Technical Report: Multi-Client Hybrid Bidirectional LSTM Analysis

Power Grid Analysis Team

December 09, 2025

Abstract

This report details the implementation of a **Multi-Client Feature Fusion Bidirectional LSTM** architecture for Short-Term Load Forecasting (STLF). By fusing dynamic time-series data with static client embeddings, the model learns shared behaviors across 370 clients while maintaining individual specificity. The model achieves an R^2 of 0.9897 and a MAPE of 12.71%.

1 Introduction

This report analyzes a hybrid Bidirectional LSTM (Bi-LSTM) architecture designed to learn electricity consumption behaviors of distinct clients within a single unified model. The model relies on a "Feature Fusion" strategy that combines the dynamic nature of time series data with the static characteristic attributes (ID Embeddings) of each customer.

2 Data Preparation and Input Features

A critical factor in the model's success is the transformation of raw data into meaningful features. The model's time-series input consists of **7 channels** (features) per time step.

2.1 Dynamic Features (7 Channels)

Based on the `add_time_features` and `process_part` functions in the codebase, the input features are defined as follows:

Table 1: Dynamic Input Features

Feature Name	**Formula/Description**	**Reasoning**
Scaled Consumption	Normalized Load at t	Main signal to be learned.
Hour Sin	$\sin(\frac{2\pi \times hour}{24})$	Makes time cyclical. Teaches proximity of 23:00 to 00:00.
Hour Cos	$\cos(\frac{2\pi \times hour}{24})$	Used with Sin to create unique (x,y) time coordinates.
DayOfWeek Sin	$\sin(\frac{2\pi \times day}{7})$	Makes weekdays cyclical.
DayOfWeek Cos	$\cos(\frac{2\pi \times day}{7})$	Ensures continuity between Sunday (6) and Monday (0).
Lag_24	Consumption at $t - 24$	Daily periodicity (Same hour yesterday).
Lag_168	Consumption at $t - 168$	Weekly periodicity (Same hour last week).

2.2 Static Features (Embedding)

- **Client ID**: Each client is represented by a unique integer.
- **Embedding Dimension**: Set to **16**. This represents each client as a vector in a 16-dimensional abstract space, capturing static behavioral properties.

3 Architecture Flow and Data Fusion

The proposed Multi-Client Hybrid architecture fuses time-series data with static client embeddings. Figure 1 illustrates the data flow and layer dimensions.

