SDM 274 AI and Machine Learning Final Project Report

——电力负荷预测对比研究

姓名：纪浩正　学号：12311405

课程：2025年秋季学期《SDM 274 AI and Machine Learning》　教师：林志赟

Abstract: This study investigates short-term electricity consumption forecasting using advanced deep learning models, including Convolutional Neural Networks (CNN), Long Short-Term Memory networks (LSTM), Transformer networks, and hybrid CNN-LSTM architectures. The dataset consists of household power consumption records, which were preprocessed, resampled hourly, and normalized. Input sequences were constructed with a look-back window of 168 hours and a forecast horizon of 168 hours.

The models were trained with the Mean Squared Error (MSE) loss function and the Adam optimizer, with experiments conducted to evaluate the impact of learning rate adjustments and early stopping on model performance. Performance metrics include test loss, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). In addition, exploratory analyses were performed, including backtesting, to validate predictive accuracy on historical data.

Experimental results indicate that all models successfully capture the temporal patterns of electricity consumption. CNN models converge faster but may slightly underperform in long-horizon predictions, while LSTM and Transformer models provide more stable predictions over longer horizons. The hybrid CNN-LSTM architecture achieves competitive performance, combining local feature extraction with temporal sequence modeling. Backtesting and daily aggregation confirm that the models can reliably forecast both hourly and daily consumption.

Overall, this work demonstrates the feasibility of deep learning-based forecasting for electricity consumption, highlights the influence of hyperparameter tuning, and provides a framework for combining

different neural network architectures to improve predictive performance.

1. Data Description and Preprocessing

2.1. Dataset Description

This study uses the UCI Household Electric Power Consumption Dataset, which records electricity usage of a single household with a one-minute temporal resolution. The dataset spans from December 2006 to November 2010 and contains 2,075,259 observations, making it suitable for short-term load forecasting.

```
83  ---------------------------------------------┼-------------
84  Data info:
85  -------------------------------------------------------------
86  <class 'pandas.core.frame.DataFrame'>
87  DatetimeIndex: 2075259 entries, 2006-12-16 17:24:00 to 2010-11-26 21:02:00
88  Data columns (total 7 columns):
89   #   Column                 Dtype
90  ---  ------                 -----
91   0   Global_active_power    float64
92   1   Global_reactive_power  float64
93   2   Voltage                float64
94   3   Global_intensity       float64
95   4   Sub_metering_1         float64
96   5   Sub_metering_2         float64
97   6   Sub_metering_3         float64
98  dtypes: float64(7)
99  memory usage: 126.7 MB
```

Each record includes several electrical measurements, such as Global_active_power, Global_reactive_power, Voltage, Global_intensity, and three sub-metering variables representing energy consumption of different household areas. In this study, Global_active_power is selected as the target variable, while the remaining variables are used as input features.

A small proportion of missing values (approximately 1.25%) exists due to data acquisition issues. These missing values are addressed during the data cleaning process described in the next

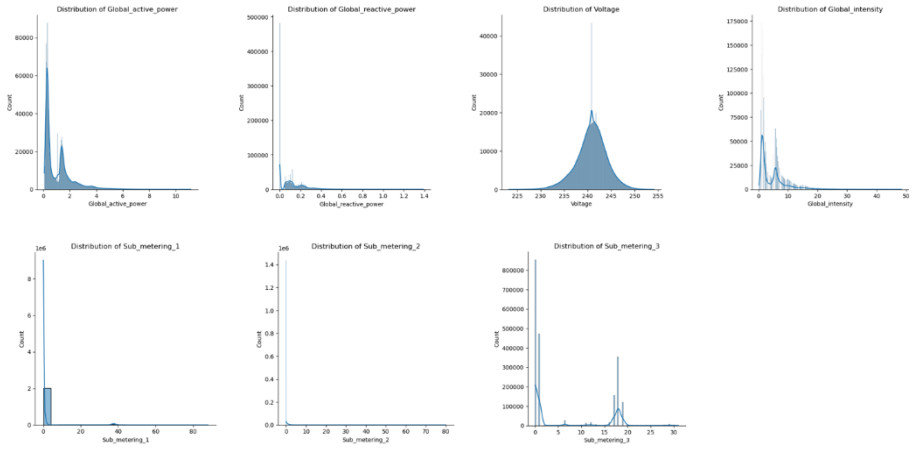
section.

```
============================================================
Data Quality Check
============================================================

Missing values count:
                 Column  Missing_Count  Missing_Percent
0    Global_active_power          25979             1.25
1  Global_reactive_power          25979             1.25
2                Voltage          25979             1.25
3       Global_intensity          25979             1.25
4         Sub_metering_1          25979             1.25
5         Sub_metering_2          25979             1.25
6         Sub_metering_3          25979             1.25

Total missing values: 181,853
Percentage of total data: 1.25%

Duplicate rows: 168560

============================================================
Data loading completed!
============================================================
```

```
------------------------------------------------------------
Basic statistics:
------------------------------------------------------------
       Global_active_power  Global_reactive_power       Voltage  \
count         2.049280e+06           2.049280e+06  2.049280e+06
mean          1.091615e+00           1.237145e-01  2.408399e+02
std           1.057294e+00           1.127220e-01  3.239987e+00
min           7.600000e-02           0.000000e+00  2.232000e+02
25%           3.080000e-01           4.800000e-02  2.389900e+02
50%           6.020000e-01           1.000000e-01  2.410100e+02
75%           1.528000e+00           1.940000e-01  2.428900e+02
max           1.112200e+01           1.390000e+00  2.541500e+02

       Global_intensity  Sub_metering_1  Sub_metering_2  Sub_metering_3
count      2.049280e+06    2.049280e+06    2.049280e+06    2.049280e+06
mean       4.627759e+00    1.121923e+00    1.298520e+00    6.458447e+00
std        4.444396e+00    6.153031e+00    5.822026e+00    8.437154e+00
min        2.000000e-01    0.000000e+00    0.000000e+00    0.000000e+00
25%        1.400000e+00    0.000000e+00    0.000000e+00    0.000000e+00
50%        2.600000e+00    0.000000e+00    0.000000e+00    1.000000e+00
75%        6.400000e+00    0.000000e+00    1.000000e+00    1.700000e+01
max        4.840000e+01    8.800000e+01    8.000000e+01    3.100000e+01
```

