Attention-Aware Deep Learning Framework for Smart Home Energy Forecasting

MSc Research Project

Programme Name: MSc in Data Analytics

Name: Harishwar Yadav Elgamoni

Student ID: x23340576

School of Computing

National College of Ireland

Supervisor: Teerath Kumar Menghwar

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. Harishwar…Yadav…Elgamoni……
……

Student ID: ……x23340576……..
……

Programme: ……MSc…Data Analytics…… **Year:** ……2024-2025…….
Module: ……(Research) Practicum……

Lecturer: ……14-08-2025…….……
Submission Due Date: ……

Project Title: ……. Attention-Aware Deep Learning Framework for Smart Home Energy Forecasting

Word Count: …… **Page Count:** ……7……..……..……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Harishwar……

Date: ……14-09-2025……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only

Signature:	
Date:	
Penalty Applied (if applicable):	

Attention-Aware Deep Learning Framework for Smart Home Energy Forecasting

Name: Harishwar Yadav

ID : x23340576

System Infrastructure Requirements

- **Operating System Compatibility**
    - **Preferred Environment**: Windows 10 or Windows 11 — offers the widest support for development tools and integrated libraries
    - **Alternative Environment**: Ubuntu 20.04 LTS or newer — ideal for opensource development and server-based configurations
- **Processor Specifications**
    - **Minimum**: Multi-core CPU (Intel Core i5 / AMD Ryzen 5 or newer) for basic operations
    - **Recommended**: Quad-core or higher for faster processing, model training, and parallel task execution
- **Memory Requirements**
    - **Minimum**: 16 GB RAM — suitable for standard testing and moderate data handling
    - **Optimal**: 32 GB RAM or more to efficiently process large datasets and run simultaneous experiments
- **Enhanced Performance Option**
    - NVIDIA GPU with CUDA support — enables significantly faster deep learning computations and training processes
- **Storage Capacity**
    - 15–20 GB of available disk space for datasets (raw and processed), model

checkpoints, embeddings, visualization files, and required libraries

Software Setup Guidelines

- **Programming Language**
    - Python 3.9 or later — recommended for best compatibility with modern libraries
- **Key Data Analysis Libraries**
    - pandas — for structured data handling and transformation *(pip install pandas)*
    - numpy — for numerical calculations and array-based processing *(pip install numpy)*
    - matplotlib — for generating static charts and visual outputs *(pip install matplotlib)*
    - seaborn — for advanced and statistical visualizations *(pip install seaborn)*
    - plotly — optional library for interactive and dynamic graphs *(pip install plotly)*
    - scikit-learn — for preprocessing, baseline ML models, and evaluation metrics *(pip install scikit-learn)*
- **Machine Learning & Deep Learning Tools**
    - torch (PyTorch) — primary framework for building, training, and deploying neural network models *(pip install torch)*

1. Purpose & Scope

- Guide for implementing and evaluating energy forecasting models from the *Final Report – Energy*. • Focuses on:
    - Dataset descriptions.
    - Preprocessing steps.
    - Models evaluated (no extra models added).
    - Key results and conclusions.

2. Datasets

2.1 SmartHome Environmental Time-Series Dataset •

Multivariate, minute-level sequential data. •

Features:

- Energy usage ("Appliances").
- Indoor temp/humidity (T1–T9, RH1–RH9).
- Outdoor weather (T_out, RH_out, wind speed, visibility, pressure).
- Date/time/day of week.

- Model used: **T-FormerReg (Time-Series)**.

2.2 Appliance-Level Household Energy Usage Dataset

- Structured, non-sequential tabular data.
- Features:
  - Appliance type, household size, season.
  - Outdoor temperature, timestamp details.
  - Mixed categorical & numerical.
- Model used: **T-FormerReg (Tabular)**.

3. Preprocessing

3.1 Time-Series Dataset

- Missing values → forward fill + interpolation.
- Temporal encoding → sine/cosine for periodicity.
- Normalization → Z-score standardization.
- Patterns:
  - Evening peak (17:00–19:00).
  - Higher consumption at high indoor temp.
  - Mild seasonal variation.

3.2 Tabular Dataset

- Categorical → embeddings.
- Numerical → min–max normalization.
- Time/date → discrete segments (hour blocks, weekday/weekend).
- Stratified train/val/test split by appliance type.

