Politecnico Di Torino

Master's degree in Data Science and Engineering

Master's Degree Thesis:

Developing a Custom Attention Mechanism for Multivariate Time Series Forecasting

Supervisors
Prof. Paolo Garza
Dr. Lorenzo Bongiovanni

Candidate
Ndjekoua Nzeumi Tito

March 2025

Summary

This thesis work focuses on the development of a new attention mechanism with the goal of tailoring the Transformer architecture to multivariate time series. In particular, two attention mechanisms were compared: the classical one, commonly used in 'vanilla' transformers, and the one we introduced, that we will refer to as 'cross-attention'. The latter is a modification aimed to improve the projection representation in the attention to learn the most critical features for multivariate time series forecasting.Therefore we introduced two main use cases for embeddings to evaluate the performance of cross-attention in both long and short sequences.

In the first scenario, the input sequence consists of embeddings spanning 12 hours, with the goal of predicting the following hour. This experiment allows us to assess how well cross-attention captures long-term dependencies and temporal patterns over extended periods. However, while it may effectively retain information over time, its responsiveness to early events could be improved.

The second use case extends the input sequence to 47 hours, again predicting the subsequent hour. This test helps determine whether our cross-attention mechanism can enhance its understanding of daily cycles and potentially identify weekly trends. The main challenge is to not dilute the attention of transformer to the most relevant points by increasing the sequences length and up the model complexity. Furthermore, we propose a solution to mitigate the capacity of the model to learn to copy patterns in the data rather than to learn dynamics of time series, which is one

of common time-series forecasting issues. This thesis explores a solution based on the state-of-the-art transformer model and the comparison of different Attention mechanisms characterized different computations time and accuracy result in the defined use cases.

Acknowledgements

I would like to begin by expressing my heartfelt gratitude to my parents, Mr. Ndjekoua Pierre and Mrs. Kemajou Eugenie. Their unwavering love and support have been the bedrock of my journey, providing me with the strength and motivation to push forward. Their encouragement, patience, and countless sacrifices have shaped me into the person I am today. I am deeply indebted to them for their constant belief in me.

I am also profoundly grateful to Ing. Fabrizio Dominici for giving me the incredible opportunity to be part of the talented team at LINKS Foundation. His trust and support have been instrumental in allowing me to embark on this enriching experience.

A special note of appreciation goes to Dr. Lorenzo Bongiovanni, whose invaluable guidance and mentorship have played a crucial role throughout my time at LINKS. From the very beginning, Lorenzo not only introduced me to this remarkable opportunity but also provided me with the tools, resources, and insightful advice needed to navigate this challenging yet rewarding project. His dedication and willingness to share his expertise have made a significant impact on my professional growth.

I would also like to extend my sincere thanks to Ing. Tekamte Arsene Bolivard for his technical expertise and invaluable insights. His ability to break down complex concepts and offer clear, thoughtful guidance has been an immense help in deepening my understanding and refining my approach to this project.

Lastly, I am truly appreciative of Professor Paolo Garza for his

instrumental role in connecting me with key figures within the company's HR and executive teams. His proactive efforts and support greatly facilitated my transition into this professional environment and ultimately helped secure my thesis position.

To each of you, I extend my deepest gratitude. Your guidance, encouragement, and belief in me have not only contributed to the success of my academic work but have also profoundly influenced my personal and professional development.

Contents

Acronyms

AI: Artificial intelligence

SA: Self-Attention

ML: Machine Learning

DL: Deep Learning

NN: Neural Network

FFN: Feed Forward Network

CA: Cross Attention

MH-A: Multi-Head-Attention

MTS: Multivariate Time-Series

RNN: Recurrent Neural Network

LSTM: Long Short Term Memory

MSE: Mean Squared Error

List of Tables

List of Figures

1 Introduction

In an era where data-driven solutions are reshaping industries, time series analysis has emerged as a pivotal tool for deriving actionable insights from sequential data. This thesis explores the application of a new kind of attention mechanism, that we call Cross-Attention. This approach aims to enhance the performance of predictive transformer models for time dependent data. Conducted in collaboration with a private non-profit foundation based in Turin, Italy, this research bridges the gap between academic innovation and real-world application, driving product advancements and societal impact.The host organization, renowned for fostering innovation, has a strategic focus on leveraging advanced methodologies to address practical challenges. My work contributes to this vision by exploring a new way to optimize energy consumption monitoring, a critical area in the global push for sustainable energy practices. As a master's student at Politecnico di Torino (PoliTo), I was entrusted with the opportunity to align my academic expertise with the mission of the foundation. This collaboration highlights the symbiotic potential of academia and industry, where theoretical frameworks can be tested and adapted to meet pressing practical demands. The thesis is structured to guide the reader through a comprehensive understanding of the topic. It begins in Section 2 with a review of the literature, offering a critical analysis of existing work on transformers for time series data. Section 5 details the methodology, the design, and implementation of the cross-attention mechanism tailored for energy monitoring. Sub-

sequently, in Section 7, the results and discussion are presented, including their implications for industry and society. Finally, the conclusion and future work are presented in Section 8, where the potential for scaling and adapting this approach to other domains is outlined.

2 Background

2.1 Multivariate Time-Series (MTS)Forecasting

A time series is a sequence of data points collected or recorded at regular time intervals. It reflects how a variable evolves over time.

A MTS refers to a dataset where multiple variables are recorded over time, capturing their evolution simultaneously. Unlike Univariate Time Series, which tracks a fluctuations in a single variable over time, multivariate data involve several interdependent variables that may influence each other.This approach is particularly useful when analyzing complex systems in which multiple factors interact, such as financial markets, climate patterns, or industrial processes. By considering multiple variables together, multivariate time series models can uncover deeper insights, improve predictive accuracy, and reveal relationships that wouldn't be apparent when analyzing each variable separately.

2.2 Why Forecasting Time Series

The process of predicting future events,trends, or outcomes based on historical data, patterns,and analysis is called forecasting. It helps in making informed decisions by estimating what might happen next. Time-series forecasting is essential to predict future values based on historical data. It is crucial for planning, decision-making, and understanding trends and patterns in various domains. Forecasting helps mitigate risks, allocate resources effectively, and make informed decisions. Here are some phenomena

that describe time-series :

- Stock market prices
- Weather patterns
- Electricity consumption
- Economic indicators (e.g., GDP, inflation rates)
- Patient vital signs in healthcare (e.g., heart rate, blood pressure)

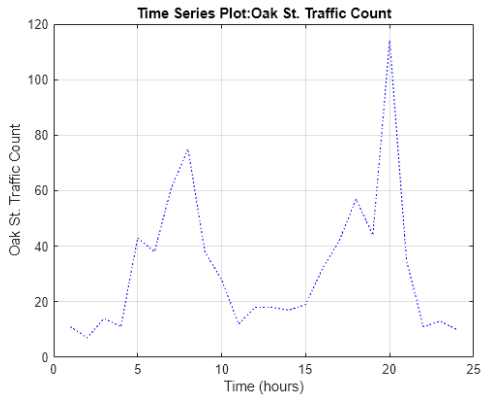

Figure 2.1: Data registered from Traffic Count .

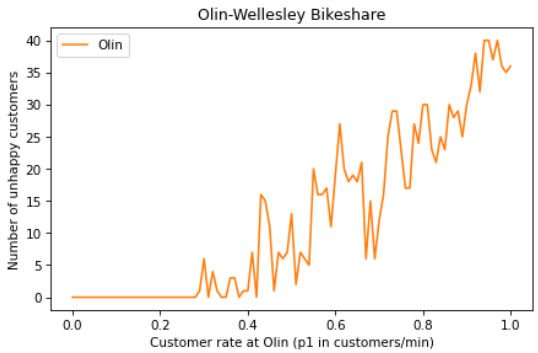

Figure 2.2: Bike-share Customer satisfaction Data.