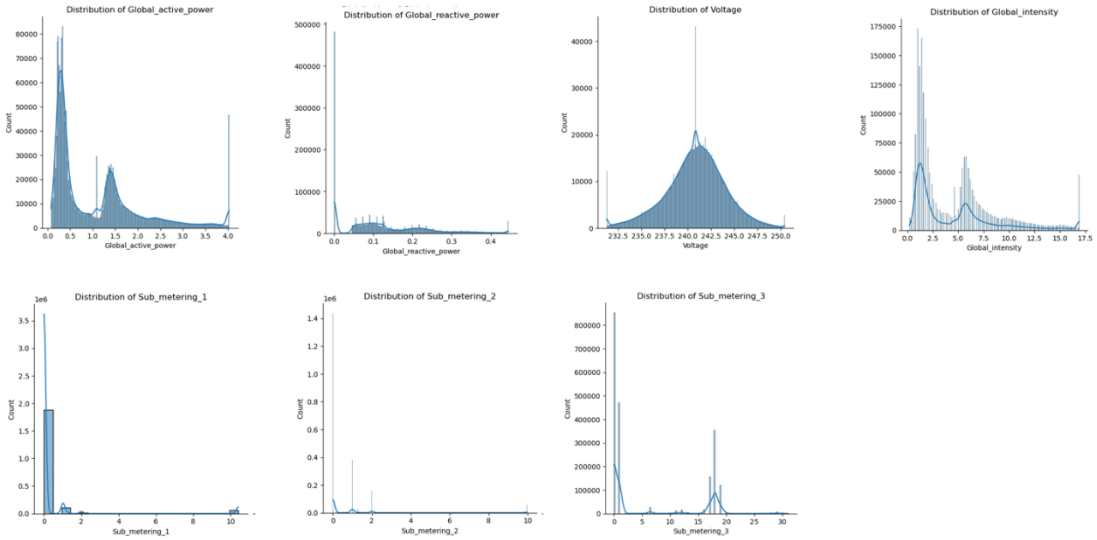
2.2. Data Cleaning

The raw dataset contains missing values and abnormal measurements caused by sensor errors. To ensure data quality, missing values were first identified across all numerical features. Since the proportion of missing data is relatively small, missing entries in the main electrical variables were filled using the mean value of each feature, which preserves the overall statistical properties of the time series. After imputation, no missing values remained in the dataset. The figure below shows the data distribution before clipping.



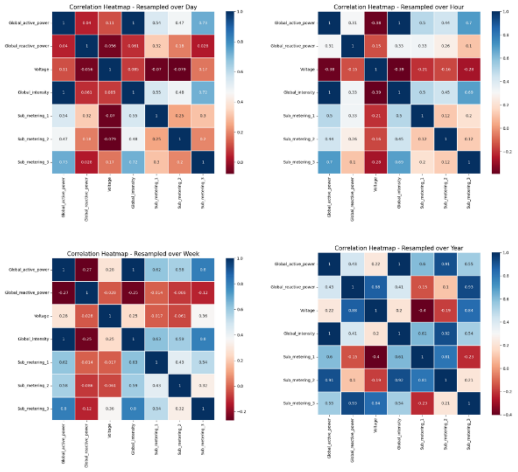
Outliers were then handled using the three-sigma (3σ) rule. For each feature, values outside the range $[\boldsymbol{\mu - 3\sigma, \mu + 3\sigma}]$ were considered abnormal. Instead of removing these observations, extreme

values were clipped to the corresponding boundaries, reducing the influence of outliers while maintaining the continuity and length of the time series. This approach is well suited for time-series forecasting tasks and provides stable inputs for subsequent deep learning models. The figure below shows the data distribution after clipping.



