A Comparative Study of 1D CNN and LSTM for Household Energy Consumption Prediction

A Project Report

Submitted by:

SMARAN SATAPATHY (Regd. No: 2241019342)

T VINITA (Regd. No: 2241022012)

SAI HARISANKAR BEHERA (Regd. No: 2241002106)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHONOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Faculty of Engineering and Technology, Institute of Technical Education and Research

SIKSHA 'O' ANUSANDHAN (DEEMED TO BE) UNIVERSITY

Bhubaneswar, Odisha, India

ACKNOWLEDGEMENT

We, the undersigned students of **B.Tech of Computer Science(AI & ML) Department** hereby declare that we own the full responsibility for the information, results etc provided in the project titled **“A Comparative Study of 1D CNN and LSTM for Household Energy Consumption Prediction”** submitted to **Siksha ‘O’ Anusandhan Deemed to be University,Bhubaneswar** for the partial fulfillment of the subject **Machine Learning Projects with Python (CSE 4192)** have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

SMARAN SATAPATHY **T VINITA**

SAI HARISANKAR BEHERA

Place: Bhubaneshwar

Date: 27/12/2025

DECLARATION

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/fact/source in our submission. We understand that any violation of the above will cause for disciplinary action by the University and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken when needed.

SMARAN SATAPATHY
(2241019342)

T VINITA
(2241022012)

SAI HARISANKAR BEHERA
(2241002106)

Date: 27/12/2025

ABSTRACT

The significance of this project lies in the comparison of Long Short-Term Memory Networks (LSTM Networks) and 1D Convolutional Neural Networks (1D-CNN Networks) in predicting the consumption of electricity in a house. The central idea of this project is the prediction of the Global Active Power consumption in the next hour given the consumption of the past 24 hours. The data used in this project consists of over 2 million samples resampled hourly.

The design of the LSTM model consists of two layers for handling long-term dependencies, while the 1D CNN is implemented with 64 filters. From the experimental outputs, it is clear that the performance of the LSTM model is better than that of the 1D CNN, producing lower Root Mean Squared Error values with 0.5291 for the LSTM model against 0.5852 for the 1D CNN. Although the 1D CNN provides much faster computation with parallel processing, it is clear that the LSTM model is more effective at handling cyclical processes and time series analysis. The outcome clearly identifies that 1D CNN is an efficient method, but it is better suited to energy demand analysis.

Keywords: Household Power Consumption, LSTM, 1D-CNN, Time-Series Forecasting, Global Active Power, Deep Learning, RMSE, Neural Networks.

List of Figures

Timeline/Gantt Chart

Description	Duration
Problem Understanding and Requirement Analysis	2 Dec – 4 Dec
Data Collection and Dataset Exploration	5 Dec- 7 Dec
Data Preprocessing & Feature Engineering	8 Dec – 12 Dec
Model Development (LSTM & 1D-CNN)	13 Dec – 16 Dec
Model Training & Evaluation	17 Dec – 19 Dec
Result Analysis & Model Comparison	20 Dec – 21 Dec
Documentation & Final Submission	22 Dec – 24 Dec

TABLE OF CONTENTS

1.1 Problem Definition

This project has the objective to predict future electricity use by households from the past series of data. This project provides the prediction for the next hour's Global Active Power. Comparison of two machine learning models: Long Short-Term Memory (LSTM) Network, 1-Dimensional Convolutional Neural Network (1D-CNN).

1.2 Project Overview/Specifications

Specification	**Detail**
Initial data size	2,075,259 entries
Training Set Size	33,868 hours
Testing Set Size	720 hours
Resampling Strategy	Hourly Frequency
Evaluation Metrics	Root Mean Squared Error(RMSE) and Mean Absolute Error(MAE)
Graphs	Loss curves(training stability), Predicted vs Actual plots(tracking accuracy) and Residual plots(error magnitude analysis)
Models Used	Long Short Term Memory(LSTM) and 1 Dimensional Convolutional Neural Network(1D CNN)
Epochs trained	100 epochs
Optimizer	Adam
Data Shuffling	Disabled

1.3 Hardware Specification

The project is developed and executed in the Google Colab environment. Following are the hardware considerations:

Platform: Google Colab

Processing Units: CPU

Memory: 8 GB RAM

1.4 Software Specification

1.4.1 Core Programming Language and Environment

This project uses **Python**, which is normally run inside a **Jupyter Notebook** environment - as might be guessed by the file name ending in *.ipynb*, and is hosted on **Google Colab**.

1.4.2 Data Handling and Manipulation Libraries

- **Pandas:** Used for data import from the CSV file, creation of a Datetime Index by concatenating the Date and Time columns, resampling the data to an hourly frequency, and creating features.
- **NumPy:**It is used for carrying out arithmetic operations involving arrays, such as generating sequences for models as well as computing evaluation criteria like **Root Mean Squared Error**.

1.4.3 Deep Learning Framework and Libraries

- **TensorFlow/Keras:** For defining and training the neural network models**.**
- **Model Structure:**

  LSTM Structure: Uses "Sequential," two layers of LSTMs with 50 units and the relu activation function, and two Dropout layers with a dropout rate of 0.2, followed by a Dense layer with one unit.

  1D-CNN model structure: Consists of Sequential layers and uses Conv1D (64 filters, with a kernel size of 3), MaxPooling1D (pool size 2), Dropout (0.2), Flatten layers, and two Dense layers (50 units and 1 unit).

1.4.4 Model Preprocessing and Evaluation Tools

- **Scikit-learn:**

  MinMaxScaler: It is used for scaling.

  mean_squared_error and mean_absolute_error : Functions used for calculating performance metrics on the testing set.

- **Matplotlib:** It is utilized for creating graphs for the analysis, such as loss curves, graphs for the predicted versus actual values, residual graphs, and residual histograms.

2. LITERATURE SURVEY

2.1 Existing System:

(i) Household Electricity Consumption Prediction Under Multiple Behavioural Intervention Strategies Using Support Vector Regression.

This study evaluates the feasibility of using the relationship between outdoor temperature and heat demand to create long-term forecasts for district heating networks. Using Alvalade, Lisbon as a case study, the research found that while weather changes alone resulted in acceptable error margins (below 20%), introducing building renovation scenarios significantly increased prediction errors up to 59.5%.

(ii) Electricity Consumption Prediction using Machine Learning.

This paper explores various machine learning techniques—including Linear Regression, KNN, Random Forest, and ANN—to forecast power usage based on historical hourly data. The research demonstrates that these algorithms can accurately predict demand, with the K-Nearest Neighbours (KNN) model notably achieving a 90.92% accuracy rate in the tested scenarios.

(iii) Predicting residential electricity consumption patterns based on smart meter and household data: A case study from the Republic of Ireland.

This research analyzes residential electricity patterns in the Republic of Ireland by combining smart meter data with socio-demographic information. It identifies six distinct intra-day load profiles and uses an Elastic Net model to achieve nearly 85% accuracy in predicting a household's annual load profile based on factors like occupation, employment status, and household size.

2.2 Proposed System:

The system also uses and compares two different architectures used by neural networks to make predictions, namely:

• **Long Short Term Memory (LSTM) Model:** This model can detect long-term patterns in time series data by the use of two layers: *50 units* in each of the LSTM networks, along with the activation function *relu*, followed by dropout layers where the *dropout probability* value is *0.2*.

• **1D ConvolutionalNeural Network (1D-CNN) Model:** The feature extraction for *local temporal* information is accomplished using the *Conv1D layer* (with 64 filters and kernel size 3), followed by the *MaxPooling1D* layer for *down-sampling*. The architecture includes the *Flatten layer* and the *hidden layer* with densely connected neurons (Dense layer with 50 units).

2.3 Feasibility Study:

System Type	Core Algorithms	Key Features / Findings	Performance Highlights
Traditional Statistical ([1], [3])	Linear Regression, Questionnaires	Focus on heat demand/outdoor temperature or socio-demographics.	< 20% error for weather-only; up to 59.5% error with renovation scenarios.
Machine Learning [2]	KNN, XGBoost, Random Forest, ANN.	Used year-long hourly power utility data.	KNN achieved 90.92% accuracy.
Behavior-Based SVR ([1],[2])	Support Vector Regression (SVR) with RBF Kernel.	Incorporates personality traits, energy behaviours, and intervention strategies.	RBF-SVR outperformed Linear/Polynomial SVR in monthly forecasting.
Socio-Analytics ([3])	Elastic Net, K-Medoids Clustering	Uses smart meter data paired with socio-demographic features (Irish households).	Elastic Net achieved ~85% test accuracy for user profile classification.
Proposed System (Project)	LSTM & 1D-CNN	Uses 24-hour lookback on hourly resampled data to predict the next hour.	LSTM RMSE: 0.5291; outperforms CNN in peak prediction.