2.3 The Forecasting of Multivariate Time-Series and leveraging Machine Learning

MTS Forecasting involves predicting future values of multiple related time-series variables simultaneously. Unlike univariate forecasting, it considers the interdependencies among variables to improve predictive accuracy.Here where Machine Learning models come into play:

- Enhanced Prediction Accuracy: By leveraging relationships between variables, multivariate forecasting produces more reliable predictions.

- Dynamic Systems Modeling: AI uses multivariate time-series data to analyze complex systems where variables influence each other.
- Real-Time Decision Making: Applications in fields like healthcare and finance rely on AI models to process multivariate data and provide instant insights.
- Enables Advanced Models: Techniques like deep learning (e.g., LSTMs, transformers) can process multivariate data for superior performance in forecasting tasks.

2.4 Challenge of Multivariate Time-Series Forecasting

MTS Forecasting presents unique challenges due to the complexity of handling multiple interdependent variables. Identifying and modeling the relationships among these variables is difficult, especially when the data exhibits non-linear patterns, missing values, or irregular time intervals. The high dimensionality of multivariate data increases computational demands and the risk of overfitting, requiring sophisticated techniques such as deep learning with transformer architectures to manage these issues effectively. Additionally, external factors such as noise, environmental changes (e.g., climate, temperature), or unexpected events can significantly affect forecasting accuracy, complicating its applications in dynamic real-world scenarios such as healthcare, finance, and energy management.

2.5 What is Transformer

The Transformer is a deep learning architecture introduced by Vaswani and al. [1]that processes sequential data using self-attention mechanisms instead of traditional recurrence-based approaches.

Unlike RNNs and LSTMs, which handle sequences step by step, Transformers eliminate this constraint, enabling parallel computation and making it easier to capture long-range dependencies. Their architecture is built around **MH-A, Positional Encoding, and Feedforward Layers,** which work together to efficiently model complex patterns in data. Thanks to these advantages, Transformers have become a cornerstone of modern deep learning, powering breakthroughs in natural language processing, computer vision, and beyond.

2.6 Attention Mechanism

Central to Transformer, is the self-attention module. It can be viewed as a fully connected layer with weights that are dynamically generated based on the pairwise similarity of input patterns. As a result, it shares the same maximum path length as fully connected layers, but with a much less number of parameters, making it suitable like hierarchical recurrent attention networks [2] for modeling long-term dependencies.

2.6.1 self-Attention (SA)

The input matrix X is projected into three distinct representations: queries (Q), keys (K), and values (V), using learned weight matrices W^Q, W^K, and W^V. The transformation is defined as follows:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V \tag{2.1}$$

The matrices W^Q, W^K, and W^V are learned during training through backpropagation. These transformations allow the model to extract optimal representations, enabling it to effectively capture relationships between tokens.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

Where:

- Q(query): What we are looking for.
- K(keys): What we are comparing to.
- V(values): What we use to construct the output.
- d_k : is the dimension of the keys and queries in each attention head.

Sometimes called intra-attention `SA` is a cornerstone of the transformer architecture, a model widely used in natural language processing, time series analysis, and other domains. At its core, attention allows the model to focus selectively on relevant parts of the input data while processing it. This is particularly useful for handling sequential Data such as time series, where dependencies across different time steps are essential. In time series, attention helps:

→ Capture long-term dependencies by focusing on all time steps simultaneously.

→ Highlight key time points that contribute most to future predictions.

→ Reduce the reliance on fixed-length representations, improving model flexibility.

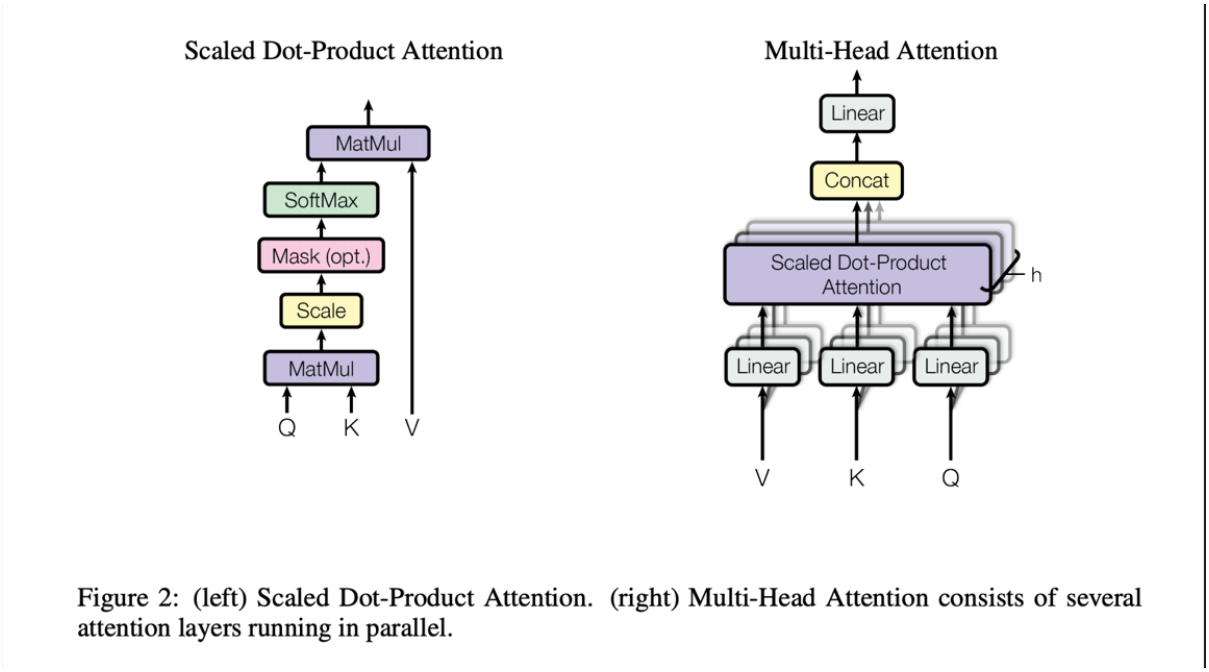

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Figure 2.3: Attention Mechanism.

2.6.2 Multi-Head-Attention (MH-A)

MH-A (Fig. 2.3)runs multiple self-attention mechanisms in parallel to improve learning. The number of heads h is greater than 1, allowing the Transformer to focus on different parts of the input representation simultaneously. Each head learns a different projection of Q, K, V using separate weight matrices:

$$Q_i = XW_i^Q, \quad K_i = XW_i^K, \quad V_i = XW_i^V, \quad \text{for } i = 1, \ldots, h. \tag{2.3}$$

Each head captures unique relationships or patterns within the data, enabling the model to learn richer representations compared to a single-head attention mechanism. The outputs of all heads are then concatenated and projected using a final weight matrix W^O:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O. \tag{2.4}$$

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V) \tag{2.5}$$

3 Problematic addressed

3.1 Motivation

This study aims to improve the way the embedding space of the time- series input features is handled by vanilla transformer. Traditional implementations of the Transformer process all embeddings uniformly, without distinguishing between different types of features inherent in multivariate time series data. Specifically, in multivariate settings, data can often be decomposed into two fundamental categories:

1. `**Value features**:` These capture the characteristics of individual variables (e.g., temperature, pressure, or sales volume) across the dataset.

2. `**Time features**:` represent a temporal "label" of each point in the time series that is important because it relates to external facts that influence the trend of the energy consumption time series.

Current attention mechanisms in the Transformer do not inherently differentiate between these feature types. As a result, they fail to treat value-feature attention and time-feature attention separately, which can lead to suboptimal representation learning. Even in cases where features are explicitly separated, it is common practice to concatenate or sum the attention of 'value-features' (e.g., weather parameters like temperature, wind speed) and 'time-features' (e.g., day of the week, day of the year, month) as input to the model. However, existing methods do not sufficiently allow time features to influence value features. This oversight potentially

limits the model's ability to fully capture and utilize the intricate relationships within multivariate time series data.