2.3. Correlation Analysis

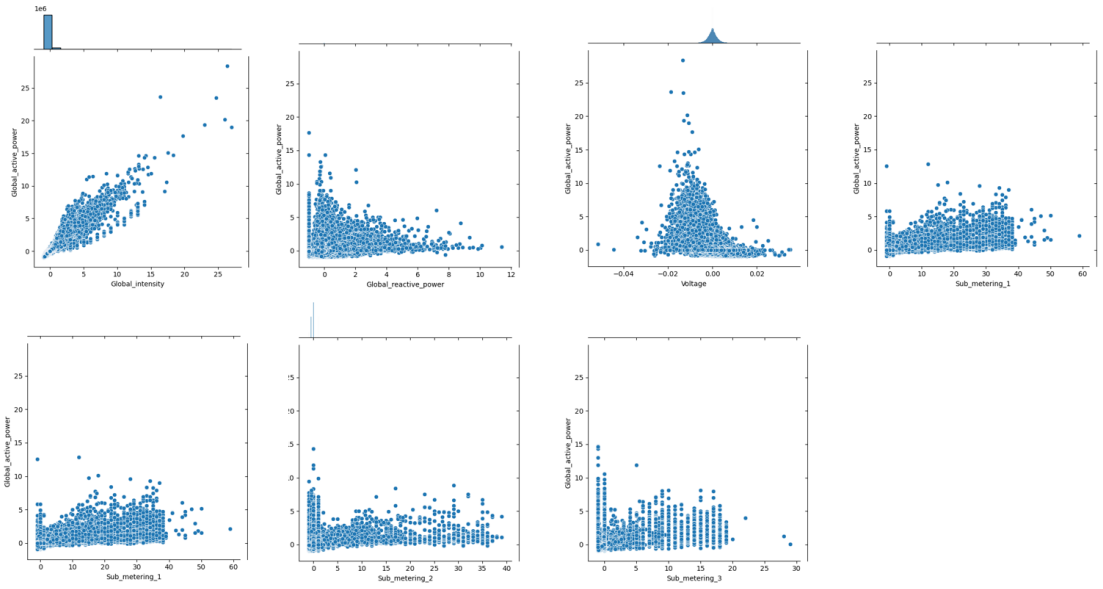
Correlation analysis was conducted to investigate the relationships between Global_active_power and other variables at different temporal resolutions. The data were resampled at hourly, weekly, monthly, and yearly scales, and the correlation



structures were found to vary with the resampling interval.

This indicates that temporal aggregation influences the statistical relationships among variables.

Despite these differences, Global_active_power consistently shows the strongest correlation with Global_intensity and the sub-metering variables, particularly Sub_metering_1, Sub_metering_2, and Sub_metering_3, across all resampling scales. These variables directly reflect household electricity usage and therefore exhibit strong linear dependence with the total active power consumption.



The correlation patterns are visualized using heatmaps at different time scales, as well as joint distribution plots generated by sns.jointplot, which further illustrate the strong dependence between Global_active_power and the most relevant features. These observations support the selection of Global_intensity and sub-metering variables as important input features for the forecasting models.

2.4. Data Normalization and Feature Engineering

To transform the raw data into a supervised learning format suitable for deep learning models, a sliding window strategy

was adopted. After resampling, all features were first normalized to the range $[0,1]$using Min-Max normalization, ensuring numerical stability during model training.

The normalized multivariate time series was then converted into a supervised learning dataset by introducing lagged input features and corresponding prediction targets. For each time step, observations from the previous time step $t-1$ were used as input, while the value at the current time step t was used as the prediction target. This transformation preserves temporal dependencies and enables the models to learn time-dependent patterns.

After the time-series conversion, only the features at time $t-1$ and the target variable at time t were retained, while unnecessary future-step variables were removed. This feature engineering process results in a structured input-output representation that is compatible with CNN, LSTM, and Transformer models.

2.5. Dataset Splitting

After feature engineering, the processed time-series data were divided into training, validation, and test sets following a chronological order to avoid information leakage. The dataset was split using a ratio of 70% for training, 15% for validation, and 15% for testing.

The input features consist of lagged observations at time $t-1$, while the target variable corresponds to the value at time t. To meet the input requirements of deep learning models, the feature data were reshaped into a three-dimensional format (samples, time steps, features), where the time step was set to one.

All datasets were then converted into PyTorch tensors, and mini-batch data loaders were constructed with a batch size of 70. Data shuffling was disabled to preserve the temporal order of the time series during training and validation.

2. Model Implementation

3.1. Convolution Neural Network (CNN)

A one-dimensional Convolutional Neural Network (CNN) was implemented to model local temporal patterns in the time-series data. The input consists of a single time step with multiple feature channels, which is suitable for one-step-ahead forecasting.

The CNN architecture includes a 1D convolutional layer with 64 filters and a kernel size of 3, followed by a ReLU activation function to introduce nonlinearity. A dropout layer with a rate of 0.2 is applied to reduce overfitting. The extracted features are then flattened and passed to a fully connected layer to generate the final prediction.

The model was trained using the mean squared error (MSE) loss function and optimized with the Adam optimizer. Training stability was further improved using early stopping and a learning rate scheduler, which are described in detail in the following section.

3.2. Long Short-Term Memory Network (LSTM)

A Long Short-Term Memory (LSTM) network was implemented to capture long-range temporal dependencies in the time-series data. The input at each time step consists of the lagged features, and the model predicts the target variable at the current time step.

The architecture consists of a single LSTM layer with 100

hidden units, followed by a dropout layer with a rate of 0.2 to prevent overfitting. Only the output corresponding to the last time step is passed to a fully connected layer to produce the final prediction. This design allows the LSTM to selectively retain relevant information from previous time steps while discarding less important signals.

The model was trained using the mean squared error (MSE) loss function and optimized with the Adam optimizer. Early stopping and a learning rate scheduler were applied to stabilize training and prevent overfitting, following the same procedure as in the CNN model.

3.3. Transformer Model

A Transformer-based model was implemented to capture both short-term and long-range dependencies in the time-series data using a self-attention mechanism. Each input time step, consisting of the lagged features, is first projected into a higher-dimensional hidden space.

The core architecture consists of a Transformer encoder with 2 layers and 8 attention heads. A dropout layer with a rate of 0.2 is applied to prevent overfitting. Similar to the LSTM model, only the output corresponding to the last time step is fed into a fully connected layer to produce the final prediction. This allows the model to integrate information across the entire sequence while focusing on the most relevant temporal patterns.

The model was trained using the mean squared error (MSE) loss function and optimized with the Adam optimizer. Early stopping and a learning rate scheduler were employed to stabilize training and prevent overfitting, following the same procedure

as in the CNN and LSTM models.

