# Supplementary Materials

**Supplementary Material S1. Basic Prompt Template**

**System Message:**

"You are a time-series forecasting model responsible for generating short-term predictions based solely on historical data patterns."

**User Prompt:**

```
{
  "prompt_title": "Basic Time-Series Forecasting Prompt",
  "prompt_description": "Use historical observations to predict future values while maintaining continuity and respecting observed seasonal patterns.",
  "persona": {
    "role": "Time-Series Prediction Agent",
    "domain": "Statistical and Pattern-Based Forecasting",
    "instructions": "Do NOT use external knowledge — rely only on the provided numerical sequence and observed statistical behavior."
  },
  "dataset_metadata": {
    "sampling_frequency": "15-minute interval",
    "total_points": "{len(train_data)}",
    "value_range": "{train_min:.1f} to {train_max:.1f}",
    "statistics": {
      "mean": "{train_mean:.1f}",
      "standard_deviation": "{train_std:.1f}",
      "recent_trend_summary": "{recent_trend}"
    }
  },
  "observed_patterns": {
    "daily_cycle": "96 data points per day, trend: low at night → rise in morning → high daytime plateau → evening peak → nighttime decline",
    "weekly_cycle": "Higher load during weekdays, lower variability on weekends",
```

```
    "continuity_requirement": "Future predictions should remain consistent with historical short-term behavior"
  },
  "prediction_context": {
    "historical_windows": "{test_windows.tolist()}",
    "latest_known_value_description": "Each window ends with the most recent real measurement"
  },
  "forecasting_task": {
    "objective": "Predict the next value based on the latest sequence window",
    "method": "Infer from trend, seasonality, and recent variation, NOT external reasoning"
  },
  "prediction_constraints": {
    "valid_range": "{train_min:.1f} to {train_max:.1f}",
    "smooth_transition": "Avoid sudden spikes or unrealistic abrupt changes",
    "seasonal_consistency": "Consider known periodic behavior when generating predictions"
  },
  "output_format_requirements": {
    "response_type": "Python List of Floats",
    "precision": "Two decimal places",
    "example_format": "[1234.56, 1245.67, ...]",
    "rules": {
      "no_explanation": true,
      "no_text": true,
      "only_return_list": true
    }
  },
  "final_instruction": "Return only the final predicted list."
}
```

**Basic Prompt Code:**

prompt = f"""

You are a time-series forecasting model. Predict future values based solely on the provided historical data.

#Variable Descriptions:

#data: Complete time series dataset

#data_mean: Mean value of the dataset, reflecting the overall load level

#data_std: Standard deviation of the dataset, reflecting the volatility of load

#data_min: Minimum value of the dataset, representing the lowest load point

#data_max: Maximum value of the dataset, representing the highest load point

#train_data: Training set data (first 85% of the data)

#train_trend: Linear trend slope of the training data, positive value indicates upward trend

#train_sample: Sampling of training data, used to display data format

#test_windows: Test window list, each window contains historical data sequence for prediction

#len(test_windows): Number of future time points to be predicted

## Historical Data Characteristics

- Value range: {train_min:.1f} to {train_max:.1f}

- Mean: {train_mean:.1f}, Standard deviation: {train_std:.1f}

- Recent pattern: {recent_trend}

- Total data points: {len(train_data)} (recorded at 15-minute intervals)

## Observed Temporal Patterns

1. Daily seasonality: 96 points per day, following the pattern: nocturnal low → morning rise → daytime plateau → evening peak → nighttime decay

2. Weekly seasonality: Higher values on weekdays and lower values on weekends

3. Trend continuity: Maintain short-term smoothness

## Forecasting Task

Predict the next value based on the following historical sliding windows.

In each window, the last entry represents the most recent observed data point:

Historical windows (the last value of each list is the most recent known observation):

{test_windows.tolist()}

## Prediction Constraints

1. Prediction must stay within the valid range ({train_min:.1f} ~ {train_max:.1f})

2. Values must vary smoothly with no abrupt jumps

3. Respect periodic behavior based on the temporal index (daily/weekly seasonality)

## Output Format

Return only a Python list, e.g.: [1234.56, 1245.67, ...]