3. SYSTEM ANALYSIS & DESIGN

3.1 Requirement Specification

Its main task is the short-term load forecasting, which includes the prediction of consumption for Global Active Power for the upcoming hour. For any given prediction, the system retains data for the last 24 hours.

• **Functional Requirements:** The system should operate on a raw dataset containing **over 2 million records**, it should clean the null entries of **25,979** in **Sub-metering_3**-by performing time interpolations and back/forward fillings. To make the system computation-friendly for the effective analysis of morning peaks and evening peaks, there should be a resampling of the dataset into an hourly series that would reduce the size of the dataset to about **30,000 records**.

• **Performance Requirements:** The system shall compare two Deep Learning approaches: *Long Short Term Memory* Networks and *1D-Convolutional Neural Networks*.

• **Success Criteria:** *RMSE* and *MAE* offer measures of success where the lower this is the better. The project will create three graphs: Loss Curve Graphs, Graphs of Predicted against Actual Values, Residual Error Graphs.

3.2 Flowcharts

While the sources provide a "System Architecture" diagram for general machine learning workflows, the specific project follows a linear data-to-prediction pipeline rather than a complex relational database structure:

1. Data Source: household_power_consumption.txt resampled to df_hourly**.**

2. Processing Flow:

Dataset insertion→ Numeric Conversion → Resampling → Interpolation → Feature Engineering → Normalization → Sequence Creation → Model Training → Evaluation.

3. Architecture Flow: The model utilizes a Sequential method where data flows through stacked recurrent or convolutional layers into a final dense prediction neuron.

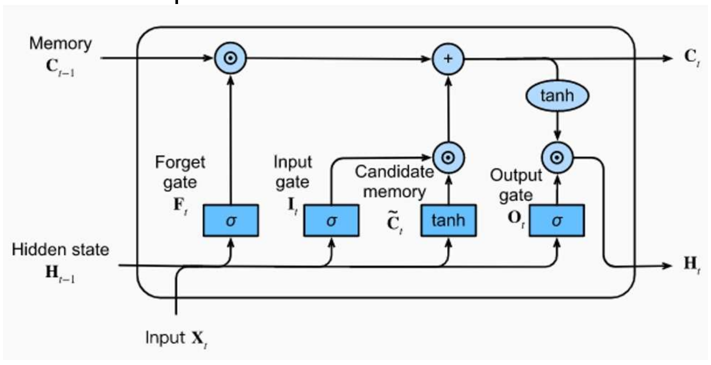


Figure 1: LSTM Architecture

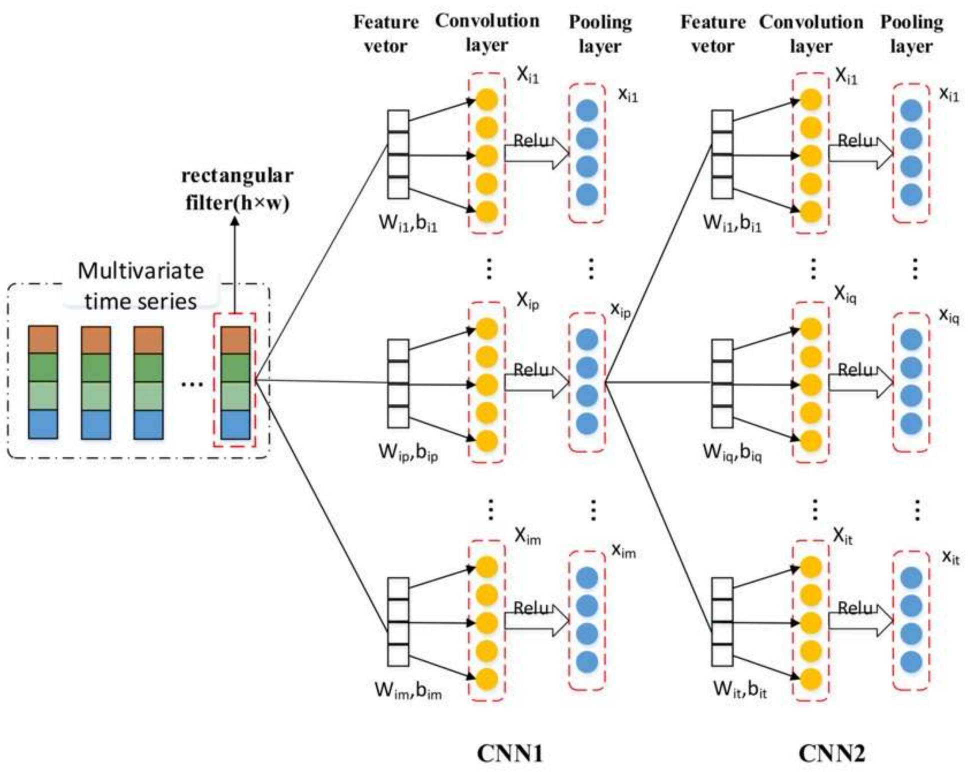


Figure 2: CNN Architecture

3.3 Design and Test Steps / Criteria

The design process follows a rigorous experimental setup to ensure model stability:

• Design Steps:

1. Temporal Feature Engineering: Extraction of 'Hour' and 'DayOfWeek' alongside a 1-step lag of Global Active Power to provide historical context.

2. Normalization: Use of **MinMaxScaler** to scale all features into a consistent range for efficient gradient descent.

3. Sliding Window Logic: Creating 3D input arrays with a lookback of 24 time steps.

• Test Criteria:

◦ **Generalization:** The model should maintain a low and stable validation loss (approx. 0.19) over 100 epochs**.**

◦ **Peak Tracking:** The model must accurately track cyclical patterns without excessive "smoothing" of the highest consumption events.

◦ **Error Distribution:** Residuals should be centred near zero with minimal large positive spikes.

3.4 Algorithms and Pseudo Code

3.4.1 LSTM Algorithm (Temporal Memory)

The LSTM is designed with two stacked layers to capture long-term dependencies.

• Step 1: Initialize a Sequential model.

• Step 2: Add an LSTM layer with 50 units, relu activation, and return_sequences=True to pass the sequence to the next layer.

• Step 3: Add a Dropout layer (0.2) to mitigate overfitting.

• Step 4: Add a second LSTM layer with 50 units and relu activation.

• Step 5: Add a Dense output layer (1 unit) for the final prediction.

• Step 6: Compile with the Adam optimizer and MSE loss.

• Step 7: Train for 100 epochs with shuffle=False to preserve time-series order.

Layer No.	**Layer Type**	**Specification / Hyperparameters**	**Output Shape**	**Param #**
1	LSTM	50 units, relu activation, return_sequences=True	(None, 24, 50)	12,200
2	Dropout	0.2 (20% regularization rate)	(None, 24, 50)	0
3	LSTM	50 units, relu activation	(None, 50)	20,200
4	Dropout	0.2 (20% regularization rate)	(None, 50)	0
5	Dense	1 unit (Output Layer)	(None, 1)	51
Total	Trainable	32,451 parameters	—	—

3.4.2 1D-CNN Algorithm (Local Feature Extraction)

The 1D-CNN focuses on local temporal patterns and trends across the 24-hour window.

• Step 1: Initialize a Sequential model.

• Step 2: Add a Conv1D layer with 64 filters and a kernel size of 3,.

• Step 3: Add a MaxPooling1D layer (pool size 2) to downsample features.

• Step 4: Flatten the 2D feature map into a 1D vector.

• Step 5: Add a Dense hidden layer (50 units) with relu activation.

• Step 6: Add a final Dense output layer (1 unit).

• Step 7: Compile with Adam/MSE and train for 100 epochs with a batch size of 32.

Layer No.	**Layer Type**	**Specification / Hyperparameters**	**Output Shape**	**Param #**
1	Conv1D	64 filters, kernel size 3, relu activation	(None, 22, 64)	1,984
2	MaxPooling1D	Pool size 2	(None, 11, 64)	0
3	Dropout	0.2 (20% regularization rate)	(None, 11, 64)	0
4	Flatten	Converts 2D feature map to 1D vector	(None, 704)	0
5	Dense	50 units, relu activation (Hidden Layer)	(None, 50)	35,250
6	Dense	1 unit (Output Layer)	(None, 1)	51
Total	Trainable	37,285 parameters	—	—

3.5 Testing Process

The testing phase evaluates the models on a 720-hour (30-day) unseen dataset.