3. Model Training and Hyperparameter Tuning

4.1. Loss Function Design

All three models—CNN, LSTM, and Transformer—were trained using the Mean Squared Error (MSE) loss function, which is commonly applied in regression and time-series forecasting tasks. The MSE loss measures the average squared difference between the predicted and actual values, making it suitable for evaluating short-term load prediction performance.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)$$

4.2. Optimizer Configuration

For optimization, the Adam optimizer was employed across all models. Adam combines the advantages of adaptive learning rates and momentum, providing stable and efficient convergence for deep neural networks.

The Adam optimizer updates model parameters $\boldsymbol{\theta_t}$ at each iteration $\boldsymbol{t}$ as:

$$\boldsymbol{m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta L(\theta_{t-1})}$$

$$\boldsymbol{v_t = \beta_2 v_{t-1} + (1-\beta_2)\left(\nabla_\theta L(\theta_{t-1})\right)^2}$$

$$\boldsymbol{\hat{m}_t = \frac{m_t}{1-\beta_1^t}}$$

$$\boldsymbol{\hat{v}_t = \frac{v_t}{1-\beta_2^t}}$$

$$\boldsymbol{\theta_t = \theta_{t-1} - \eta\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}}$$

Here:

- $\boldsymbol{\nabla_\theta L(\theta_{t-1})}$ is the gradient of the loss with respect to parameters,
- $\boldsymbol{m_t}$ and $\boldsymbol{v_t}$ are the first and second moment estimates,

- β_1, β_2 are exponential decay rates,
- η is the learning rate, and ϵ is a small constant for numerical stability.

This update rule combines momentum and adaptive learning rates, allowing stable and efficient convergence for all three models.

4.3. Early Stopping Strategy

A step-based scheduler was used to reduce the learning rate by a factor of 0.5 every 10 epochs, allowing the optimizer to fine-tune weights as training progresses.

$$\eta_t = \eta_0 \cdot \gamma^{[t/s]}$$

4.4. Learning Rate Scheduling

Training was monitored on the validation dataset, and stopped automatically if the validation loss did not improve for 10 consecutive epochs. This prevents excessive overfitting while preserving model generalization.

All models were trained on mini-batches with a batch size of 70, and the input features were reshaped into the format (samples, time steps, features) compatible with each model architecture.

$$L_{val}(t) \geq \min\left(L_{val}(t-P) > \cdots, L_{val}(t-1)\right)$$

4. Results Analysis and Innovative Exploration

All the models were evaluated under four training strategies:

A. No early stopping, fixed learning rate

B. No early stopping, learning rate scheduler

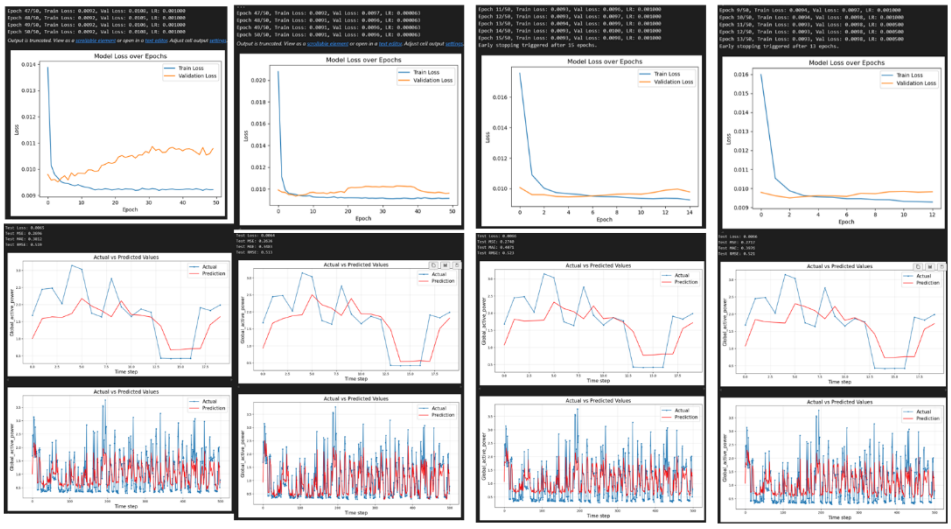
C. Early stopping, fixed learning rate

D. Early stopping, learning rate scheduler

5.1. CNN Performance Evaluation

The corresponding test performance is summarized below:

Strategy	Loss	MSE	MAE	RMSE	Time
A	0.0065	0.2696	0.3812	0.519	1m 3.2s
B	0.0064	0.2636	0.3483	0.513	1m 6s
C	0.0066	0.2740	0.4071	0.523	17.2s
D	0.0066	0.2717	0.3976	0.521	15.7s



It can be observed that without early stopping and with a fixed learning rate, the validation loss initially decreases but eventually increases, indicating overfitting. In contrast, the other strategies effectively suppress the rise of validation loss, demonstrating the stabilizing effects of early stopping and learning rate scheduling during training.

5.2. LSTM Performance Evaluation

During training, models without early stopping showed an interesting behavior between epochs 10-20, where the loss initially increased and then decreased. Specifically, for Strategy C (early stopping, fixed learning rate), the training stopped at epoch 13, while for Strategy D (early stopping, learning rate scheduler), training stopped at epoch 15.