4. Models Evaluated

- **Machine Learning:**
  - XGBoost
  - LightGBM
  - KNN
  - Random Forest (SmartHome)

```python
import matplotlib.pyplot as plt
from lightgbm import LGBMRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np


# Estimators parameter tuning
n_estimators_list = [50, 100, 120, 150]

mse_values = []
final_result = {}

print("LightGBM Estimator Tuning:\n")

for n_est in n_estimators_list:
    model = LGBMRegressor(
        n_estimators=n_est,
        learning_rate=0.001,
        max_depth=3,
        num_leaves=10,
        random_state=42
    )
    model.fit(X_train, y_train)
    preds = model.predict(X_test)

    mse = mean_squared_error(y_test, preds)
    mae = mean_absolute_error(y_test, preds)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, preds)
```

\

Light GBM

- **Deep Learning:**
    - Deep MLP (SmartHome)
    - FFNN (Appliance-level)
    - CNN

```python
# Preprocessing
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled = np.expand_dims(X_scaled, axis=2)  # (samples, features, 1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
y_train_noisy = y_train + np.random.normal(0, 2.0, size=y_train.shape)

# Tuning configurations
configs = [(4, 2), (8, 3), (16, 5)]
mse_scores = []
labels = []

print("Tuning CNN configurations:\n")

for filters, kernel in configs:
    model = Sequential([
        Conv1D(filters=filters, kernel_size=kernel, activation='relu', input_shape=(X_train.shape[1], 1)),
        Flatten(),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    model.fit(X_train, y_train_noisy, epochs=8, batch_size=64, verbose=0)

    y_pred = model.predict(X_train)
```

CNN Model

- **Hybrid (Main Model):**

○ **T-FormerReg (Time-Series)** ○ **T-FormerReg (Tabular)**

```
# Model Architecture
input_layer = Input(shape=(1, X_scaled.shape[2]))
x = transformer_encoder(input_layer)
x = GlobalAveragePooling1D()(x)
x = Dense(64, activation="relu")(x)
x = Dropout(0.1)(x)
output_layer = Dense(1)(x)

model_transformer = Model(inputs=input_layer, outputs=output_layer)

# Compile and train with tracking MSE and MAE
model_transformer.compile(optimizer="adam", loss="mse", metrics=["mae"])

history = model_transformer.fit(X_train, y_train, epochs=15, batch_size=32, validation_split=0.1)

# Get MSE and MAE values from history
mse_values = history.history['loss']
mae_values = history.history['mae']

# Plot MSE and MAE vs Epochs
epochs = range(1, len(mse_values) + 1)

plt.figure(figsize=(12, 6))
```

5. Key Results

5.1 SmartHome Time-Series

- Best: **T-FormerReg** → MSE 2.19, R^2 0.37.
- Outperformed all baselines in capturing temporal patterns.

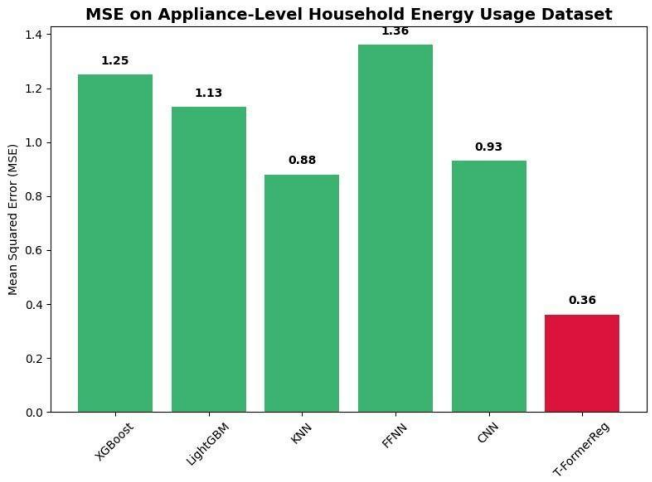
5.2 Appliance-Level Tabular

- Best: **T-FormerReg** → MSE 0.36, R^2 0.74.
- Strong contextual feature learning without heavy computation.

```
Transformer Regression Model:
MSE: 0.361826
MAE: 0.487219
RMSE: 0.601519
R²: 0.742754
```

6. Insights

- ML baselines → solid, but weak temporal/contextual awareness. • Deep learning → better representations, but higher complexity.
- Transformers → best accuracy + efficiency + adaptability.



References

Core Programming and Data Libraries

- https://docs.python.org/3/
- https://numpy.org/doc/
- https://pandas.pydata.org/docs/
- https://scikit-learn.org/stable/documentation.html

Deep Learning Frameworks

- https://pytorch.org/docs/stable/index.html
- https://www.tensorflow.org/api_docs/python/tf/keras

Ensemble Learning & Gradient Boosting

- https://lightgbm.readthedocs.io/en/latest/
- https://xgboost.readthedocs.io/en/stable/

Data Visualization Tools

- https://matplotlib.org/stable/users/index.html
- https://seaborn.pydata.org/
- https://plotly.com/python/

Development & Experimentation Environments

- https://jupyter.org/documentation