1. Forecast Execution: The system uses model.predict() on the test sequences to generate hourly forecasts.

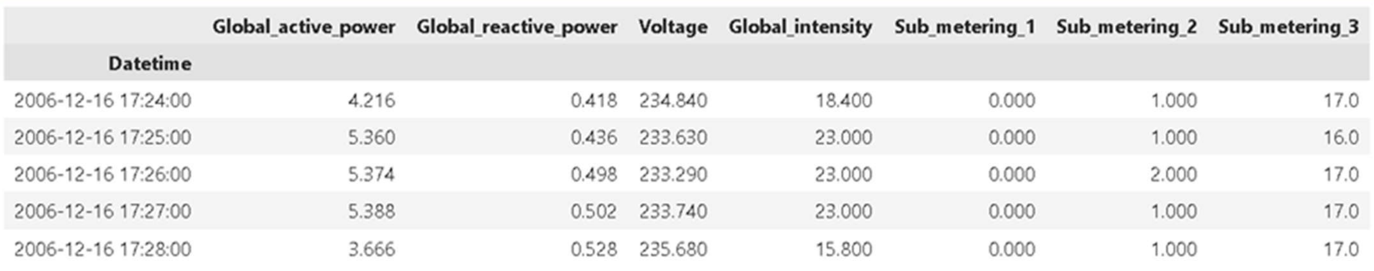
2. Metric Calculation: Calculation of RMSE and MAE; current results show the LSTM (RMSE: 0.5291) technically outperforming the CNN (RMSE: 0.5402).

3. Residual Analysis: Plotting the difference between actual and predicted values (*y_true,y_pred*),. This identifies that the CNN suffers from larger positive spikes (up to +4.0) compared to the more constrained LSTM errors.

4. Distribution Histogram: Comparing the density of errors; the LSTM distribution is taller and narrower, indicating a higher frequency of near-zero (small) errors.

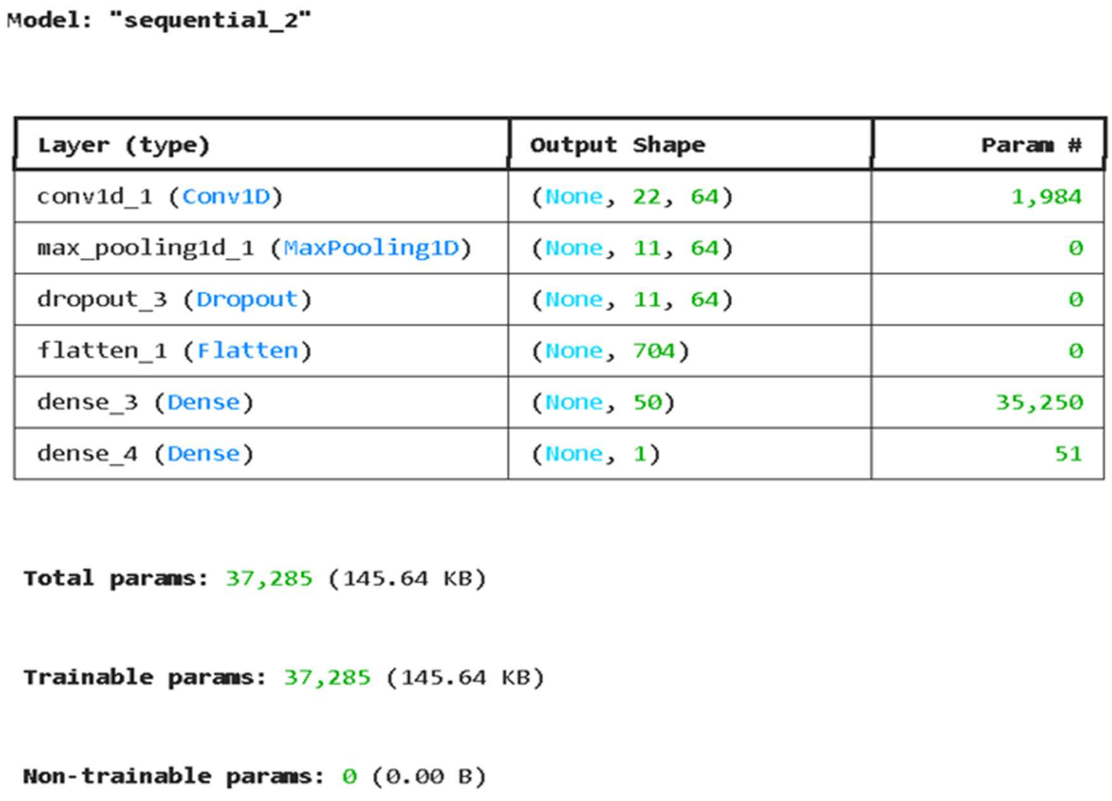
5. Step-by-Step Validation: A detailed side-by-side comparison of the first 10 forecast steps is performed to identify which model "wins" at specific hours.

4. RESULTS



Datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
2006-12-16 17:24:00	4.216	0.418	234.840	18.400	0.000	1.000	17.0
2006-12-16 17:25:00	5.360	0.436	233.630	23.000	0.000	1.000	16.0
2006-12-16 17:26:00	5.374	0.498	233.290	23.000	0.000	2.000	17.0
2006-12-16 17:27:00	5.388	0.502	233.740	23.000	0.000	1.000	17.0
2006-12-16 17:28:00	3.666	0.528	235.680	15.800	0.000	1.000	17.0

Figure 3: Sample Records from the Household Power Consumption Dataset



```
Model: "sequential_2"

┌──────────────────────────────────┬────────────────────────────┬─────────────────┐
│ Layer (type)                     │ Output Shape               │         Param # │
├──────────────────────────────────┼────────────────────────────┼─────────────────┤
│ conv1d_1 (Conv1D)                │ (None, 22, 64)             │           1,984 │
├──────────────────────────────────┼────────────────────────────┼─────────────────┤
│ max_pooling1d_1 (MaxPooling1D)   │ (None, 11, 64)             │               0 │
├──────────────────────────────────┼────────────────────────────┼─────────────────┤
│ dropout_3 (Dropout)              │ (None, 11, 64)             │               0 │
├──────────────────────────────────┼────────────────────────────┼─────────────────┤
│ flatten_1 (Flatten)              │ (None, 704)                │               0 │
├──────────────────────────────────┼────────────────────────────┼─────────────────┤
│ dense_3 (Dense)                  │ (None, 50)                 │          35,250 │
├──────────────────────────────────┼────────────────────────────┼─────────────────┤
│ dense_4 (Dense)                  │ (None, 1)                  │              51 │
└──────────────────────────────────┴────────────────────────────┴─────────────────┘

 Total params: 37,285 (145.64 KB)

 Trainable params: 37,285 (145.64 KB)

 Non-trainable params: 0 (0.00 B)
```

Figure 4: Model summary of the proposed 1D CNN architecture used for time-series regression, showing convolution, pooling, regularization, and dense layers with total trainable parameters.