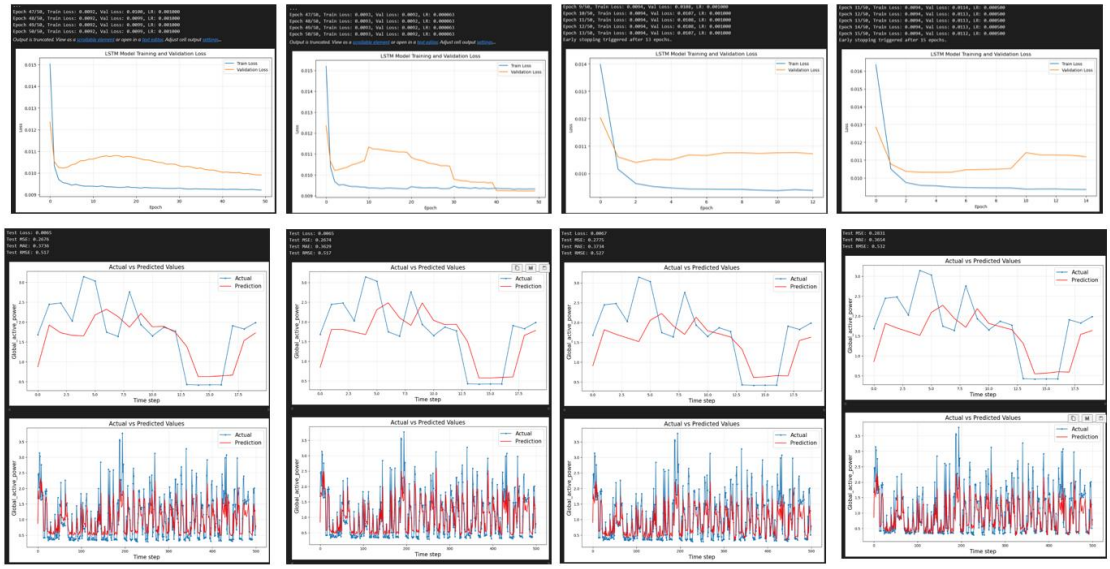
The corresponding test performance for each strategy is

summarized below:

Strategy	Loss	MSE	MAE	RMSE	Time
A	0.0065	0.2676	0.3736	0.517	1m 27.4s
B	0.0065	0.2674	0.3629	0.517	1m 31.2s
C	0.0067	0.2775	0.3734	0.527	24.6s
D	0.0068	0.2831	0.3654	0.532	24.8s

It can be observed that, in general, early stopping helped prevent further increases in the validation loss, thus improving model generalization. The model trained with learning rate scheduling performed slightly better, especially in terms of Mean Absolute Error (MAE), though the differences in MSE and RMSE are minimal across the different strategies. These results indicate that early stopping combined with a learning rate scheduler provides the best trade-off between training duration and model performance.

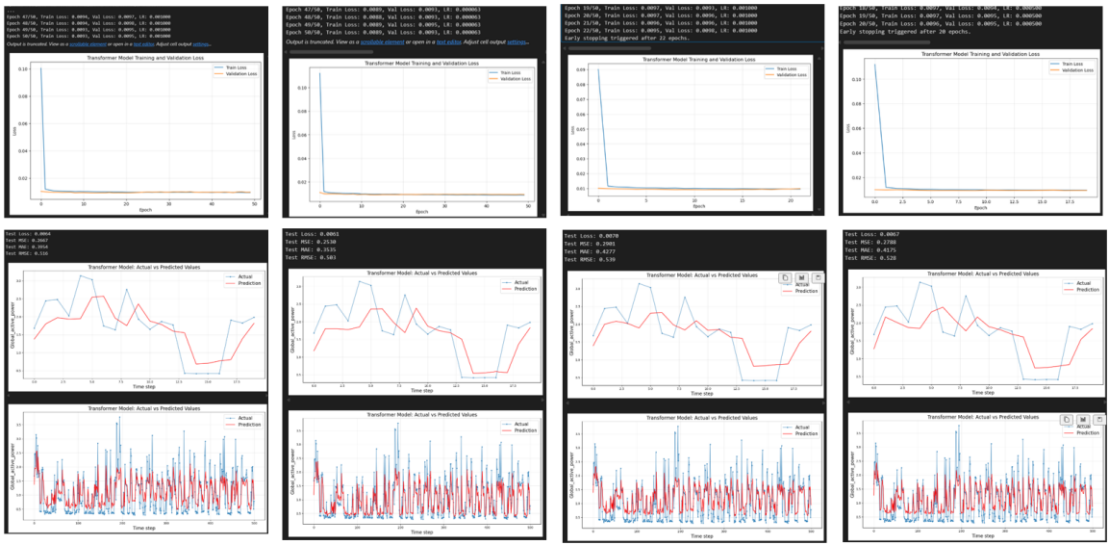
Additionally, the training times for each strategy were recorded, showing that the early stopping strategies, especially when combined with learning rate scheduling, significantly reduced the training time.



5.3. Transformer Performance Evaluation

The test performance and training time for each strategy are summarized below:

Strategy	Loss	MSE	MAE	RMSE	Time
A	0.0064	0.2667	0.3954	0.516	6m 25s
B	0.0061	0.2530	0.3535	0.503	6m 28.2s
C	0.0070	0.2901	0.4277	0.539	2m 55.3s
D	0.0067	0.2788	0.4175	0.528	2m 39.7s



During training, the Transformer's loss curve exhibited a rapid initial decrease followed by a long period of stability. After the initial drop, the loss remained relatively flat, with only minor fluctuations, indicating that the model converged steadily without large oscillations.

Although the training time of the Transformer is significantly longer than CNN and LSTM, its predictive performance is comparable, especially when considering MSE and RMSE. Early stopping effectively reduced training time while maintaining similar performance levels.