One significant drawback to take into account when working on this project is the self-attention mechanism in a standard transformer, which has a time and memory complexity of $O(N^2)$. Our aim is to prevent this from causing any disruptions in the cross-attention mechanism, where N denotes the length of the input time series. This poses a serious computational challenge when processing long sequences—such as those spanning days, weeks, or even months—since the growing sequence length drastically increases the dimensionality of the embeddings.

3.2 Solution impact for the organization

This challenge is particularly relevant in the context of the organization where this research is being conducted. The enterprise heavily relies on accurate and interpretable time series forecasting for mission-critical applications such as demand prediction, resource optimization, and anomaly detection. The limitations of existing methods in capturing nuanced interactions between value and time features have become a bottleneck in achieving the desired levels of forecasting accuracy and robustness.

To address this issue, the objective of this thesis is to design and develop a custom attention mechanism that explicitly distinguishes and independently processes value-feature attention and time-feature attention. The proposed approach aims to enhance the model's ability to learn specialized attention patterns for each feature type, enabling a more precise and context-aware representation of multivariate time series data. By subsequently combining these distinct attention outputs, the model can leverage the strengths of both perspectives, leading to improved forecasting performance.At the same time, while this issue is acknowledged,

it is not directly addressed in detail in this study.

4 Datasets

In this section, we present the Dataset used in this study. The dataset contains measured data on an energy system to predict its energy consumption. The data were collected from various sources, from sensors and recording systems, and cover a coherent time period. The datas spans from 2016 to 2018, recorded at an hourly frequency.This Dataset is essential to train and evaluate the Transformer model designed as part of this research.

4.1 Features Description

The dataset consists of `17,420 observations` and `8 variables`, described in the table below:

Table 4.1: Description of the Dataset Variables

Variable	**Description**	**Type**	**Unit**
`date`	Timestamp of the measurements	Date/Time	-
`HUFL`	High Upper Flow Level	Numeric	kWh
`HULL`	High Upper Low Level	Numeric	kWh
`MUFL`	Middle Upper Flow Level	Numeric	kWh
`MULL`	Middle Upper Low Level	Numeric	kWh
`LUFL`	Low Upper Flow Level	Numeric	kWh
`LULL`	Low Upper Low Level	Numeric	kWh
`OT`	Outdoor temperature recorded during measurement	Numeric	°C

The regression model takes the past values of all these variables as explanatory parameters in a sequence and predicts their values for the next hour

4.1.1 Exploratory Data Analysis

The dataset has been analyzed to provide key descriptive statistics for the numerical variables, as shown in the table below:

Table 4.2: Descriptive Statistics of Key Variables

Variable	Min	Max	Mean	Std Dev
HUFL	-22.71	23.64	7.38	7.07
HULL	-4.76	10.11	2.24	2.04
MUFL	-25.09	17.34	4.30	6.83
MULL	-5.93	7.75	0.88	1.81
LUFL	-1.19	8.50	3.07	1.16
LULL	-1.37	3.05	0.86	0.60
OT	-4.08	46.01	13.32	8.57

* The outdoor temperature values (OT) vary from -4.08 °C to 46.01 °C, which illustrates a wide climatic range.
* The levels of energy consumption (for example, HUFL, HULL, MUFL) show significant variations, suggesting an interesting dynamic for modeling.

In (Fig.4.2), the data presented for the model generally exhibit the characteristics of a time series, with a clear and consistent seasonality observable across each input parameter. Additionally, an identifiable trend can be seen in the curves, further reinforcing the reliability and stability of the sensor recordings.

4.2 Data Preprocessing

Here, is the description of the preprocessing steps applied to the dataset used in this study. The dataset is designed to prepare the data for training and testing a model based on Cross Attention mechanisms an traditional Attention with time series forecasting. Before training the model, the following preprocessing steps were performed:

- Handling of missing values, if present.

- Outlier detection and treatment for energy usage and temperature variables.
- Normalization of variables to ensure compatibility with the transformer model.

4.2.1 Time Features Representation

It is essential to encode temporal components appropriately to help the model learn cyclic dependencies effectively. On the matter Recent advancements like Informer [3] address long sequence dependencies efficiently.In fact imestamps in the format (yyyy-mm-dd-h) represent cyclical temporal data:
- **Hour**: 11 PM and midnight are numerically distant when encoded as integers, yet they are conceptually adjacent in time.
- **Day**: Monday and Sunday should be considered neighbors in numerical representation.
- **Month**: December and January must be treated as close, particularly in the context of winter. Since our transformer model is designed to capture intricate relationships within data, it is essential to integrate this cyclical structure. This is because:

- The self-attention mechanism needs to recognize that certain hours or days are naturally correlated.
- Proper encoding prevents misleading patterns that might arise from an improper numerical representation of time.

Cyclic Encoding of Hour:

A naive approach to encoding time-related features would be to use raw hour values (e.g., 0 to 23 for hours of the day). However, this representation is problematic for machine learning models because it introduces artificial discontinuities. For instance, in a simple numerical encoding, hour 23 and hour 0 appear distant (numerically 23 units apart), even though they are actually adjacent in the cyclic nature of time.

To address this issue, we employ cyclic encoding using sine and cosine transformations:

$$\text{Hour}_{\text{sin}} = \sin\left(2\pi\frac{\text{Hour}}{24}\right) \tag{4.1}$$

$$\text{Hour}_{\text{cos}} = \cos\left(2\pi\frac{\text{Hour}}{24}\right) \tag{4.2}$$

This transformation maps each hour onto a continuous unit circle, preserving the cyclic nature of time. Consequently:

- The Euclidean distance between consecutive hours remains consistent, avoiding abrupt jumps between 23 and 0.
- The model can better capture periodic patterns in energy surveillance sensor data, improving predictive accuracy.
- Feature interactions become more meaningful, allowing the model to generalize time-dependent patterns more effectively.

Cyclical Encoding of Months:

To effectively represent the cyclical nature of months in numerical models, we use sinusoidal transformations:

$$\text{month\_sin} = \sin\left(2\pi \times \frac{\text{month}}{12}\right) \tag{4.3}$$

$$\text{month\_cos} = \cos\left(2\pi \times \frac{\text{month}}{12}\right) \tag{4.4}$$

One of the key advantages of this encoding is that it effectively captures seasonal variations in energy consumption. For instance, heating demand surges during winter months, while air conditioning usage spikes in summer.

A major limitation of treating months as simple numerical values (e.g., 1 for January, 12 for December) is that it fails to reflect

their cyclical nature. In reality, December and January are consecutive months, yet a naive numerical encoding would place them at opposite ends of the scale (1 and 12), making them appear unrelated. By applying sine and cosine transformations, we ensure that adjacent months are represented in a way that preserves their natural continuity.

Cyclical Encoding of Days:

To properly account for the cyclical nature of weekdays in numerical models, we use trigonometric transformations as follows:

$$\text{day\_sin} = \sin\left(2\pi \times \frac{\text{day\_of\_week}}{7}\right) \tag{4.5}$$

$$\text{day\_cos} = \cos\left(2\pi \times \frac{\text{day\_of\_week}}{7}\right) \tag{4.6}$$

Energy consumption patterns vary significantly between weekdays and weekends. For example, office buildings tend to use far less electricity on Saturdays and Sundays compared to workdays.

A common pitfall of treating days of the week as simple categorical numbers (e.g., 1 for Monday, 7 for Sunday) is that it ignores their cyclic structure. This results in an artificial gap between Friday and Monday, despite these days being consecutive in reality. By leveraging sine and cosine transformations, we ensure that the representation correctly reflects their natural continuity, making adjacent days appear close in the feature space.

By integrating cyclic encoding into our regression transformer model, we ensure that time-based dependencies are well-preserved, ultimately enhancing the forecasting performance for energy monitoring applications.

4.2.2 Feature Engineering: Capturing Non-Linear Relationships

Since we anticipated non-linear relationships, we applied transformations to the environmental variables to better capture these

effects. Specifically, we created derived features by implementing quadratic terms and interaction effects.