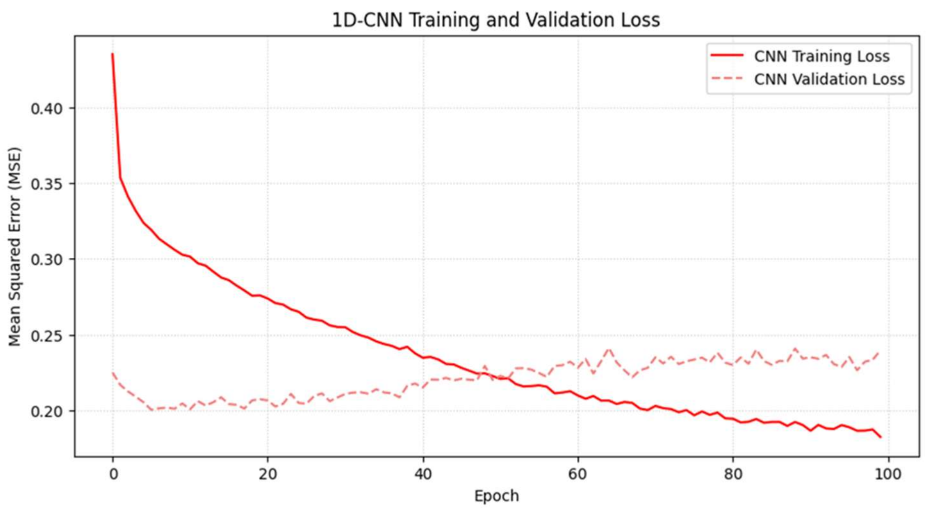


Figure 5: Training and Validation Loss of 1D-CNN

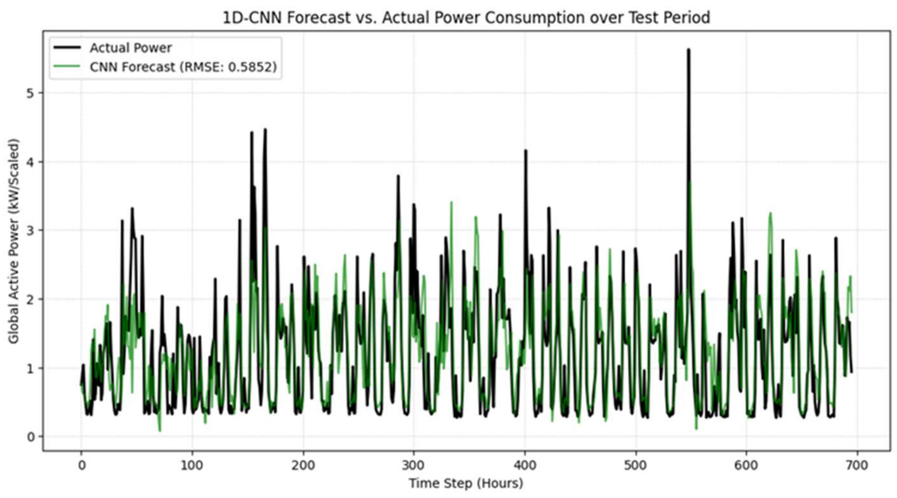


Figure 6: 1D-CNN Forecast vs. Actual Power Consumption

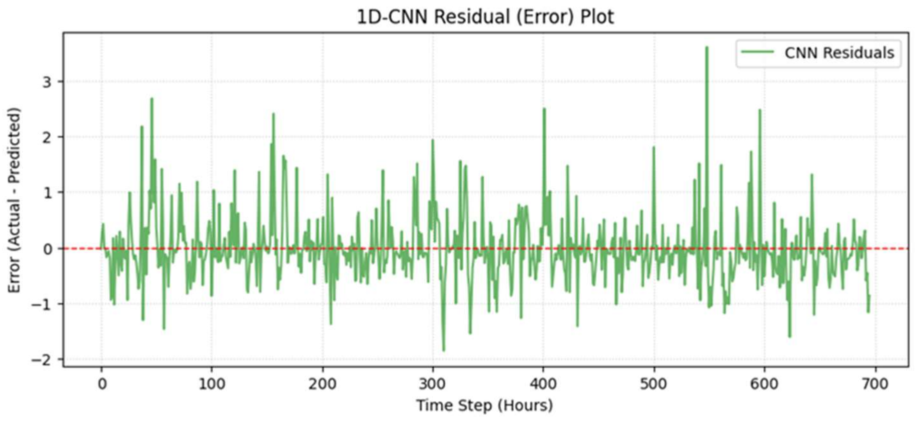


Figure 7: Residual Error Analysis of 1D-CNN

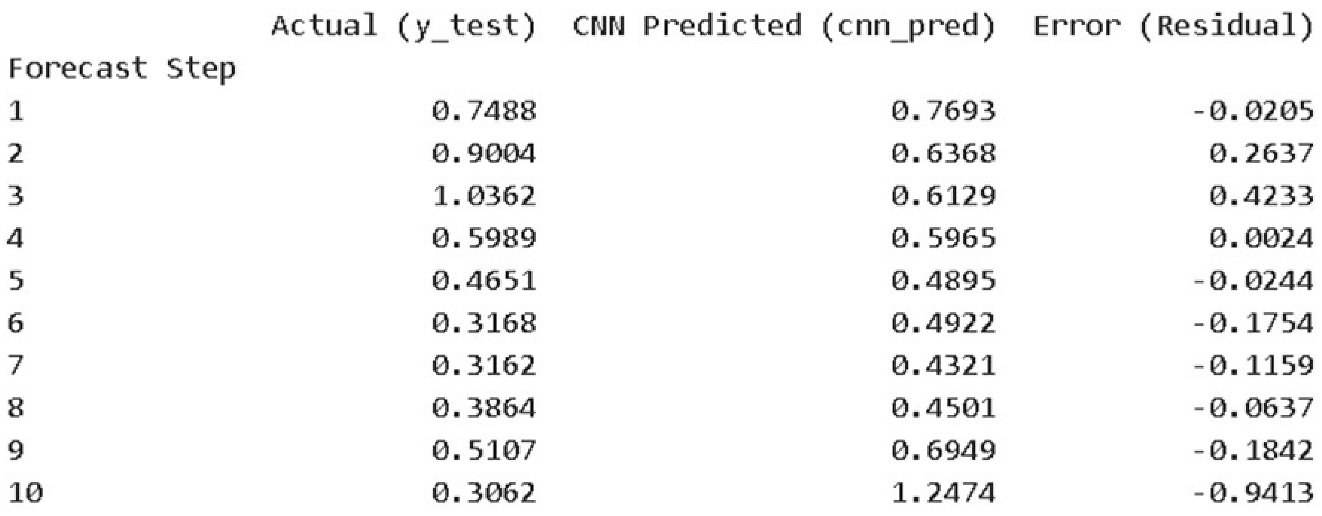
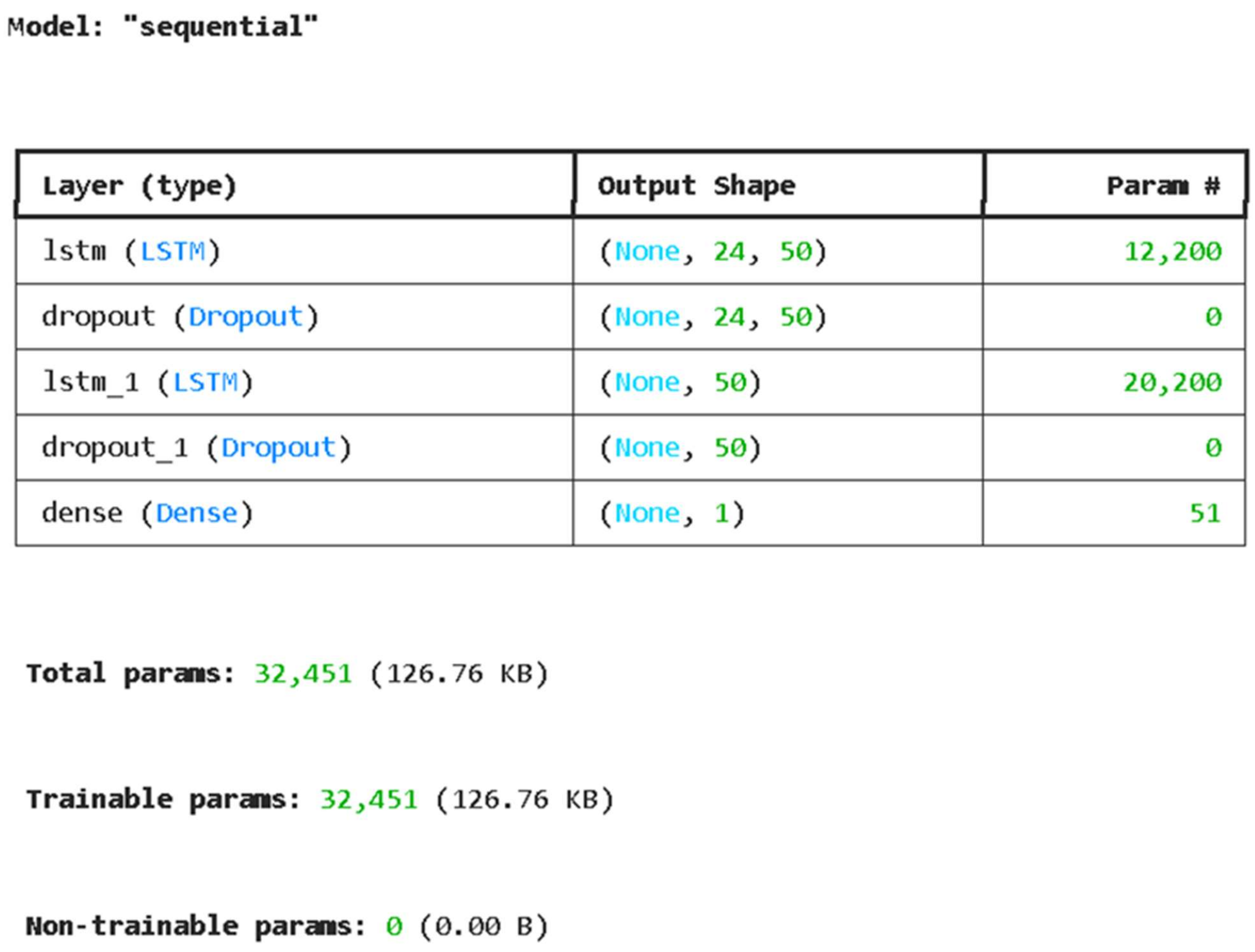

```
               Actual (y_test)  CNN Predicted (cnn_pred)  Error (Residual)
Forecast Step
1                       0.7488                    0.7693           -0.0205
2                       0.9004                    0.6368            0.2637
3                       1.0362                    0.6129            0.4233
4                       0.5989                    0.5965            0.0024
5                       0.4651                    0.4895           -0.0244
6                       0.3168                    0.4922           -0.1754
7                       0.3162                    0.4321           -0.1159
8                       0.3864                    0.4501           -0.0637
9                       0.5107                    0.6949           -0.1842
10                      0.3062                    1.2474           -0.9413
```