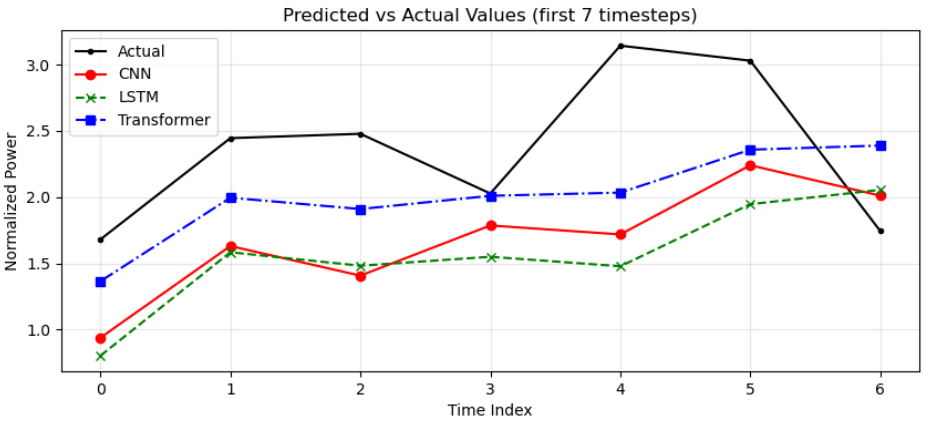
Overall, the results suggest that while Transformer models require more computational resources, they provide stable convergence and robust performance, making them suitable for

time-series forecasting tasks with sufficient training data.

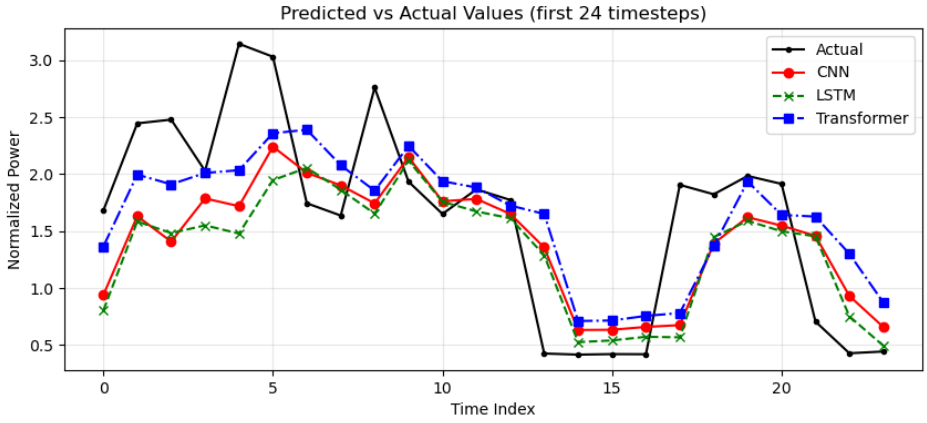
5.4. Comparison of three model

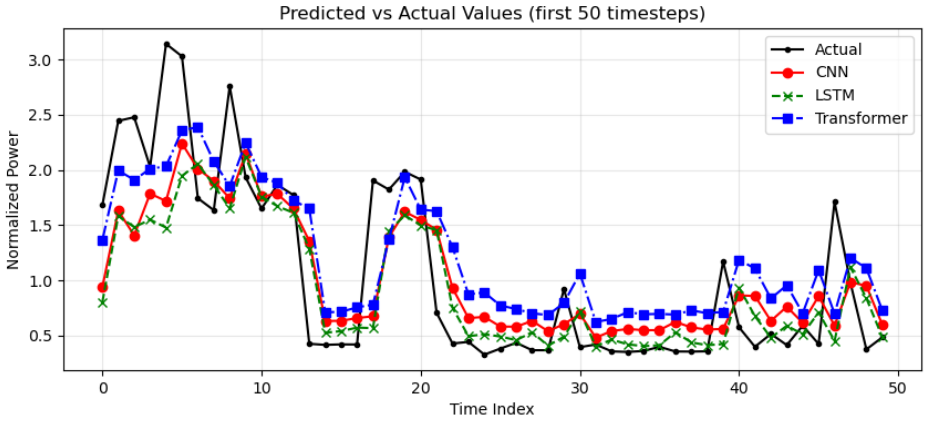
To evaluate the short-term and long-term forecasting capabilities of the three models, we compared the predicted values of CNN, LSTM, and Transformer against the actual power consumption after different time horizons. Three representative horizons were selected: 7 steps (~7 hours), 24 steps (~24 hours), and 500 steps (~500 hours).

- Short-term (7 steps): In the left-top plot, the Transformer predictions closely follow the actual values, capturing both peaks and troughs accurately. CNN predictions deviate more from the ground truth, while LSTM shows moderate accuracy.

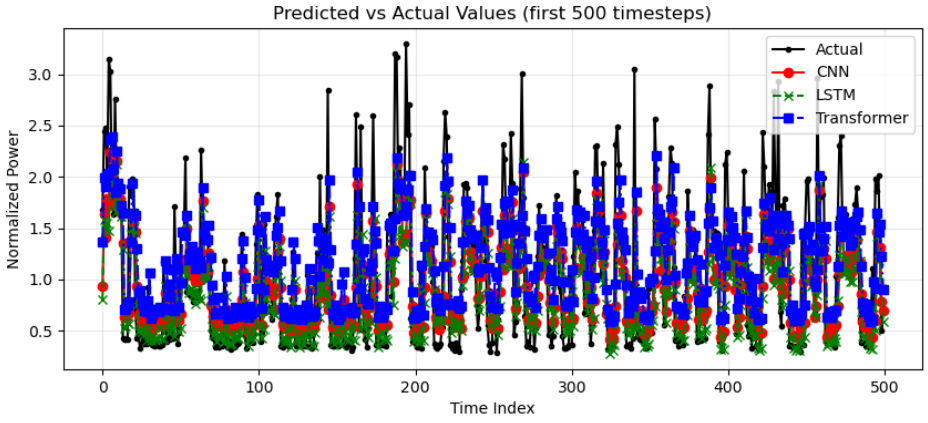


- Medium-term (24 & 50 steps): Across 24 or 50 hours, all three models capture the general trend of the power consumption. Minor differences exist in amplitude and phase, but the predicted curves are largely aligned with the actual trajectory.