Quadratic Terms : To model potential quadratic patterns in the data, we introduced the squared term for the outdoor temperature:

$$OT^2 = (\text{Outdoor Temperature})^2 \tag{4.7}$$

This allows the model to capture non-linear variations in temperature effects, which may not be adequately represented by a simple linear term.

Interaction Features: To account for the combined influence of multiple environmental factors, we created interaction terms. One such interaction is between outdoor temperature (OT) and another variable, LUFL:

$$\text{Interaction Term} = OT \times LUFL \tag{4.8}$$

4.3 Splitting the Dataset

The dataset is divided (Fig.4.1) into training, validation, and testing sets with proportions of 65%, 15%, and 20%, respectively. The division ensures that the temporal order is maintained.

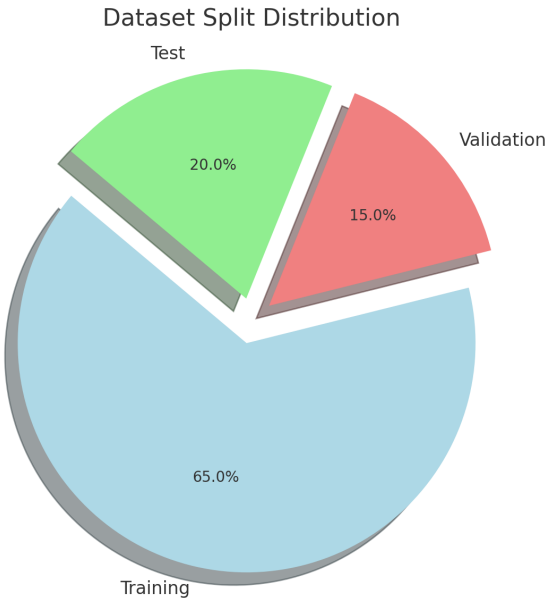

Figure 4.1: split dataset.

4.4 Normalization with RobustScaler

To eliminate discrepancies in scale and variance across the various features of the multivariate time series, adjustments are made with Robust-scaler to ensure consistency and uniformity in their representation. The numeric columns (excluding the Datetime column) are normalized using `RobustScaler`. The training data is used to fit the scaler, which is then applied for both validation and testing data. The transformed data is recombined with the datetime column for sequence generation. The RobustScaler transforms the data using the following formula:

$$X_{\text{scaled}} = \frac{X - \text{median}(X)}{\text{IQR}(X)} \tag{4.9}$$

where:

- X is the original value of the variable,
- $\text{median}(X)$ is the median of the data,
- $\text{IQR}(X)$ (Interquartile Range) is defined as:

$$\mathrm{IQR}(X) = Q_3 - Q_1 \tag{4.10}$$

which represents the interquartile range.

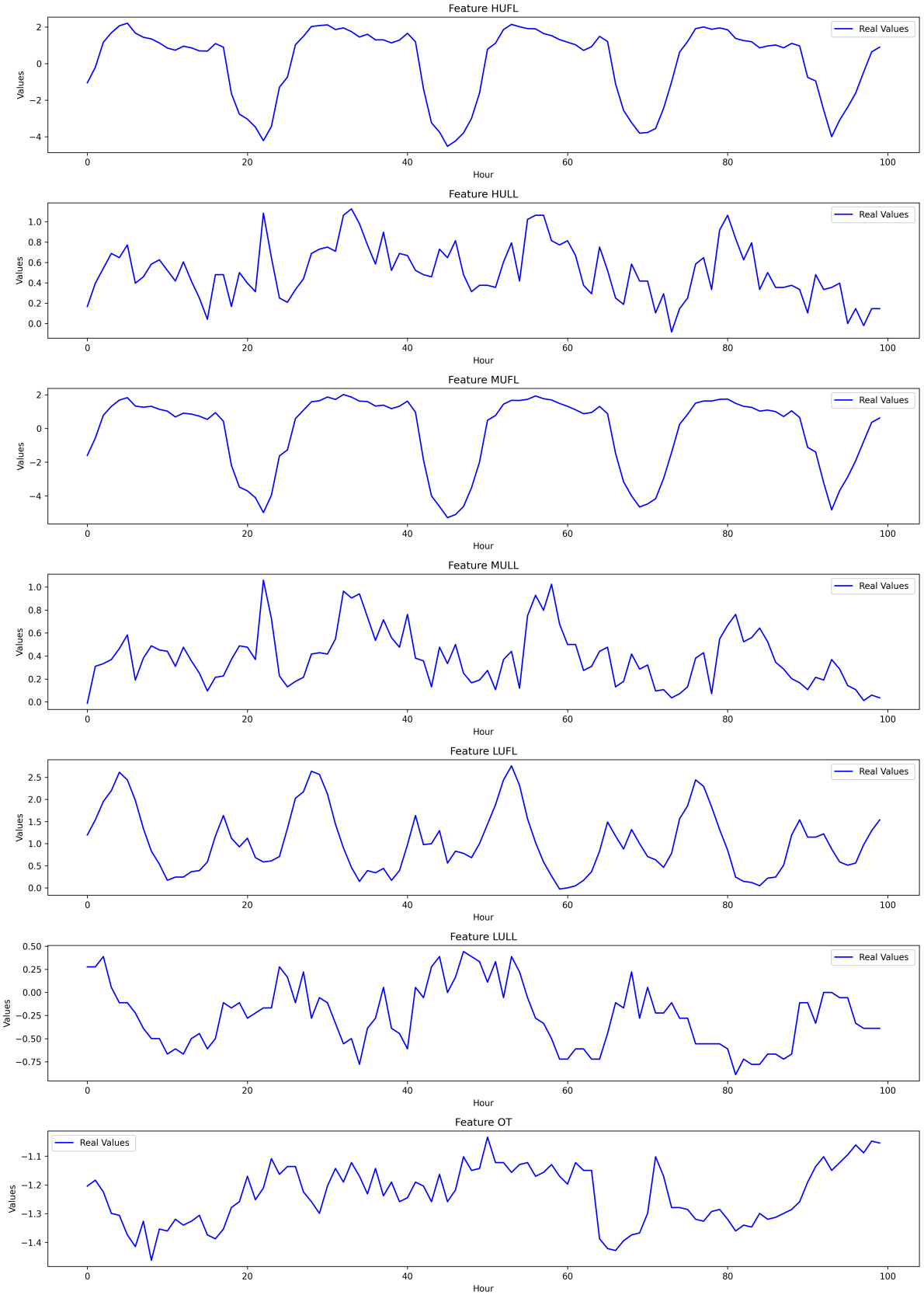

Figure 4.2: Tuned Data for Training.

5 Methods and Materials

In this section, we are going to explore methods and tools used to perform training and test. All task were performed on the same dataset using different sequence data and embedding. First of all, let us talk about the data sequences we made for each task, then we explore the pipeline used to make the predictions.

5.1 Cross Attention design

The figure (Fig.5.2)illustrates the data pipeline, with the Cross-Transformer playing a central role in the process. In the following sections, we will break down each transformation that occurs at different nodes within the pipeline, explaining them in detail.

5.1.1 Sequence Construction

For time series forecasting, the model requires structured sequential inputs, meaning each sample must include a historical window of past observations to predict future values. This is implemented using a sliding window technique in a custom function.

Generating the sequences:

- A sequence with length input(seq_length) consecutive time steps (corresponding to 23,12,47 hours) is extracted as the input window.
- The immediate next time step (1 hour ahead) is designated as the target output for prediction.

- This process is repeated across the dataset, resulting in overlapping sequences that allow the model to learn temporal dependencies effectively.

Mathematically, for a given starting index i, the input and target values are defined as follows:

$$X_i = \{x_i, x_{i+1}, ..., x_{i+seq_{length}}\} \tag{5.1}$$

$$Y_i = x_{i+seq_{length}+1} \tag{5.2}$$

where X_i represents the input sequence and Y_i corresponds to the expected output.

Each sequence-target pair is then converted into a PyTorch tensor, ensuring efficient computation and compatibility with deep learning models.