Figure 8: Comparison of actual values, 1D-CNN predictions, and residual errors for selected forecast steps.



```
Model: "sequential"

┌──────────────────────────────────┬────────────────────────┬────────────────┐
│ Layer (type)                     │ Output Shape           │        Param # │
├──────────────────────────────────┼────────────────────────┼────────────────┤
│ lstm (LSTM)                      │ (None, 24, 50)         │         12,200 │
├──────────────────────────────────┼────────────────────────┼────────────────┤
│ dropout (Dropout)                │ (None, 24, 50)         │              0 │
├──────────────────────────────────┼────────────────────────┼────────────────┤
│ lstm_1 (LSTM)                    │ (None, 50)             │         20,200 │
├──────────────────────────────────┼────────────────────────┼────────────────┤
│ dropout_1 (Dropout)              │ (None, 50)             │              0 │
├──────────────────────────────────┼────────────────────────┼────────────────┤
│ dense (Dense)                    │ (None, 1)              │             51 │
└──────────────────────────────────┴────────────────────────┴────────────────┘

 Total params: 32,451 (126.76 KB)

 Trainable params: 32,451 (126.76 KB)

 Non-trainable params: 0 (0.00 B)
```

Figure 9: LSTM Model Architecture

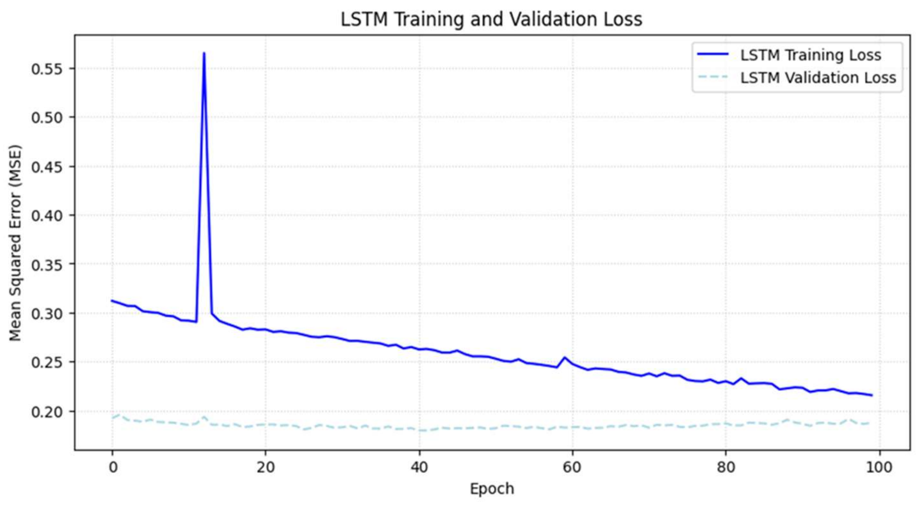


Figure 10: LSTM Training and Validation Loss

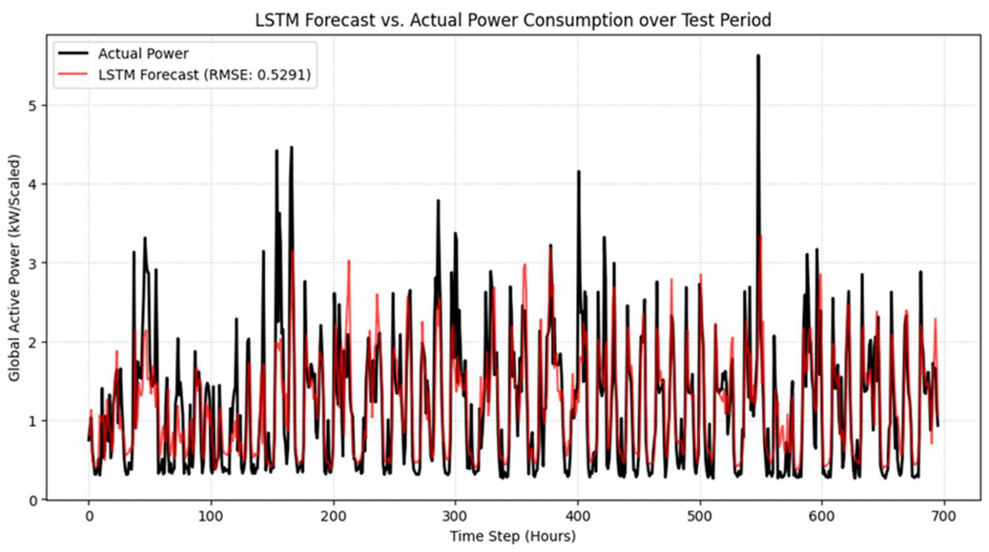


Figure 11: LSTM Forecast vs. Actual Power Consumption

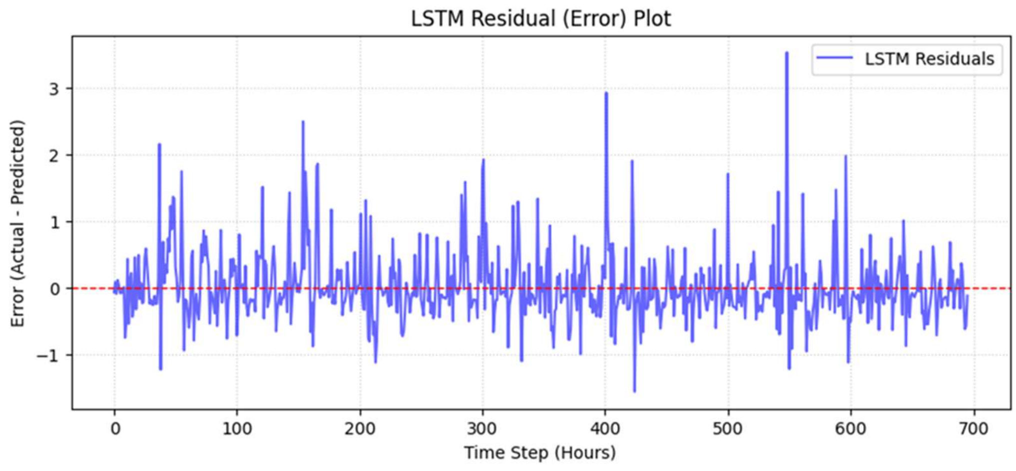
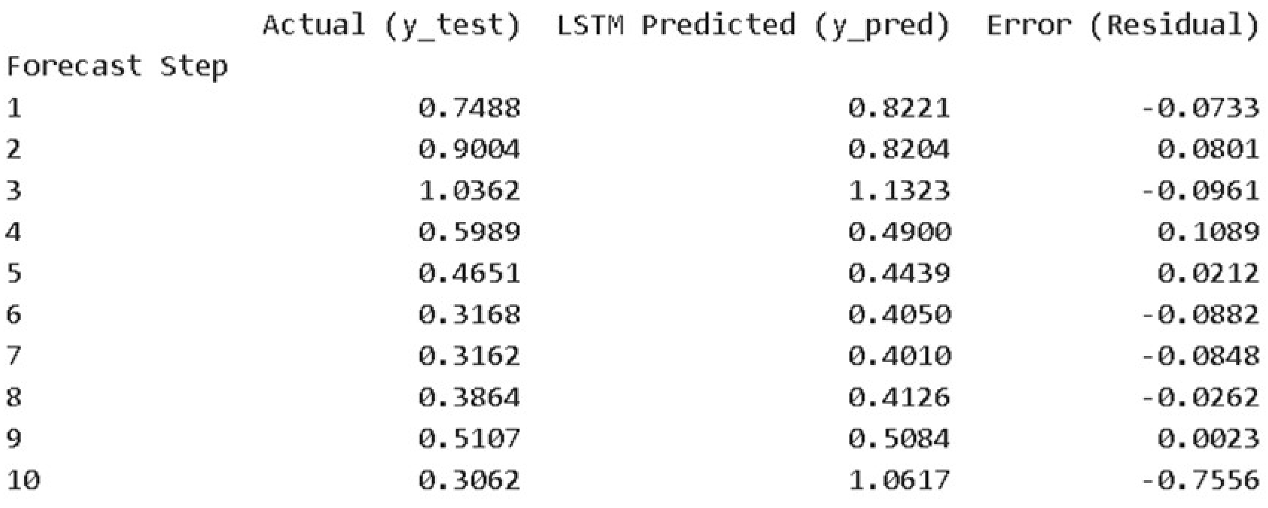