- Long-term (500 steps): Over longer horizons, the trend consistency among all models remains, with CNN, LSTM, and Transformer all able to approximate the overall pattern of consumption. Transformer maintains slightly higher fidelity in reproducing local variations.



Overall, these results indicate that while Transformer excels

in short-term precision, all models are capable of capturing the general trend over medium and long-term horizons. The advantage of Transformer is particularly notable when fine-grained short-term prediction is required.

5.5. Exploratory Hyperparameter Tuning: Learning Rate

To investigate the effect of different learning rates on the CNN model performance, we conducted an exploratory experiment with four learning rates: 0.0005, 0.001, 0.005, and 0.01. Early stopping with a patience of 10 epochs was applied to prevent overfitting. The training results are summarized below:

```
=== Training CNN with learning rate = 0.0005 ===
Early stopping triggered at epoch 16

=== Training CNN with learning rate = 0.001 ===
Early stopping triggered at epoch 16

=== Training CNN with learning rate = 0.005 ===
Early stopping triggered at epoch 13

=== Training CNN with learning rate = 0.01 ===
Early stopping triggered at epoch 12
LR=0.0005: Time=17.98s, Epoch=16, Loss=0.0148, MSE=0.2503, MAE=0.3684, RMSE=0.500
LR=0.001: Time=17.53s, Epoch=16, Loss=0.0148, MSE=0.2518, MAE=0.3843, RMSE=0.502
LR=0.005: Time=14.70s, Epoch=13, Loss=0.0155, MSE=0.2639, MAE=0.3857, RMSE=0.514
LR=0.01: Time=13.67s, Epoch=12, Loss=0.0163, MSE=0.2764, MAE=0.4014, RMSE=0.526
```

Observations:

Early stopping epochs: Higher learning rates converge faster, leading to earlier stopping (epoch 12 – 16).

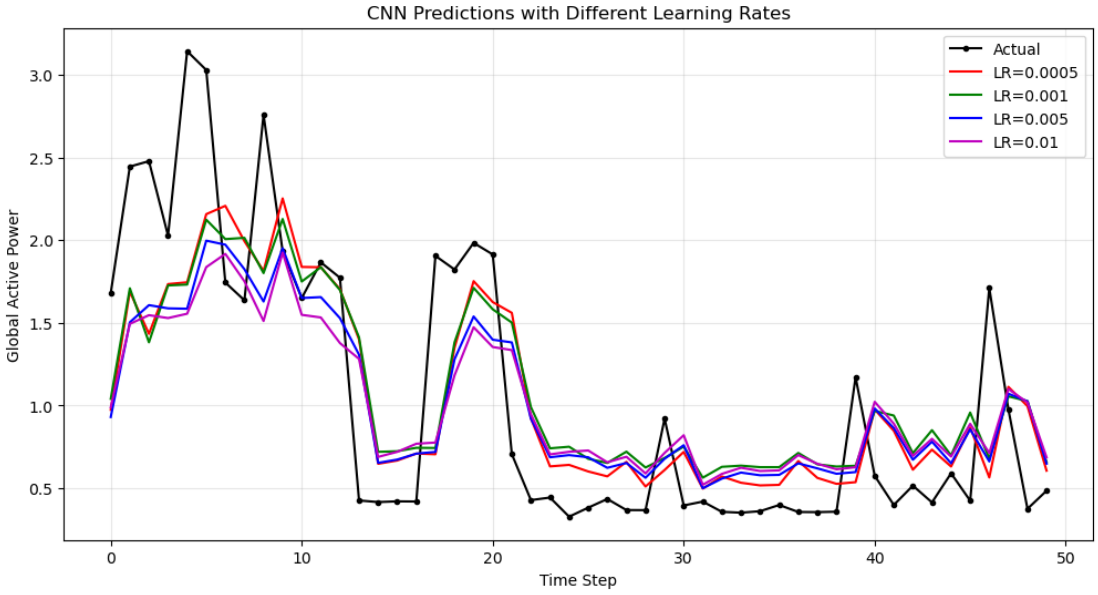
Prediction accuracy: Lower learning rates (0.0005 and 0.001) yield better test performance in terms of MSE and RMSE, while larger learning rates (0.005, 0.01) slightly degrade accuracy.

Training time: Training is faster with higher learning rates due to fewer epochs, but this comes at the cost of higher final loss and error metrics.

Loss trend: Across all learning rates, the validation loss remains relatively stable with early stopping controlling overfitting.

Conclusion:

For CNN-based short-term load forecasting, a moderate learning rate around 0.0005 - 0.001 achieves the best trade-off between training efficiency and prediction accuracy. Larger learning rates accelerate convergence but slightly sacrifice precision. This exploratory study highlights the importance of tuning learning rate as a critical hyperparameter and can guide further model optimization.