Creating DataLoaders:

To facilitate efficient batch processing, the generated sequences are organized into PyTorch DataLoaders. These enable:

- *Shuffling* of training data to enhance generalization and prevent overfitting.
- *Sequential processing* of validation and test data to preserve temporal order.

Each batch consists of 32 samples, striking a balance between computational efficiency and effective learning.

5.1.2 Embedding Layers

The embedding process (Fig.5.1)is designed with two distinct components:

1. **Temporal Embedding: Extracting Time-Based Features**
2. **Feature Embedding: Transforming Other Input Variables**

Temporal Embedding: Capturing Time-Dependent Patterns

The first stage of the embedding mechanism is dedicated to processing temporal information from the dataset's timestamp. The key steps in this process are as follows:

- *Isolating the DateTime Component:* The timestamp values, are extracted separately from the rest of the dataset.
- *Conversion to DateTime Format:* Since numerical timestamps are not inherently meaningful, they are reshaped and transformed into a standard Pandas DateTime format. This conversion enables efficient feature extraction.
- *Deriving Temporal Features:* Leveraging the Gluonts library, a variety of time-related characteristics—such as the hour of the day and the day of the week—are computed. These features allow the model to identify periodic trends within the data.
- *Linear Projection:* Once extracted, these temporal attributes are processed through a fully connected layer. This transformation maps the features into a higher-dimensional space of size d_{model}, ensuring that meaningful temporal representations are learned.

Feature Embedding: Processing Other Input Variables

The second stage of the embedding process involves encoding the remaining input features, which are distinct from the temporal component.

- *Extracting the Features:* Apart from the timestamp values, the remaining attributes of the input sequence consist of numerical values that characterize the dataset. These features are isolated for independent processing.

- *Linear Transformation:* To bring these attributes into the same representational space as the temporal features, they are passed through a separate fully connected layer. This operation projects them into a d_{model}-dimensional space, enabling consistent integration with the temporal embeddings.

By keeping the two embeddings separate, the model gains flexibility in processing temporal and feature-based information independently, which can be particularly beneficial when applying mechanisms such as cross-attention.

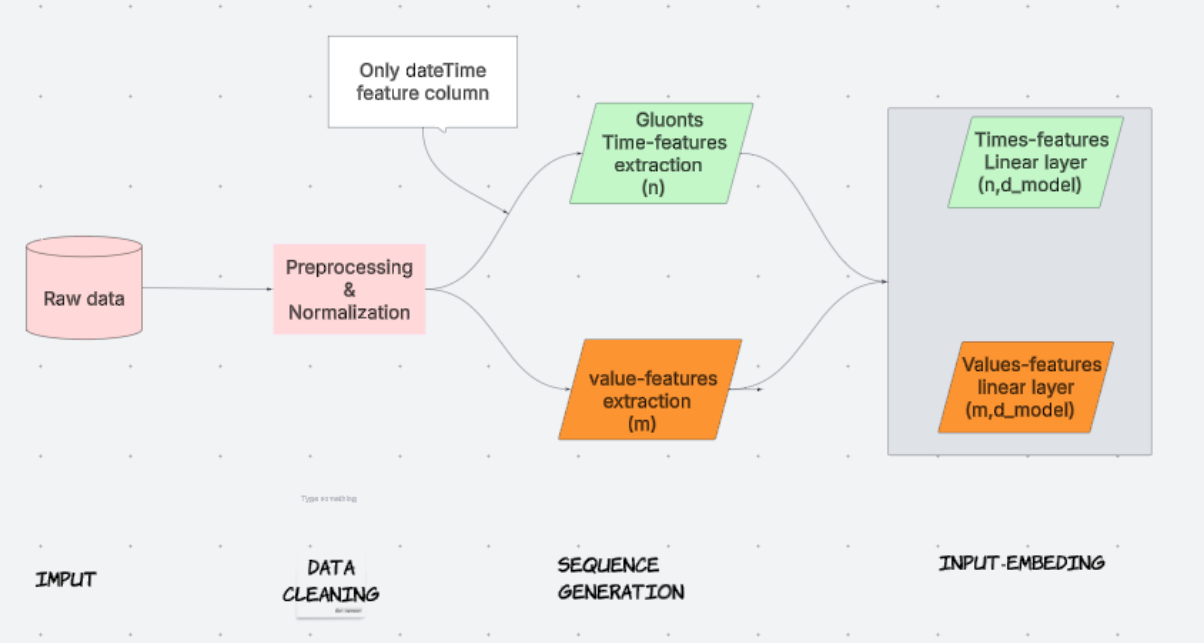

Figure 5.1: Embedding pipeline's .

5.1.3 Positional Encoding

After input embedding separation, both temporal and numerical features embedding are encoded positionally, although timestamps benefit significantly more from positional encoding compared to numerical embedding. WaveNet [4] introduced causal convolutions to capture long-term dependencies in sequential data, which inspired positional encoding in Transformers.Although WaveNet is designed for audio, the idea of efficiently capturing long-term dependencies without using RNNs inspired Transformers and their

positional encodings. The objective is to incorporate positional information into the input embeddings using sinusoidal and cosine functions, enabling Transformers to distinguish the positions of elements within a sequence. It helps capture the temporal structure inherent in the information. By leveraging sine and cosine functions, the model can effectively generalize to sequences longer than those encountered during training.

5.1.4 Keys,Query and Values Projection Layers

The Projection Layers play a pivotal role in reshaping the initial inputs into representations that are well-suited for the multi-head attention mechanism. These layers leverage fully connected linear transformations to learn projections that map the input data into a higher-dimensional space before attention is applied. Each projection takes an input vector of fixed dimension and transforms it into a representation whose size corresponds to the input dimension multiplied by the number of attention heads. This step is crucial for enabling the model to process information from multiple perspectives simultaneously and allows parallel computation across multiple attention heads, improving efficiency and expressiveness.

- **Value Self-Attention Projections:**

$$Q_v = X_v W_Q^v, \quad K_v = X_v W_K^v, \quad V_v = X_v W_V^v \tag{5.3}$$

- **Time-to-Value Attention Projections:**

$$Q_t = X_t W_Q^t, \quad K_t = X_t W_k^t, \quad V_t = X_t W_v^t \tag{5.4}$$

- **special projection:** An extra projection, denoted as $W_{\mathrm{k}'}^v$, is introduced to capture transversal information, further enriching the model's capacity to extract relevant features across different contexts.

$$K_v' = X_v W_{\mathrm{K}'}^v \tag{5.5}$$

Note: W_K^v and W_{k}^v, are two different Key projections of X_v.

5.1.5 Compute Scores Attentions

- **Value Self-Attention** ($scores_v$)**:** This is Attention between the representations of the same modality and learn the interdependence between values in the same sequence. By applying the (V_v) projection, the context is derived

$$scores_v = \text{softmax}\left(\frac{Q_v K_v^\top}{\sqrt{d_k}}\right) \tag{5.6}$$

- **Time-to-Value Attention** ($scores_{tv}$)**:** This represents the cross-attention between different modalities and captures the interdependence between timestamps and values. A new context is subsequently formed through the (V_v) projection.

$$scores_{t_v} = \text{softmax}\left(\frac{Q_t {K'_v}^\top}{\sqrt{d_k}}\right) \tag{5.7}$$

- **Time-to-Time Attention** ($scores_{tt}$)**:** This is Attention between the representations of the same modality and captures the relationships between different timestamps in the same sequence.The (V_t) projection is then used to construct the context.

$$Ascores_{tt} = \text{softmax}\left(\frac{Q_t K_t^\top}{\sqrt{d_k}}\right) \tag{5.8}$$

5.1.6 Transformer Outputs

After computing the dot-scaled product scores and applying the weights to the values (V), each context score head generates its own representation. These individual outputs are then concatenated to form a unified representation. Finally, a linear transformation is applied to reduce the dimensionality, producing the final output. This step effectively integrates information from all context heads,

ensuring a well-structured output that is optimized for the next stages of the model

- **Value Embedding:** Before producing the final output from the Transformer, we apply a weighted combination of the contexts generated by cross-attention to obtain a more refined and well-balanced representation.

$$Em_{\text{value}} = Scores_v * V_v + Scores_{tv} * V_v$$

- **Time Embedding:**

$$Em_{\text{time}} = Scores_{tt} * V_t$$

Benefits of This Approach

- **Customized Representations:** Separate projections (Kv and $K'v$) allow learning different aspects of value-features.
- **It has a time decoder of personal attention:**it is a mechanism that allows the model to focus on key moments in time to improve its forecasts.