Figure 12: Residual Error Analysis of LSTM



```
               Actual (y_test)  LSTM Predicted (y_pred)  Error (Residual)
Forecast Step
1                       0.7488                   0.8221           -0.0733
2                       0.9004                   0.8204            0.0801
3                       1.0362                   1.1323           -0.0961
4                       0.5989                   0.4900            0.1089
5                       0.4651                   0.4439            0.0212
6                       0.3168                   0.4050           -0.0882
7                       0.3162                   0.4010           -0.0848
8                       0.3864                   0.4126           -0.0262
9                       0.5107                   0.5084            0.0023
10                      0.3062                   1.0617           -0.7556
```

Figure 13: LSTM Sample Predictions and Residuals

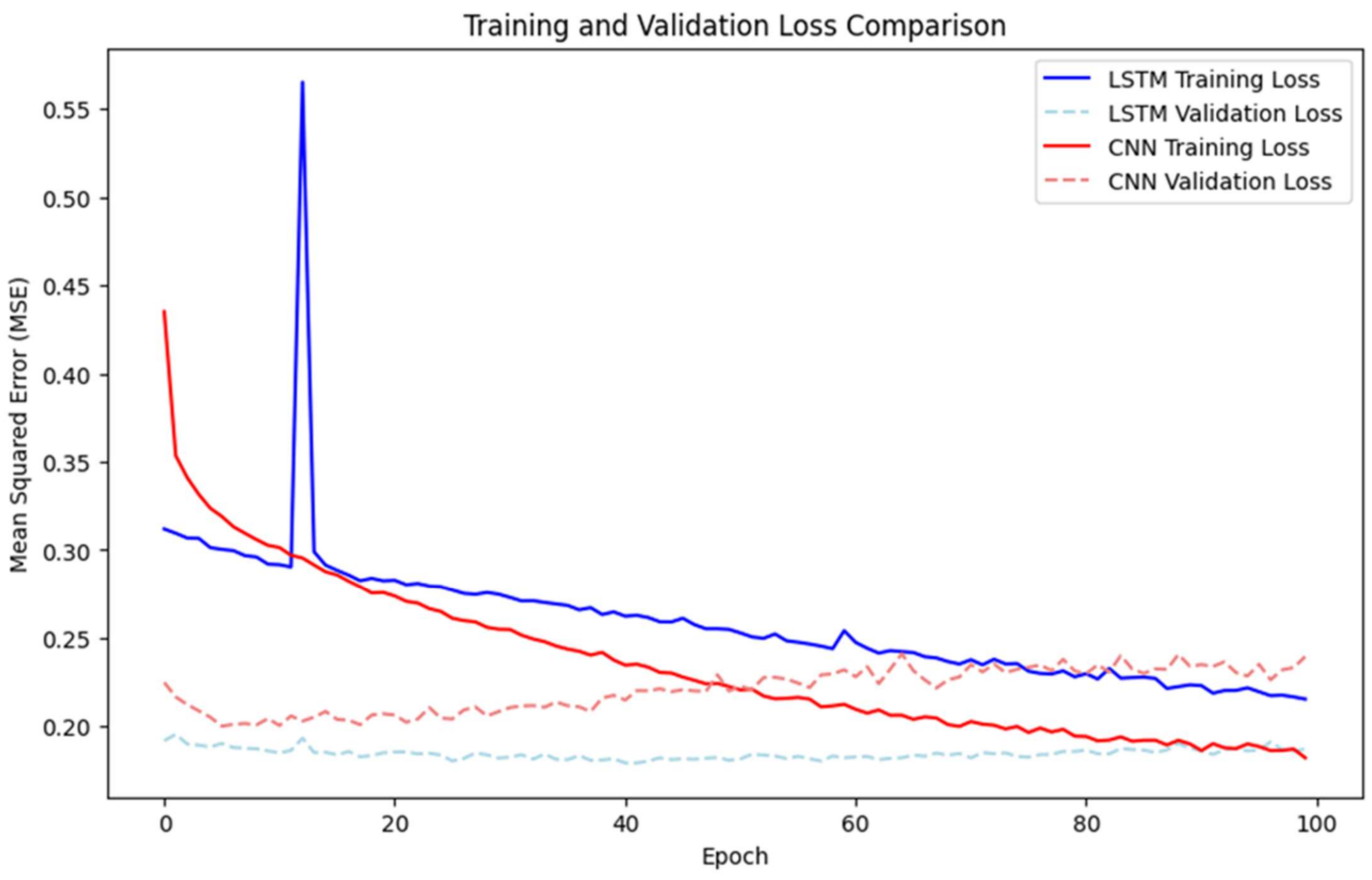


Figure 14: Training and Validation Loss Comparison (CNN vs LSTM)

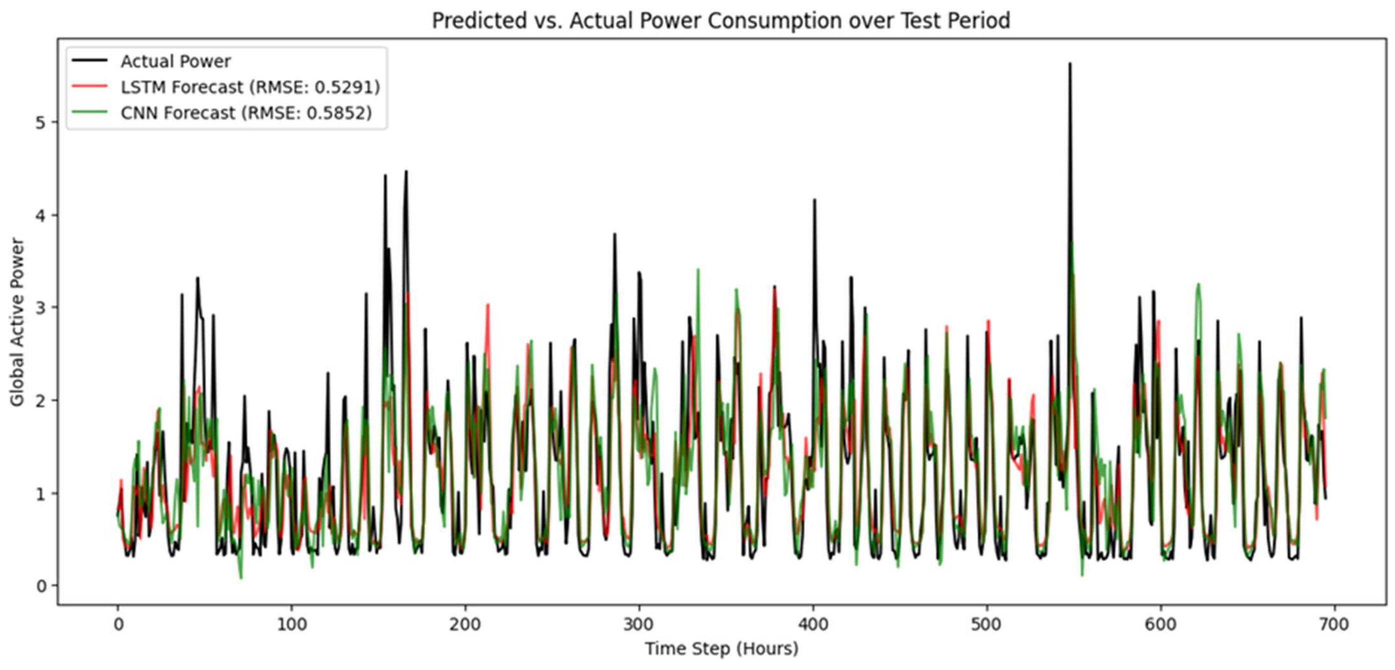


Figure 15: Predicted vs. Actual Power Consumption (CNN vs LSTM)