5.6. Exploratory Attempt: CNN + LSTM Forecast

We conducted an exploratory attempt using a hybrid CNN + LSTM model to forecast electricity consumption. The model architecture consists of:

- 1D Convolutional layer to extract local temporal features.
- Max-pooling layer to reduce sequence length and avoid overfitting.

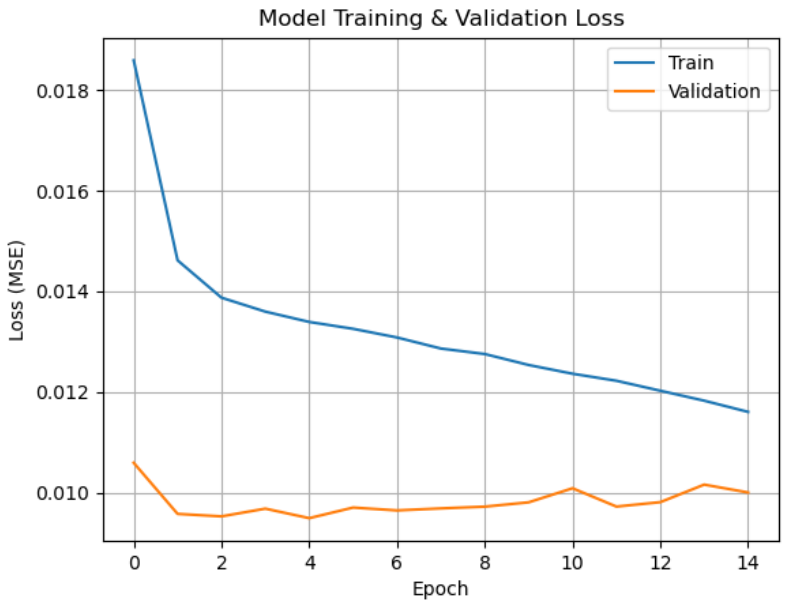
- Bidirectional LSTM to capture long-term dependencies in both directions.
- Additional LSTM layer to refine sequence representation.
- Fully connected layers for final forecast of the next 168 hours (1 week).

Training setup:

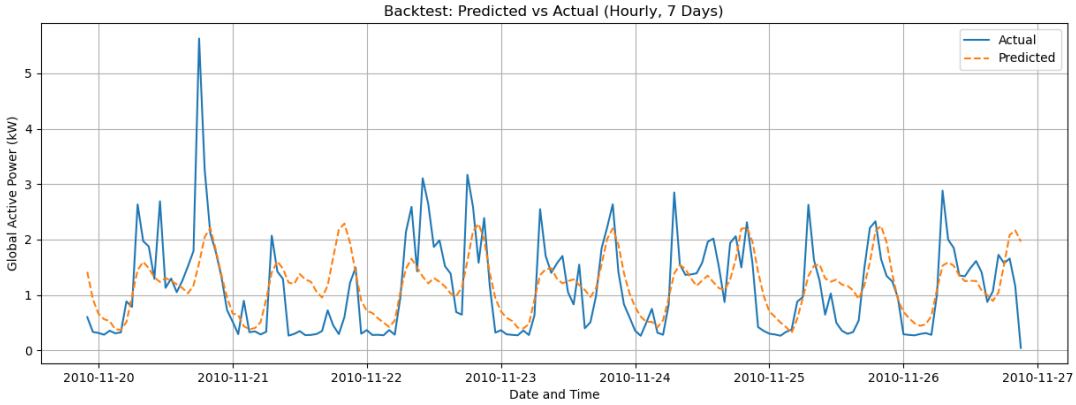
- Loss function: Mean Squared Error (MSE)
- Optimizer: Adam
- Early stopping patience: 10 epochs
- Look-back window: 168 hours (1 week)
- Forecast horizon: 168 hours (next week)

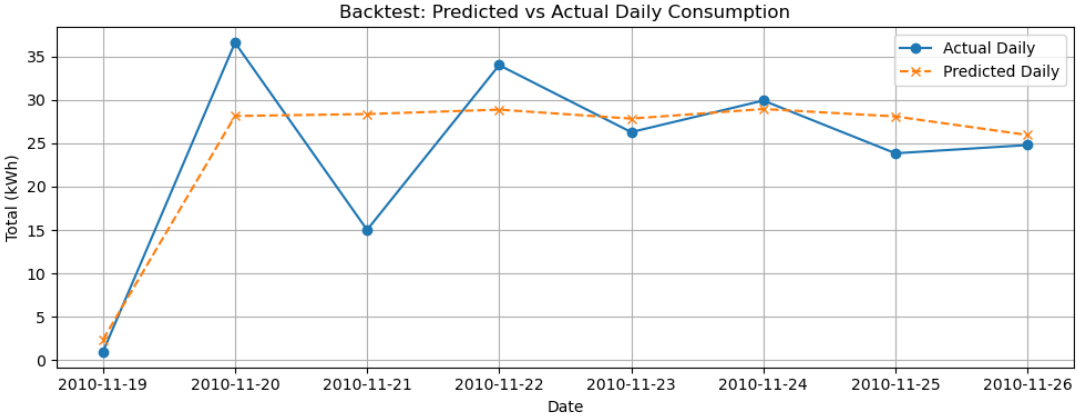
Analysis:

- Early stopping triggered at epoch 15, indicating convergence before the maximum 50 epochs.



- Global MAE and RMSE indicate reasonable predictive performance for weekly forecasts.
- The CNN layer helps extract short-term temporal patterns, while the LSTM layers model long-term dependencies, improving forecast accuracy for the next 168 hours.





- Forecasted daily electricity consumption shows the model captures the general trend, although some daily fluctuations are smoothed.