Begin forecasting now:

"""

**Supplementary Material S2. CoT Prompt Template**

**System Message:**

"You are a specialized expert in electric power load forecasting. Analyze the provided historical time-series load data and generate short-term predictive outputs."

**User Prompt:**

```
{
  "prompt_title": "Short-Term Electric Load Forecasting – Structured Instruction Protocol",
  "prompt_description": "Using the given historical electricity load measurements, seasonal patterns, and statistical characteristics, generate predictions for the specified future time steps.",
  "persona": {
    "role": "Electric Power System Load Forecasting Specialist",
    "domain": "Energy Engineering / Time-Series Forecasting",
    "instructions": "Your forecasting must be based strictly on the provided numeric context, seasonal behavior, constraints, and dataset structure."
  },
  "dataset_metadata": {
    "data_type": "Electrical load measurement",
    "sampling_frequency": "15-minute interval",
    "total_points": "<len(data)>",
    "time_coverage": "<len(data) / 96> days (approx.)",
    "train_test_split": {
      "training_percentage": "85%",
      "training_points": "<len(train_data)>",
      "testing_percentage": "15%",
      "testing_target": "Future prediction"
    },
    "unit": "MW"
  },
  "statistical_summary": {
    "global_statistics": {
```

```
    "mean": "<data_mean> MW",
    "std_dev": "<data_std> MW",
    "min_value": "<data_min> MW",
    "max_value": "<data_max> MW",
    "range": "<data_max - data_min> MW"
  },
  "training_trend_characteristics": {
    "linear_trend_slope": "<train_trend>",
    "normalized_rate_of_change": "<(train_data[-1] - train_data[0]) / len(train_data)> MW per sample"
  }
},
"known_seasonality_patterns": {
  "daily_cycle": {
    "period": "96 points (24h × 4 intervals/h)",
    "behavior": "Night low → Morning rise → Midday plateau → Evening peak → Night decline"
  },
  "weekly_cycle": {
    "period": "672 points (96 × 7)",
    "patterns": {
      "weekday": "Higher and stable",
      "weekend": "Lower and variable"
    }
  },
  "annual_behavior": {
    "summer": "Increased AC-driven load",
    "winter": "Increased heating-driven load",
    "transition_seasons": "Moderate and more stable"
  }
},
```

```
"training_data_overview": {
  "sample_preview": "<train_sample>",
  "interpretation": {
    "trend_summary": "<describe_trend(train_data)>",
    "cyclic_behavior": "Clear daily patterns and weak weekly periodicity",
    "volatility_profile": "<describe_volatility(train_data)>",
    "anomaly_assessment": "<describe_anomalies(train_data)>"
  }
},
"forecasting_task": {
  "objective": "Predict the next <len(test_windows)> future values.",
  "input_structure": "Each prediction window contains the latest historical points.",

  "constraints": {
    "continuity": "Sequential values should exhibit smooth temporal transitions.",
    "bounded_output": "Forecast must fall within historical value range (<data_min> to <data_max> MW).",
    "seasonality_preservation": "Forecast must reflect daily and weekly rhythmic structure.",
    "trend_consistency": "Short-term directional behavior must remain coherent with observed historical trends."
  }
},
"prediction_input_windows": "<test_windows>",
"output_format_requirements": {
  "response_type": "Python List of Floats",
  "numeric_precision": "2 decimal places",
  "format_example": "[value1, value2, value3, ...]",
  "rules": {
    "no_explanations": true,
    "no narrative text": true,
```

"only the final predicted list may be output"

}

},

"final_instruction":

"Return ONLY the final prediction list, formatted according to the output specification."

}

**CoT Prompt Code:**

prompt = f"""You are a professional time-series forecasting model. Your objective is to predict future data points based on the provided historical time-series data.