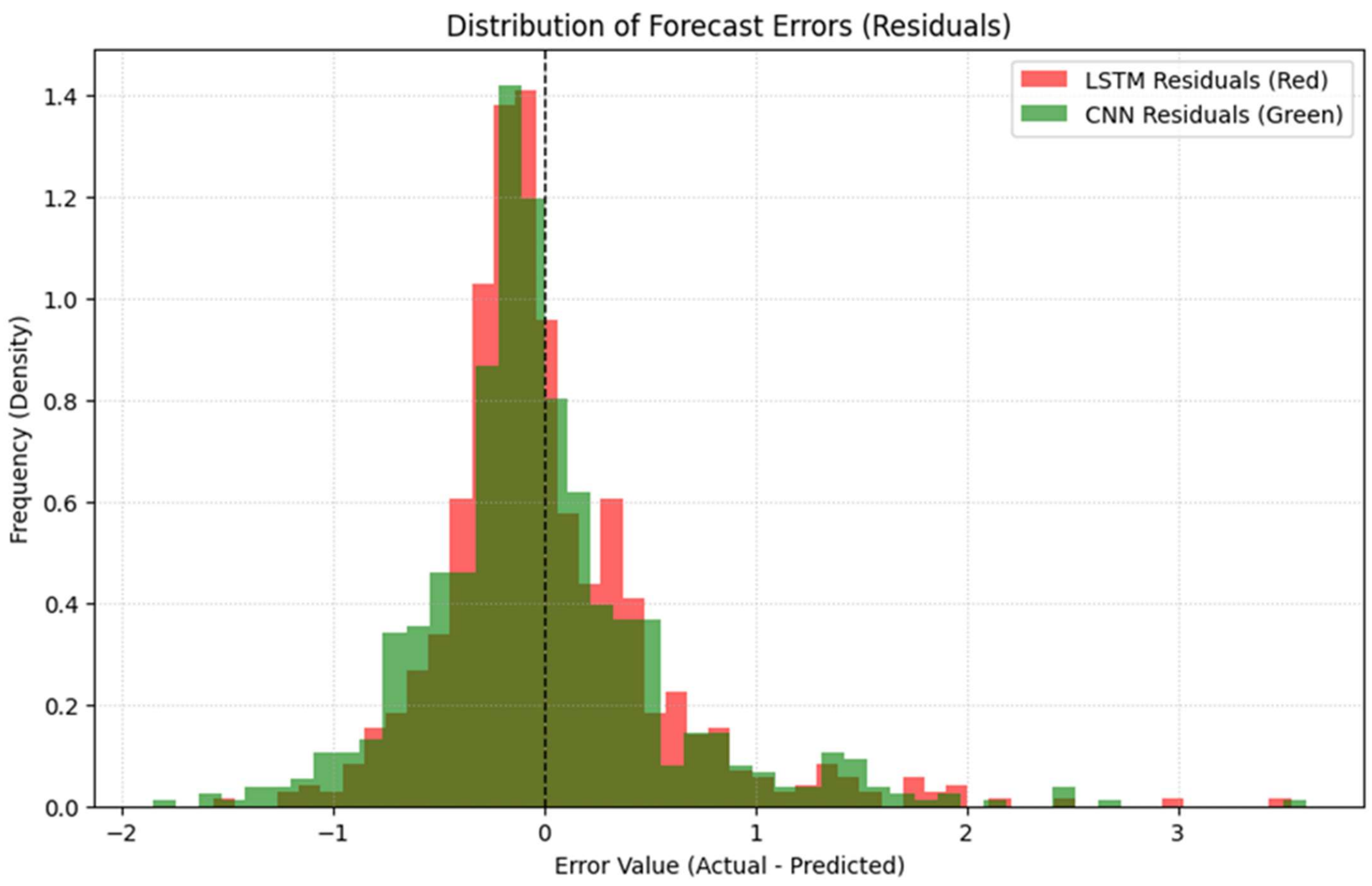


Figure 16: Distribution of Forecast Errors (Residuals)

Metric	**LSTM Model**	**1D CNN Model**
Root Mean Squared Error (RMSE)	0.5291	0.5852
Mean Absolute Error (MAE)	0.3622	0.4024
Validation Error	0.1872	0.2395
Epochs Trained	100	100

Figure 17: Final model comparison

5.CONCLUSION

The project was able to provide a viable solution for forecasting short-term residential electricity consumption patterns via deep learning approaches. The proposed system showed how consumption patterns, which are affected by temporal behavior, can actually be modeled using time-series approaches based on the data gathered from smart-metering. The research work compared the performance of two different deep learning models, namely **LSTM** and **1D-CNN**, for their ability in predicting the hourly power consumption.

A full data preprocessing workflow was implemented to guarantee the accuracy of the forecasting models. Raw data was preprocessed and converted into an integrated form for all time series data at an hourly resolution for smoothing out variations and supporting model stability. Various attribute features based on date and lag features based on consumption amounts were employed to help the models identify the consumption patterns in the dataset. **Min-Max normalization** was also used for effective neural network model training.

The proposed **LSTM** model was able to capture the *long-term dependencies* existing within the electrical energy consumption data. The ability of the model to utilize memory facilitated the capture of the changes and cycles within the electrical energy data over time. Therefore, the model was able to make smooth predictions and perform well when learning the sequences. However, the model took *longer* to train.

On the contrary, the role of the **1D-CNN** model involved emphasizing the discovery of *local temporal relationships* within the data through convolutional filters. Its inherent ability to work in parallel ensured that its computations were completed at a significantly higher speed. Although the CNN model successfully explored the short-term variations in the usage of electricity, its dependency-extraction capability is comparatively *weaker* than that of the LSTM model.

Performance evaluation done using conventional regression error calculation tools like **Root Mean Squared Error** and **Mean Absolute Error** on both models indicated that both models had been able to make accurate predictions. Though, the **LSTM model** overall was able to make predictions that were more *stable* and *accurate* compared to both models during the testing phase, especially during *long-term predictions*.

Overall, this work has emphasized the capabilities of deep learning architectures for predicting energy demand and has also confirmed that the choice of the model should be driven by the characteristics of the dependency in the data. Even if 1D-CNN provides an efficient solution for overcoming the complexity, LSTM turned out to be *more appropriate* for simulating complex time series phenomena for residential electric energy consumption. The results of this project may help to choose the right procedures for efficient energy planning.

Limitations:

Even if the proposed system for electricity consumption forecasting performs well, there are a *few limitations* that need to be taken into consideration. The models have been trained on historical consumption values and time-related attributes. *Influential external variables* like weather patterns, population presence, use of appliances, and lifestyles have *not been considered*. These elements witness a *significant impac*t on electricity consumption in real scenarios. Henceforth, it *may influence the forecast* values to some extent.

Another limitation would be the static *24-hour look-back window* used in forecasting. Although it is effective in teaching models on a day-wise consumption process, it might *not capture* all trends in consumption on a *monthly or yearly basis*. This may be overcome by increasing the look-back window but would again add to the complexity of models.

Data quality is also another factor that affects the performance of the model. *Missing data* is handled using interpolation and the forward and backward approaches. Although interpolation and the use of the forward and backward approaches ensure data continuity, the approach leads to *estimations* and not actual data. Where large amounts of data are missing, estimates may not be an accurate indication of the actual consumption patterns.

From a computation standpoint, the training time of the LSTM model takes *longer* considering that the model is sequential, which might limit the use of the model in real-time or large-scale problems. In contrast, the 1D-CNN model, which takes less time to train, lacks the capability to capture long-term memory like the LSTM model, which might *affect the performance* of the model.

Lastly, the system is then validated utilizing a single dataset associated with a distinct household setting. Therefore, the accuracy and performance of the machine model may *differ* in relation to distinct households and regions as well as different consumption patterns. The machine model *may not generalize* well without further updates and modifications.

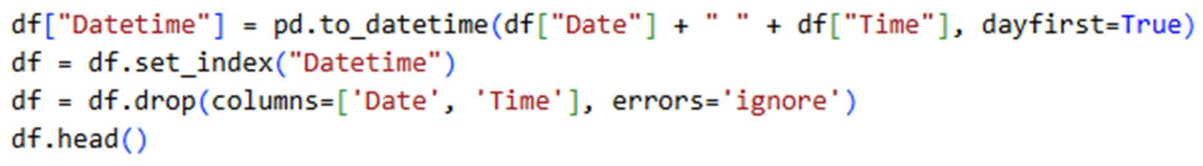
6. REFERENCES:

[1]. Meng Shen, Huiyao Sun and Yujie Lu (2017) *'Household Electricity Consumption Prediction Under Multiple Behavioural Intervention Strategies Using Support Vector Regression'*.

