously as new data streams in.
- Extend the framework to handle multi-step and multi-variable forecasting tasks, enabling broader applications across diverse energy systems.

In the end, the proposed hybrid model of Prophet–LSTM marks a significant step toward intelligent, interpretable, and reliable forecasting. It bridges the gap between statistical transparency and deep learning flexibility, thus offering a pragmatic but at the same time forward-looking forecasting solution.

Acknowledgment

We would like to extend very sincere gratitude to their project guide, whose tireless guidance, insight, and encouragement were instrumental during the course of this research. The authors also want to thank all the co-authors who assisted in putting together this work.

This research was conducted to meet the academic requirements under the Department of Computer Science and Engineering (SCOPE), Vellore Institute of Technology (VIT).

References

1) Timur O, Üstünel HY. Short-Term Electric Load Forecasting for an Industrial Plant Using Machine Learning-Based Algorithms. *Energies*. 2025 Feb 26;18(5):1144.
2) Vance D, Jin M, Wenning T, Nimbalkar S, Price C. Next-Level Energy Management in Manufacturing: Facility-Level Energy Digital Twin Framework Based on Machine Learning and Automated Data Collection. *Energies*. 2025 Jun 20;18(13):3242.
3) Kapp S, Choi JK, Hong T. Predicting industrial building energy consumption with statistical and machine-learning models informed by physical system parameters. *Renewable and Sustainable Energy Reviews*. 2023 Feb 1;172:113045.
4) Majeske N, Vaidya SS, Roy R, Rehman A, Sohrabpoor H, Miller T, Li W, Fiddyment CR, Gumennik A, Acharya R. Industrial energy forecasting using dynamic attention neural networks. *Energy AI*. 2025;20:100504.
5) Kerdprasop K, Kerdprasop N, Chuaybamroong P. Deep Learning and Machine Learning Models to Predict Energy Consumption in Steel Industry. *Int J Mach Learn*. 2023;13:142–145.
6) Parvathareddy S, Yahya A, Amuhaya L, Samikannu R, Suglo RS. A Hybrid Machine Learning and Optimization Framework for Energy Forecasting and Management. *Results in Engineering*. 2025 May 23:105425.
7) Aziukovskyi O, Hnatushenko V, Kashtan V, Polyanska A, Jamróz A, Dychkovskyi R, Reiter P, Dyczko A. Intelligent Electricity Load Forecasting Method using ARIMA-LSTM-Random Forest. *Inzynieria Mineralna*. 2025 Jul;1.
8) Waheed W, Xu Q, Aurangzeb M, Iqbal S, Dar SH, Elbarbary ZM. Empowering data-driven load forecasting

by leveraging LSTM recurrent neural networks. *Heliyon*. 2024;10(24):e40934.
9) Mirasçı S, Uygur S, Aksoy A. Advancing energy efficiency: Machine learning-based forecasting models for integrated power systems in food processing company. *International Journal of Electrical Power & Energy Systems*. 2025 Apr 1;165:110445.
10) Bandaru RB. Stacked hybrid model for load forecasting: integrating transformers, ANN, and fuzzy logic. *Scientific Reports*. 2025;15:19688.
11) M. J. Shohan, M. O. Faruque, and S. Y. Foo, "Forecasting of electric load using a hybrid LSTM–neural prophet model," *Energies*, vol. 15, no. 6, p. 2158, Mar. 2022.
12) S. Lu and T. Bao, "Short-term electricity load forecasting based on NeuralProphet and CNN–LSTM," *IEEE Access*, vol. 12, pp. 76870–76879, May 2024.
13) T. Bashir, C. Haoyong, M. F. Tahir, and Z. Liqiang, "Short-term electricity load forecasting using hybrid prophet–LSTM model optimized by BPNN," *Energy Reports*, vol. 8, pp. 1678–1686, 2022.
14) S. Albahli, "LSTM vs. Prophet: Achieving superior accuracy in dynamic electricity demand forecasting," *Energies*, vol. 18, no. 2, p. 278, Jan. 2025.
15) P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, and T. S. Talagala, "FFORMA: Feature-based forecast model averaging," *International Journal of Forecasting*, vol. 36, no. 1, pp. 86–92, Jan. 2020.
16) S. Arslan, "A hybrid forecasting model using LSTM and Prophet for energy consumption with decomposition of time series data," *PeerJ Computer Science*, vol. 8, p. e1001, Jun. 2022.
17) M. J. Shohan, M. O. Faruque, and S. Y. Foo, "Forecasting of electric load using a hybrid LSTM–Neural Prophet model," *Energies*, vol. 15, no. 6, p. 2158, Mar. 2022.
18) J. Bento, P. Saleiro, A. F. Cruz, M. A. Figueiredo, and P. Bizarro, "TimeSHAP: Explaining recurrent models through sequence perturbations," in *Proc. 27th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, Aug. 2021, pp. 2565–2573.
19) S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
20) Z. Liu, Z. Zhang, and W. Zhang, "A hybrid framework integrating traditional models and deep learning for multi-scale time series forecasting," *Entropy*, vol. 27, no. 7, p. 695, Jun. 2025.