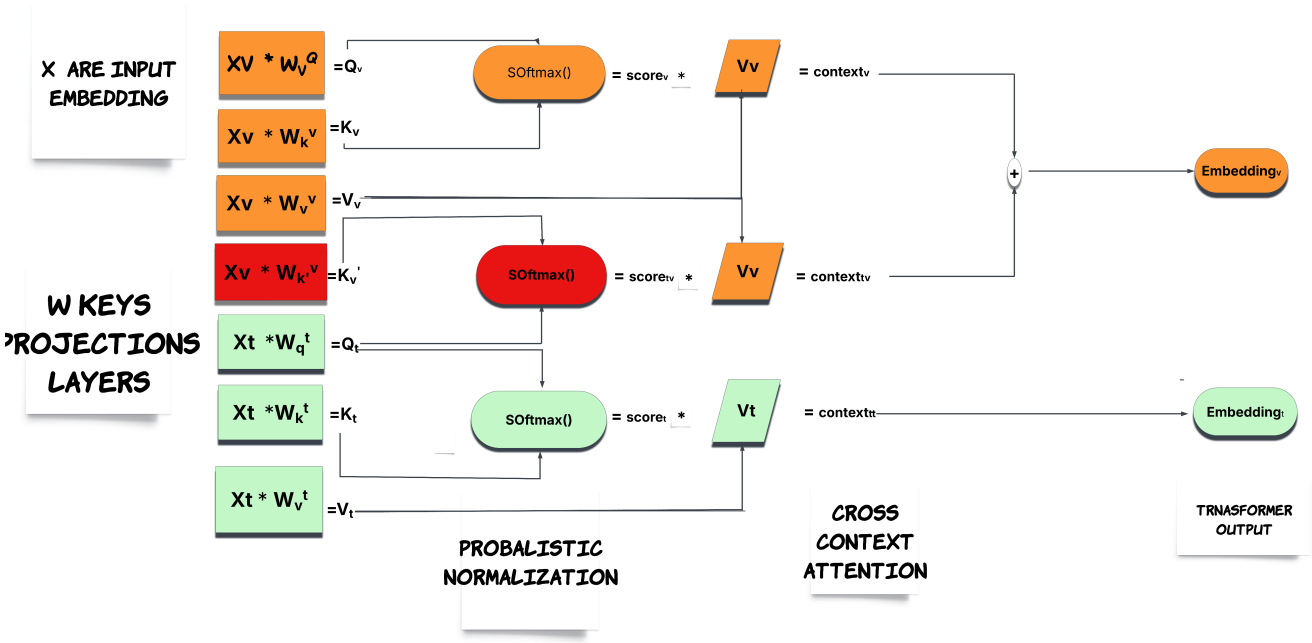

Figure 5.2: Cross Attention Mechanism.

5.2 Custom Loss Function

We decided to design a custom loss function(Fig.5.3) integrates the standard MSE error with an additional penalty designed to discourage the model from consistently predicting the last observed value.

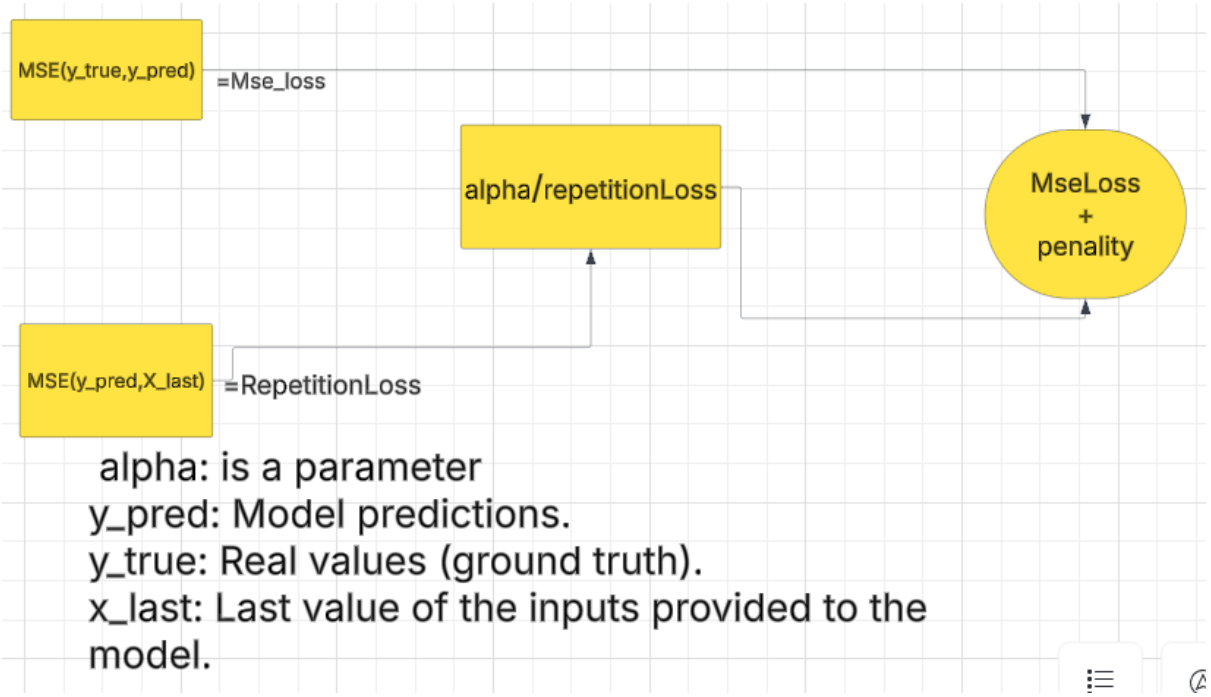

Figure 5.3: custom loss.

5.2.1 Mean Squared Error (MSE)

The primary loss component employed in the model is the Mean Squared Error (MSE), mathematically defined as follows:

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_{\text{pred},i} - y_{\text{true},i})^2 \tag{5.9}$$

where:

- $y_{\text{pred},i}$ represents the predicted value for the i-th time step.
- $y_{\text{true},i}$ denotes the corresponding ground truth value.
- N signifies the total number of elements in the prediction window.

This function measures the average squared difference between predicted and actual values, penalizing larger deviations more heavily.

5.2.2 Regularization Term: Preventing Predictive Stagnation

An additional penalty term is incorporated to mitigate the model's tendency to generate predictions excessively close to the last observed input point, x_{last}. This term is formulated as:

$$\mathcal{L}_{\text{penalty}} = \frac{1}{N} \sum_{i=1}^{N} (y_{\text{pred},i} - x_{\text{last},i})^2 \tag{5.10}$$

where $x_{\text{last},i}$ represents the final input value prior to forecasting. The intuition behind this regularization is to encourage the model to learn meaningful temporal patterns rather than simply extrapolating the last known value, a common pitfall in time series forecasting.

5.2.3 Final Loss Function Formulation

The overall loss function integrates the standard MSE loss with the additional penalty term, weighted by a learnable parameter α:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{MSE} + \frac{\alpha}{\mathcal{L}_{\text{penalty}}} \tag{5.11}$$

Here, α is a trainable parameter that dynamically adjusts the weight of the penalty term during training.

5.2.4 Impact of the Additional Penalty on Time Series Forecasting

The inclusion of $\mathcal{L}_{\text{penalty}}$ serves as a corrective mechanism to discourage the model from generating overly conservative forecasts that merely replicate past observations. Without this penalty, the

model might converge to a trivial solution where predictions remain excessively close to x_{last}, particularly in cases of noisy or slow-moving time series.