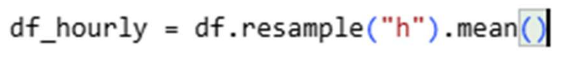
[2]. Reddy, G.V., Aitha, L.J., Poojitha, C., Shreya, A.N., Reddy, D.K. and Meghana, G.S. (2023) *'Electricity Consumption Prediction Using Machine Learning'*.

[3]. Guo, Z., O'Hanley, J.R. and Gibson, S. (2022) *'Predicting residential electricity consumption patterns based on smart meter and household data: A case study from the Republic of Ireland'*.

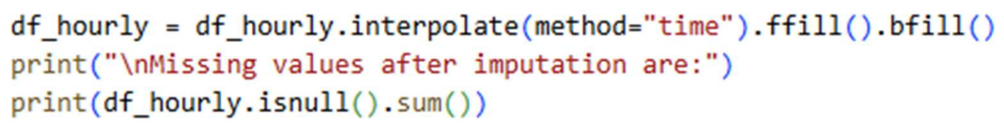
7. APPENDICES



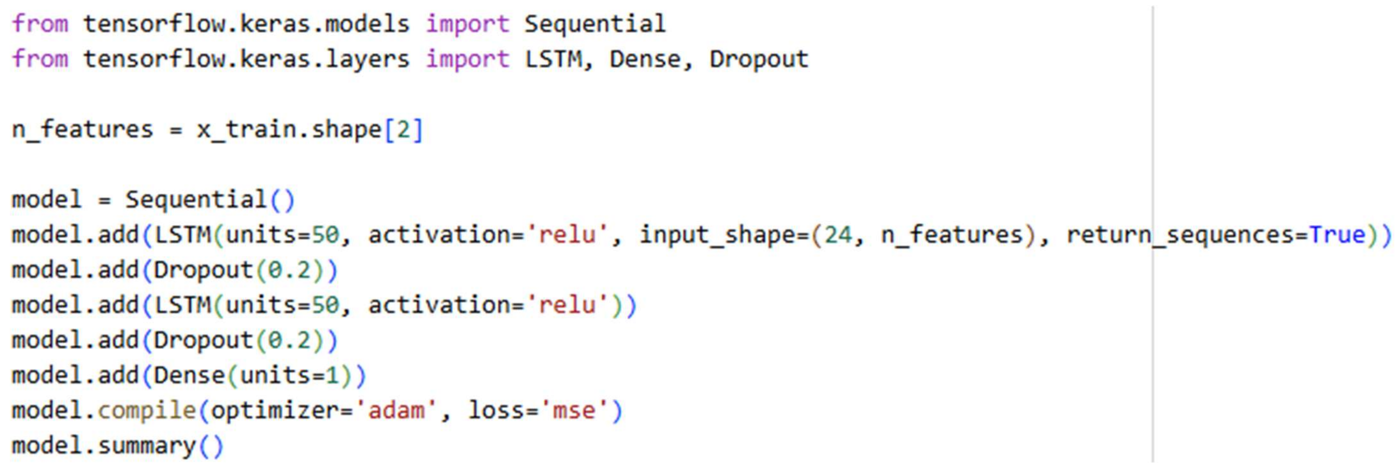
```
df["Datetime"] = pd.to_datetime(df["Date"] + " " + df["Time"], dayfirst=True)
df = df.set_index("Datetime")
df = df.drop(columns=['Date', 'Time'], errors='ignore')
df.head()
```



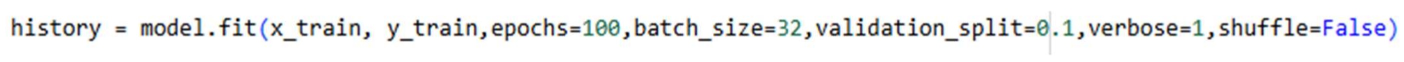
```
df_hourly = df.resample("h").mean()
```



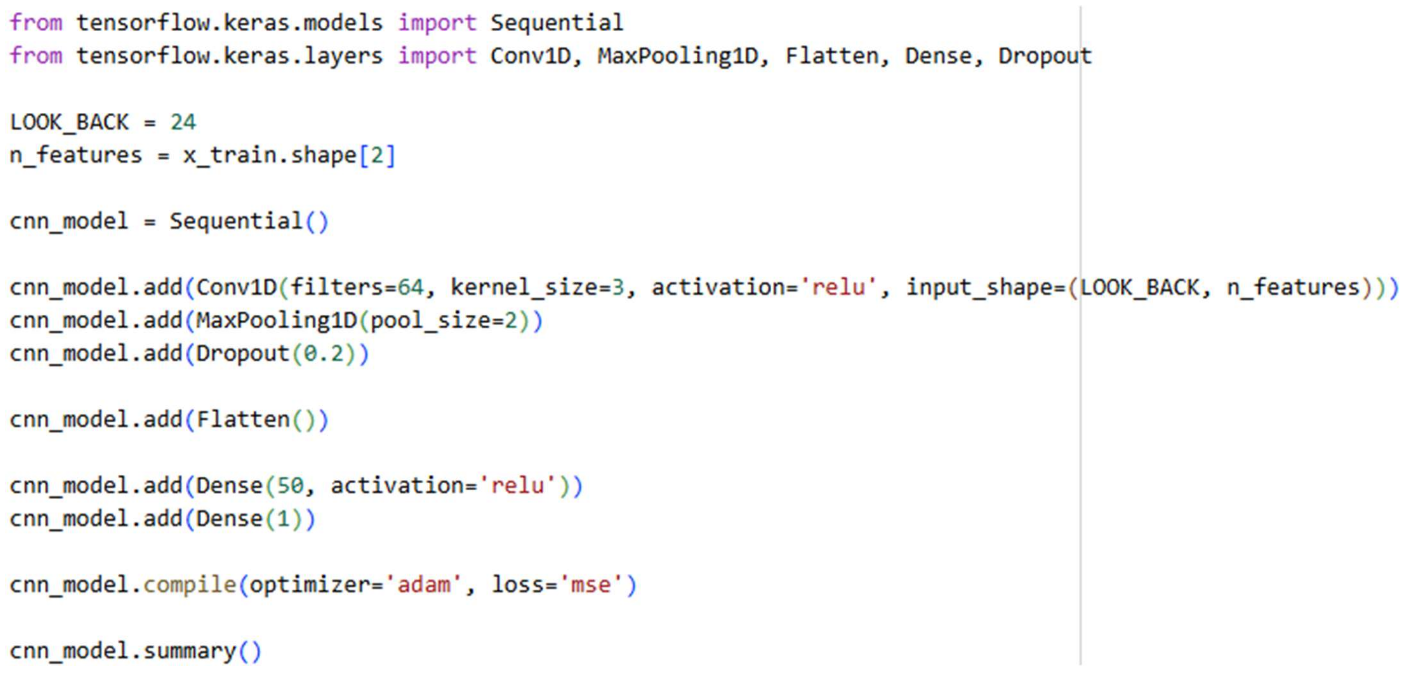
```
df_hourly = df_hourly.interpolate(method="time").ffill().bfill()
print("\nMissing values after imputation are:")
print(df_hourly.isnull().sum())
```



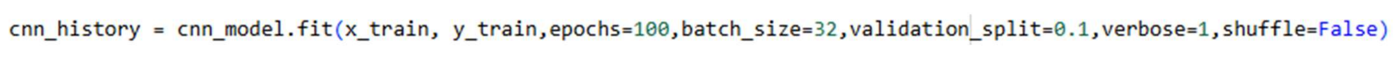
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

n_features = x_train.shape[2]

model = Sequential()
model.add(LSTM(units=50, activation='relu', input_shape=(24, n_features), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mse')
model.summary()
```



```
history = model.fit(x_train, y_train,epochs=100,batch_size=32,validation_split=0.1,verbose=1,shuffle=False)
```



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout

LOOK_BACK = 24
n_features = x_train.shape[2]

cnn_model = Sequential()

cnn_model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(LOOK_BACK, n_features)))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Dropout(0.2))

cnn_model.add(Flatten())

cnn_model.add(Dense(50, activation='relu'))
cnn_model.add(Dense(1))

cnn_model.compile(optimizer='adam', loss='mse')

cnn_model.summary()
```



```
cnn_history = cnn_model.fit(x_train, y_train,epochs=100,batch_size=32,validation_split=0.1,verbose=1,shuffle=False)
```