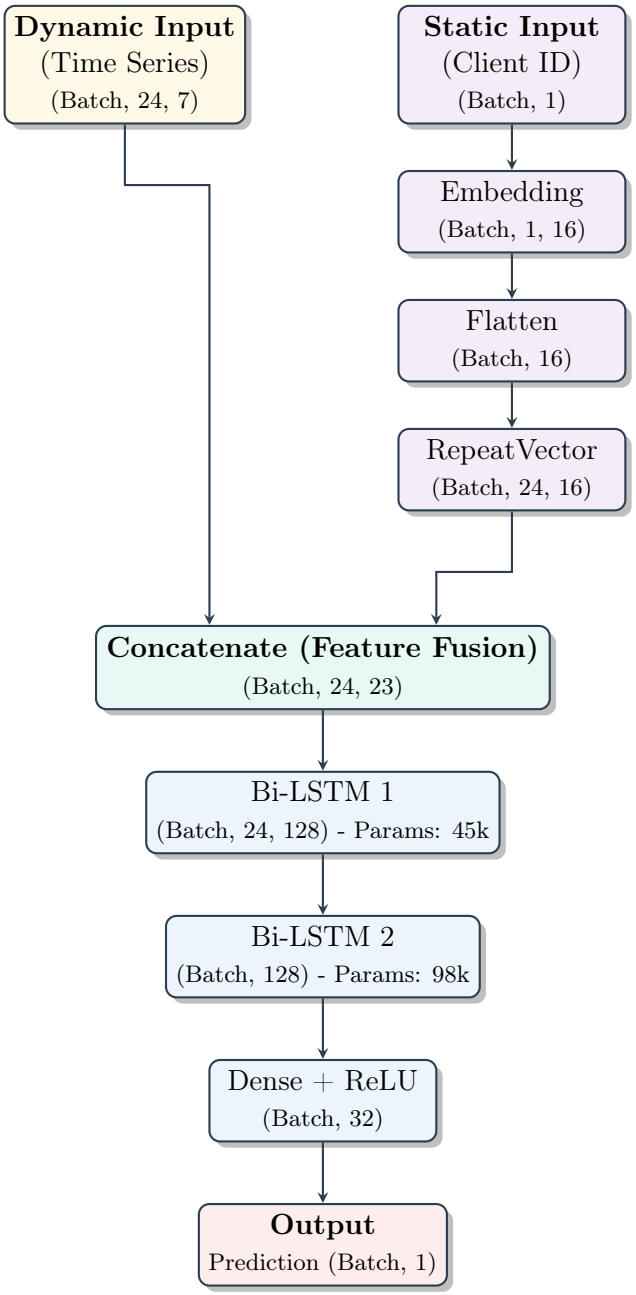


Figure 1: Multi-Client Hybrid Bi-LSTM Architecture Diagram detailing tensor shapes and parameter counts.

3.1 Implementation of Feature Fusion

The "Feature Fusion" mechanism is implemented using TensorFlow/Keras Functional API. The core logic involves duplicating the static Client ID vector across all time steps to match the temporal dimension of the dynamic input.

The following Python code snippet from our training script (`scripts/lstm_training.py`) demonstrates this process:

```
# Input 1: Time series sequence (Dynamic)
# Shape: (Batch, 24, 7)
sequence_input = tf.keras.Input(shape=(sequence_length, input_dim), name='consumption_sequence')

# Input 2: Client ID (Static)
# Shape: (Batch, 1)
client_input = tf.keras.Input(shape=(1,), name='client_id')

# 1. Learnable Embedding
# Transforms integer ID into a 16-dimensional vector representing client behavior
# Shape: (Batch, 1, 16)
client_embedding = tf.keras.layers.Embedding(
    input_dim=n_clients,
    output_dim=embedding_dim,
    name='client_embedding'
)(client_input)

# 2. Flatten & Repeat to match time steps
# Shape: (Batch, 16) -> (Batch, 24, 16)
client_embedding_flat = tf.keras.layers.Flatten()(client_embedding)
client_embedding_expanded = tf.keras.layers.RepeatVector(sequence_length)(client_embedding_flat)

# 3. Concatenate (FUSION)
# Combines 7 dynamic features with 16 static features at every time step
# Output Shape: (Batch, 24, 23)
combined = tf.keras.layers.Concatenate(axis=-1)([sequence_input, client_embedding_expanded])
```

Listing 1: Keras Implementation of Static-Dynamic Fusion

This fusion ensures that the LSTM considers "Who is this client?" (Static Embedding) at every single step of "What is happening now?" (Dynamic Sequence), effectively conditioning the forecast on the client's latent identity.The data flow is built upon the concatenation of static and dynamic data.

1. **Inputs**: The model receives dynamic data of shape `(Batch, 24, 7)` and Customer IDs of shape `(Batch, 1)`.
2. **Broadcasting**: The Client ID passes through the Embedding layer $\rightarrow$ `(1, 16)`. The `RepeatVector` layer copies this vector 24 times (once for each time step), resulting in `(Batch, 24, 16)`.
3. **Fusion (Concatenation)**:
   - Dynamic Data: 7 features
   - Static Data: 16 features
   - **Total Input Features** (x): $7 + 16 = \mathbf{23}$

Thus, the LSTM receives a vector of size 23 at every time step.

4 Mathematical Parameter Analysis

We verify the parameter counts found in the model summary using the standard LSTM parameter formula:

$$P_{lstm} = 4 \times [h \times (h + x + 1)]$$

Where h is Hidden Units, x is Input Features, and 1 is the Bias. For Bidirectional layers, $P_{bi} = 2 \times P_{lstm}$.

4.1 Layer 1: bidirectional_1

- **Input Features (x)**: 23 (7 dynamic + 16 embedding)
- **Hidden Units (h)**: 64 (per direction)

$$P = 2 \times 4 \times [64 \times (64 + 23 + 1)]$$

$$P = 8 \times [64 \times 88] = 8 \times 5632 = \mathbf{45,056}$$

Matches the model summary exactly.

4.2 Layer 2: bidirectional_2

- **Input Features (x)**: 128 (Output of Layer 1: 64 Forward + 64 Backward)
- **Hidden Units (h)**: 64

$$P = 2 \times 4 \times [64 \times (64 + 128 + 1)]$$

$$P = 8 \times [64 \times 193] = 8 \times 12352 = \mathbf{98,816}$$

Matches the model summary exactly.