By inversely scaling the penalty term, the loss function strongly penalizes scenarios where the predicted values are nearly identical to the last observed input. This encourages the model to capture actual temporal dynamics rather than defaulting to a naive persistence model. However, it is crucial to address potential numerical instabilities arising from small denominator values in $\frac{\alpha}{\mathcal{L}_{\text{penalty}}}$. To mitigate this issue, a small constant ϵ is typically added, yielding a revised formulation:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{MSE} + \frac{\alpha}{\mathcal{L}_{\text{penalty}} + \epsilon} \tag{5.12}$$

where ϵ is a small positive value (e.g., 10^{-6}) to prevent division by zero and avoid numerical instability.

5.3 Algorithms and Models

Deep Transformer models have shown promising results in time series forecasting[5], surpassing alternative neural models such as CNNs and RNNs.Therefore, Transformer-based models have the potential to model complex dynamics of time series data that are challenging for sequence models.

5.3.1 Vanilla Transformer

As noted in the Introduction, we will analyze two separate models to evaluate the effectiveness of our redesigned Cross-Attention mechanism. The first model(Fig.5.4) is based on the Vanilla Transformer architecture, which is widely utilized across various machine[6] learning tasks due to its strong capability to capture long-range dependencies and interactions within sequential data[7]. This makes it particularly well-suited for time series modeling.

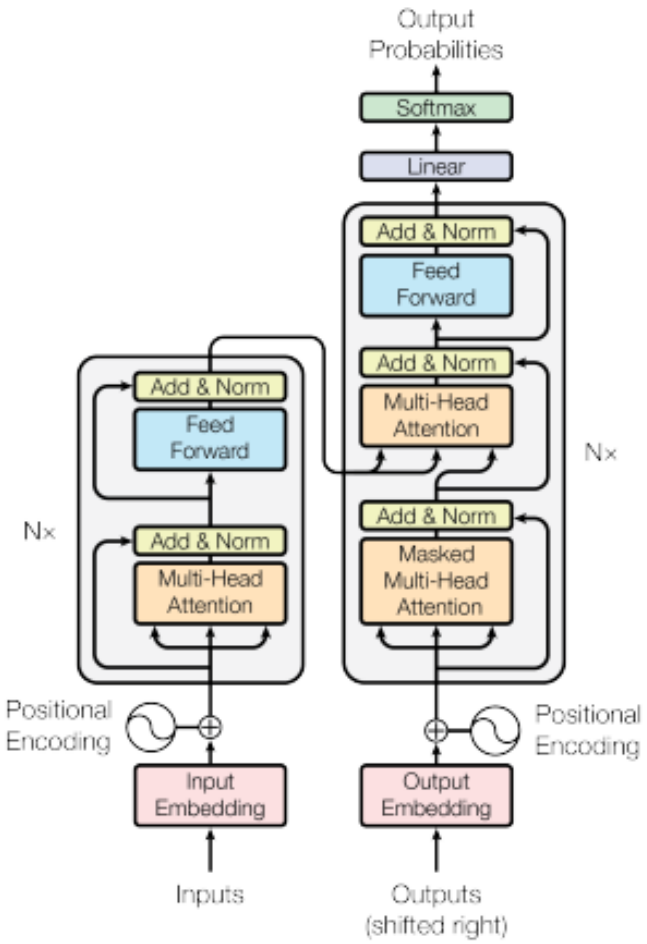

Figure 5.4: Vanilla Transformer.

5.3.2 Custom Transformer:Module Level Variant

In their study, Park et al. [8] demonstrated the effectiveness of Transformer-based models for electricity demand forecasting by proposing a modified multi-head Transformer model.

The Module Level Variant Transformer (Fig.5.5) retains the core architecture of the Vanilla Transformer but introduces structural modifications aimed at enhancing efficiency and capturing long-range dependencies in time series forecasting. Unlike the standard Transformer, which applies uniform attention across the entire sequence, this variant includes:

- A modular attention mechanism that distinguishes between short-term and long-term dependencies [5].

- Optimized attention mechanisms such as Sparse Attention [3] and Grouped Query Attention to improve computational efficiency [1].
- Feature separation between temporal variables and energy-related data, improving forecasting accuracy [8].

Recent studies have demonstrated the effectiveness of such adaptations in time-series forecasting [9], particularly in energy consumption prediction [8].

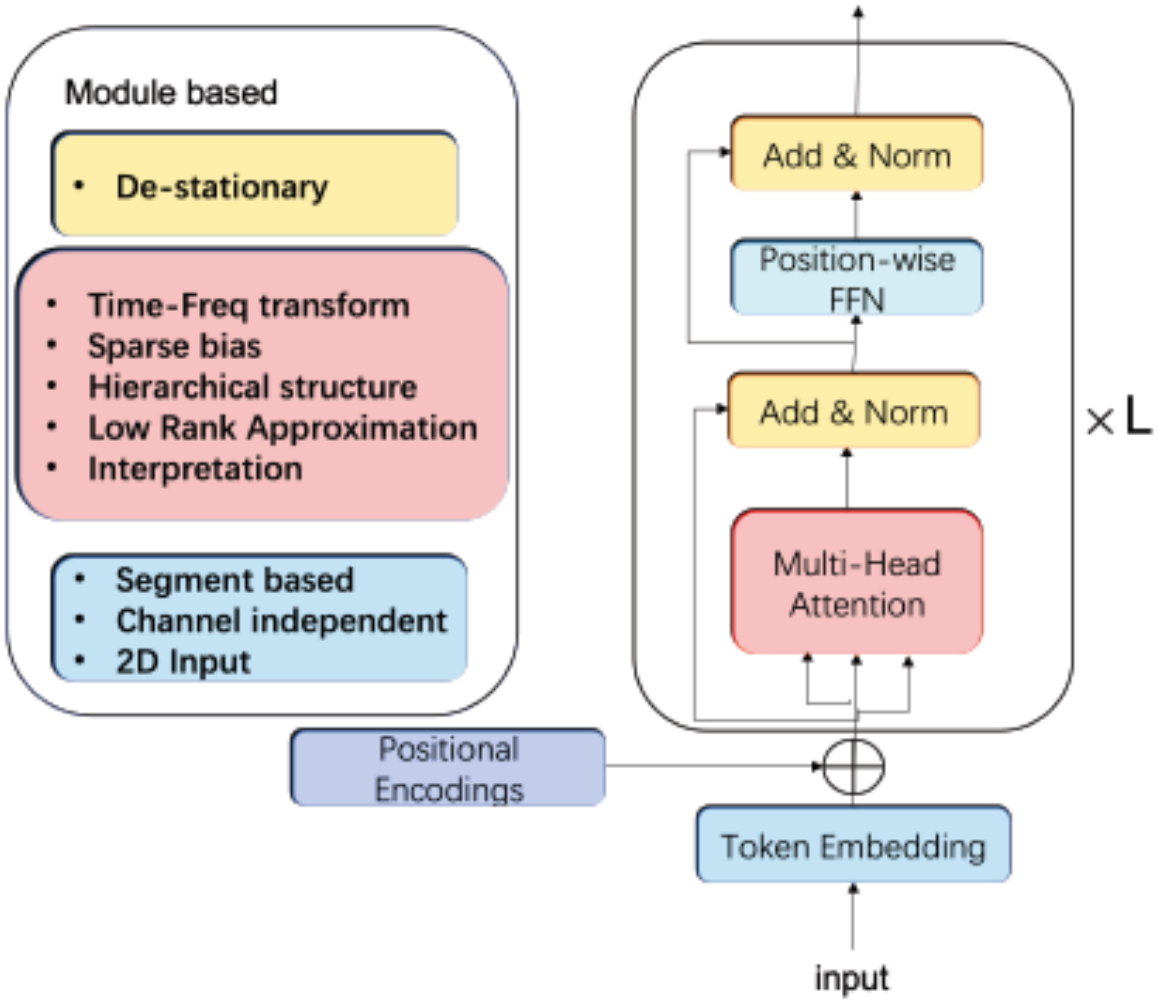

Figure 5.5: Module level variant Transformer.