#Variable Descriptions:

#data: Complete time series dataset

#data_mean: Mean value of the dataset, reflecting the overall load level

#data_std: Standard deviation of the dataset, reflecting the volatility of load

#data_min: Minimum value of the dataset, representing the lowest load point

#data_max: Maximum value of the dataset, representing the highest load point

#train_data: Training set data (first 85% of the data)

#train_trend: Linear trend slope of the training data, positive value indicates upward trend

#train_sample: Sampling of training data, used to display data format

#test_windows: Test window list, each window contains historical data sequence for prediction

#len(test_windows): Number of future time points to be predicted

## Data Background

1. Data Type: Electrical load demand (Unit: Megawatts, MW)

2. Sampling Frequency: 15-minute interval

3. Dataset Size: A total of {len(data)} data points, equivalent to approximately {len(data) / 96:.1f} days

4. Dataset Partitioning:

Training set: First 85% of the data ({len(train_data)} points)

Testing set: Remaining 15% of the data (to be forecasted)

## Statistical Characteristics

Global Statistics (Entire Dataset):

Mean: {data_mean:.2f} MW

Standard deviation: {data_std:.2f} MW

Minimum value: {data_min:.2f} MW

Maximum value: {data_max:.2f} MW

Value range: {data_max - data_min:.2f} MW

Training Data Trend:

Linear trend slope: {train_trend:.6f} (positive indicates upward trend, negative indicates downward trend)

Rate of change: {(train_data[-1] - train_data[0]) / len(train_data):.4f} MW per sample

## Known Seasonality Patterns

1. Daily Seasonality: 96 points per day (24 hours × 4 intervals/hour)

   Typical daily load pattern: nighttime low → morning ramp-up → midday plateau → evening peak → nighttime decline

2. Weekly Seasonality: 672 points per week (96 points/day × 7 days)

   Weekdays (Mon–Fri): Higher and more stable demand

   Weekends (Sat–Sun): Lower and less structured demand

3. Long-Term Seasonality:

   Summer (June–August): Increased load due to cooling systems

   Winter (December–February): Increased load due to heating systems

   Transitional seasons (spring/autumn): Moderately lower and more stable load

## Training Data Overview (85% of total dataset)

Below are sampled records from the training set ({len(train_sample)} points in total):

{train_sample.tolist()}

## Visual Interpretation of Training Data

1. Overall Trend: {describe_trend(train_data)}

2. Seasonal Pattern: Clear daily periodicity and weaker weekly periodicity

3. Volatility Characteristics: {describe_volatility(train_data)}

4. Anomaly Behavior: {describe_anomalies(train_data)}

## Forecasting Task

You are required to predict the next {len(test_windows)} future time points.

## Prediction Procedure

For each forecasting step, you will receive:

A historical sliding window containing the most recent observations

You must infer the next value based on previously observed temporal structure

## Prediction Principles

1. Continuity Constraint: Consecutive predictions should be smooth and follow natural temporal transitions

2. Range Constraint: Forecasted values must lie within: ({data_min:.2f} ~ {data_max:.2f} MW)

3. Seasonality Preservation: Daily and weekly periodic structures should be maintained

4. Trend Retention: Short-term directional behavior must align with the historical trend

## Forecast Input

You will receive {len(test_windows)} historical windows.

Each window ends with the most recent known value, and you must infer the next unseen value.

## Output Format Requirements

Please strictly follow the output rules:

1. Return a single Python list containing all predicted values

2. Format: [prediction_1, prediction_2, prediction_3, ...]

3. Each value must be rounded to two decimal places

4. Do not include any additional explanation or commentary

## Prediction Windows

Below are the historical sliding windows requiring prediction (the final entry in each list is the most recent known value; predict the next value):

{test_windows.tolist()}

Based on the above dataset characteristics and analysis, generate the predicted values and return the final output list.

"""

return prompt

**Supplementary Material S3. Hyperparameter Settings for All Comparative Models**

Table S1 shows complete hyperparameter configurations and inference settings for statistical, deep learning, hybrid, and LLM-based models used in this study.