5 Design Decisions

5.1 Why Embedding + RepeatVector?

Without embeddings, the model cannot distinguish between a factory and a household. The Embedding layer converts specific IDs into learnable vectors. `RepeatVector` attaches this "ID tag" to every time step (t_0 to t_{24}), allowing the LSTM toContextualize whether a sudden spike is normal for that specific client type.

5.2 Why Bidirectional?

- **Forward LSTM**: Learns patterns from past ($t - 24$) to present (t).
- **Backward LSTM**: Looks from future to past. Since the entire window is available during training, learning how t relates back to $t-5$ strengthens gradient flow, especially for periodic signals like electricity.

Table 2: Final Performance Metrics

Metric	Value
Mean Absolute Error (MAE)	42.66 kW
Root Mean Square Error (RMSE)	395.71 kW
Mean Absolute Percentage Error (MAPE)	12.71%
Weighted MAPE (WMAPE)	6.46%
Coefficient of Determination (R^2)	0.9897

6 Experimental Results

6.1 Quantitative Metrics

6.2 Comparative Analysis

Table 3 presents a comparison of our Multi-Client Hybrid Bi-LSTM model against other approaches reported in the literature.

IMPORTANT NOTE: THE STUDIES LISTED BELOW UTILIZE DIFFERENT DATASETS AND SCENARIOS. THIS IS A BROAD LITERATURE COMPARISON. DETAILED RESEARCH IS CONTINUING.

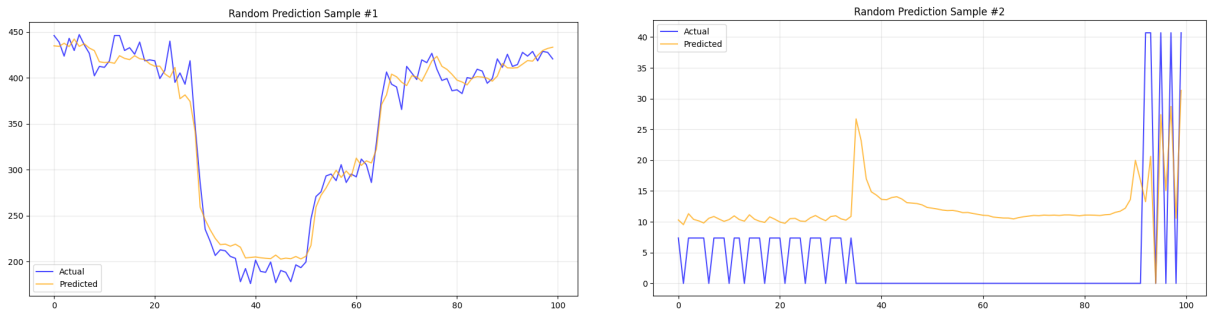
Table 3: Performance Comparison with Literature (NOTE: DIFFERENT DATASETS INCLUDED)

Study / Model	MAPE (%)	MAE (kW)	RMSE (kW)	R^2	Scenario	Source
Facebook Prophet (Yildiz et al., 2017)	20–30	–	–	–	Single	Link
LSTM (Apolo et al., 2022)	30–45	–	–	–	Single	Link
SARIMA (Sharma et al., 2023)	9.3	–	–	–	Daily	Link
Prophet (Sharma et al., 2023)	11.3	–	–	–	Daily	Link
DeepAR (Sharma et al., 2023)	11.4	–	–	–	Daily	Link
DNN + Optimization (Khaleq et al., 2025)	12.2	0.110*	–	0.719	Single	Link
Ours (Multi-Client Bi-LSTM)	**12.71**	**42.66**	**395.71**	**0.99**	**Multi**	**This Work**

* Normalized RMSE (load on 0–1 scale).

6.3 Visual Analysis

Figure 2 demonstrates the model's accuracy on unseen test data.



Sample 1 Sample 2

Figure 2: Actual vs Predicted consumption (MAPE ≈ 12.7%).

7 Conclusion

The Multi-Client Feature Fusion architecture proved highly effective. By combining 7 dynamic temporal features with 16-dimensional static client embeddings, the model achieved a high global

fit ($R^2 \approx 0.99$) and a safe error margin (MAPE 12.71%), validating the architecture design choices.