**Table S1. Hyperparameter Settings for All Comparative Models**

| Model Category | Model | Component / Setting | Hyperparameter | Value | Note / Rationale |
|---|---|---|---|---|---|
| Statistical Model | ARIMA | Model Order | (p, d, q) | (2, 1, 1) | Selected via AIC minimization. |
| Deep Learning Models | BiLSTM | Architecture | Hidden Size | 64 | Balanced capacity and overfitting risk. |
| | | | Number of Layers | 2 | |
| | | Training | Learning Rate | 0.001 | Adam optimizer default. |
| | | | Dropout Rate | 0.2 | For regularization. |
| | | | Batch Size | 32 | |
| | | | Epochs | 100 | With early stopping (patience=10). |
| | TimesNet | Architecture | Model Dimension | 64 | Follows the original paper's configuration for similar-scale data. |
| | | Training | Learning Rate | 0.001 | |
| | | | Batch Size | 32 | |
| | MICN | Architecture | Hidden Size | 64 | Balanced representation capacity. |
| | | | Number of Layers | 2 | Default setting in the original paper. |
| | | Training | Learning Rate | 0.001 | Adam optimizer default. |
| | | | Batch Size | 32 | Consistent with other deep learning models. |
| | | | Epochs | 100 | With early stopping (patience=10). |
| Hybrid Model | VMD-BiLSTM | VMD Decomposition | Number of Modes (K) | 8 | Determined by observing central frequencies. |
| | | | Penalty ($\alpha$) | 2000 | Default value for balancing bandwidth. |
| | | | Tolerance ($\varepsilon$) | 1e-7 | Convergence criterion. |

| Model Category | Model | Component / Setting | Hyperparameter | Value | Note / Rationale |
|---|---|---|---|---|---|
| | | BiLSTM Component | Configurations | Same as BiLSTM above | |
| LLMs: Prompt & Inference Settings | All LLMs | Prompt Framework | CoT Enabled | Yes/No | Ablation study: Base (No) vs. CoT (Yes). |
| LLMs: API-based (Zero-Shot) | GPT-4 | Inference (General) | Self-Consistency | Enabled for CoT prompts | |
| | | | Self-Consistency Samples (n) | 5 | Aggregates multiple reasoning paths. |
| | | | Max Tokens to Generate | 512 | Sufficient for a list of 16 float values. |
| | | | Output Format Constraint | JSON list | Enforced via system prompt. |
| | | API Inference | Model Version | gpt-4-turbo | As of the experiment date. |
| | | | Temperature | 0.1 | Low temperature for deterministic output. |
| | | | Token Budget (Context Window) | 8192 | Max context supported. |
| | Claude-3 | API Inference | Model Version | claude-3-sonnet | |
| | | | Temperature | 0.1 | |
| | | | Token Budget (Context Window) | 8192 | |
| | Gemini-Pro | API Inference | Model Version | gemini-1.5-pro | |
| | | | Temperature | 0.1 | |
| | | | Token Budget (Context Window) | 8192 | |
| LLMs: Open-Source (Local) | Llama-3-8B | Local Inference | Model Variant | Meta-Llama-3-8B-Instruct | |
| | | | Temperature | 0.1 | |
| | | | Token Budget (Context Window) | 1024 | Limited by GPU memory. |
| | DeepSeek-R1 | Local Inference | Model Variant | DeepSeek-R1 | Base model for Ele-LLM. |

| Model Category | Model | Component / Setting | Hyperparameter | Value | Note / Rationale |
|---|---|---|---|---|---|
| | Qwen2.5-7B | Local Inference | Temperature | 0.1 (0.3 for Base*) | Base model used Temp=0.3 for less greedy decoding. |
| | | | Token Budget (Context Window) | 1024 | |
| | | | Model Variant | Qwen2.5-7B-Instruct | |
| | | | Temperature | 0.1 | |
| | | | Token Budget (Context Window) | 1024 | |
| Fine-tuned LLM | Ele-LLM | Fine-tuning (LoRA) | Base Model | DeepSeek-R1 | |
| | | | LoRA Rank (r) | 16 | |
| | | | LoRA Alpha ($\alpha$) | 32 | |
| | | | Dropout | 0.05 | |
| | | | Learning Rate | 2e-4 | |
| | | Inference | Settings | Same as DeepSeek-R1 | Inherits base model's inference config. |