6 Experiments

6.1 Vanilla Transformer And Custom Transformer

In this experiment that focused on the vanilla Transformer and the cross attention Transformer, we trained both models using the dataset and the data processing described in Section 4. The objective is to evaluate its performance and assess effectiveness of the cross attention architecture compared with the standard self attention.

6.2 Sequence Lenght Test

6.2.1 Short Sequence Test

The model was designed to forecast the values feature of the next hour using daily classification sequence inputs, initially with a 12-hour window and later expanded to 23 hours. The primary objective is to assess whether the model effectively captures rapid, localized variations.

6.2.2 Long Sequence Test

Additionally, the new model's performance was evaluated in an extended prediction scenario spanning two days. Specifically, we tested a 47-hour input sequence to predict one hour ahead, allowing us to analyze how previous days influence the model's forecasting accuracy.

6.3 Model Initialization and Hyperparameters

The work of Wen et al. (2017) on multi-horizon forecasting may be relevant in certain aspects of the optimization and training the model. Because his work provide [10] adaptive mechanisms to optimize sequence forecasting. For the Transformer model utilized in this study we configured with carefully selected hyperparameters to balance performance and computational efficiency. The key settings include:

- $d_{\text{model}} = 80$: The embedding dimension, determining the size of the feature representation.
- $n_{\text{head}} = 1$: A single attention head is used to maintain model simplicity and reduce computational overhead.
- num_layers $= 2$: The number of layers in the Transformer's encoder stack.
- dim_feedforward $= 128$: The size of the feedforward network within each encoder block.
- seq_len $= 23$: The length of the input sequences, corresponding to past observations used for predicting the next time step.
- dropout $= 0.3$: The dropout rate applied to mitigate overfitting.
- num_epochs $= 300$: The maximum number of training epochs.
- learning_rate $= 0.0001$: The initial learning rate for gradient updates.
- patience $= 20$: The number of consecutive epochs without improvement before triggering early stopping.

To optimize the learning process, the **AdamW optimizer** is employed. This algorithm extends Adam by incorporating **weight**

decay, which acts as a form of L2 regularization, preventing excessive parameter updates and improving generalization.

6.3.1 Learning Rate Scheduling with Warmup

A **linear learning rate scheduler with warmup**, implemented using the Hugging Face `transformers` library, is integrated to ensure a smooth optimization process:

- **Warmup phase**: The learning rate is **gradually increased** at the beginning of training to stabilize the model and prevent erratic updates.
- **Decay phase**: After the warmup period, the learning rate **gradually decreases**, preventing large weight updates that could destabilize convergence.

The scheduler is defined as follows:

$$\text{num\_warmup\_steps} = 0.1 \times \text{total\_steps} \tag{6.1}$$

$$\text{total\_steps} = \text{num\_epochs} \times \text{number of batches per epoch} \tag{6.2}$$

During training, weight updates follow this sequence:

1. **Compute the loss gradient** via backpropagation: loss.backward().
2. **Update model parameters** using AdamW: optimizer.step().
3. **Adjust the learning rate** according to the scheduler: scheduler.step()

6.3.2 Training with Early Stopping

To prevent overfitting and optimize training duration, we create the `eval_early()` function that implements an **early stopping strategy**, which monitors the validation loss and halts training if no improvement is observed for a predefined number of epochs (`patience`).

6.3.3 Training Workflow

1. Iterate over **epochs** until reaching the maximum limit or triggering early stopping.
2. For each **batch** in the training dataset:
   (a) Compute model predictions on the input sequence.
   (b) Calculate the **MSE integrated in WeightedLoss function** between predictions and true values.
   (c) Perform **backpropagation** to compute gradients.
   (d) Update model parameters using **AdamW**.
   (e) Adjust the learning rate using the scheduler.
3. After each epoch, evaluate the model on the **validation set** with MSE loss .
4. Save the model if it achieves a **new lowest validation loss**.
5. If validation loss does not improve for `patience` consecutive epochs, **training stops early** to prevent unnecessary computation.

6.3.4 Evaluation on the Test Set

Once training is complete, the **best-performing model** (i.e., the one with the lowest validation loss) is loaded from the saved checkpoint (`best_model.pth`) and evaluated on the test dataset using the `evaluate_test_set()` function.

Evaluation Workflow

1. **Disable gradient updates** by setting the model to evaluation mode (`model.eval()`).
2. Iterate over **test batches**:
   (a) Generate predictions for each input sequence.

(b) Compute the test loss (**MSE**).

(c) Store the actual values (y_{true}) and predictions (y_{pred}) for further analysis.

3. Compute key performance metrics:

- **Root Mean Squared Error (RMSE)**: Measures the square root of the mean squared differences between predictions and actual values.
- **Mean Absolute Error (MAE)**: Computes the average absolute difference between predicted and true values.
- R^2 **(Coefficient of Determination)**: Evaluates how well the model explains variance in the target variable, with values closer to **1.0** indicating better fit.

6.4 Training Duration Measurement

To assess computational efficiency, the total training time is recorded and displayed in a structured format:

1. Calculate elapsed time in **seconds**.
2. Convert to **hours, minutes, and seconds**.
3. Display the total training duration as:

$$\text{Training Duration: Xh Ym Zs} \tag{6.3}$$

7 Results and Discussion

7.1 Evaluation Metrics

To compare and assess the performance of transformer models regression we use 4 metrics . Let us first remind the concept of the used metrics.

7.1.1 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) measures the average magnitude of the errors between predicted and actual values, without considering their direction. It is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{7.1}$$

where y_i represents the actual values, $\hat{y}_i$ represents the predicted values, and n is the number of observations. MAE provides an intuitive measure of prediction accuracy, where lower values indicate better performance.

7.1.2 Mean Squared Error (MSE)

The Mean Squared Error (MSE) quantifies the average squared difference between predicted and actual values. It is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7.2}$$

MSE penalizes larger errors more heavily than MAE due to the squaring term, making it particularly useful for highlighting

significant deviations.

7.1.3 Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) is the square root of the MSE, providing an error metric with the same unit as the target variable:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{7.3}$$

RMSE is widely used because it gives a more interpretable measure of model accuracy by maintaining the same unit as the original data.

7.1.4 Coefficient of Determination (R^2)

The R^2 score, or coefficient of determination, evaluates the proportion of variance in the target variable that is explained by the model. It is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{7.4}$$

where $\bar{y}$ is the mean of the actual values. An R^2 value close to 1 indicates a good fit, while a value close to 0 suggests that the model does not explain much of the variance in the data.

These metrics collectively provide a comprehensive evaluation of our models, helping to analyze absolute and relative errors in predictions.

7.2 Results

This section provides a summary of the results from all the experiments conducted across various use cases. After completing the

training phase, the resulting model was saved as a .pt file (PyTorch) and subsequently tested on a dataset that was normalized using parameters derived from the training data.

7.2.1 Comparative Evaluation of Transformer Model Performance

This section provides a comparative analysis of the results obtained from both the Vanilla Transformer and the Custom Transformer models in the context of multivariate time series forecasting. The evaluation is based on key performance metrics, including RMSE, MAE, R^2 score, test loss, and training time. To ensure stability, both models were trained using input sequences of the same length of 23 hours. The table bellow report the performance of models.

Table 7.1: Comparative Metric Values of the Models

Metrics	**RMSE**	**MAE**	**R2**	**Test Loss**	**Training-Time(min)**
`Vanilla Transformer`	0.4414	0.2942	0.7834	0.1950	28mm 36s
`Custom Transformer`	0.4411	0.2849	0.7884	0.1949	29 mm 24s

Prediction Accuracy

The predictive accuracy of the models is assessed using the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), and both models exhibit very similar error metrics, as reflected in `Table 7.1` . However,as shown in `Table 7.1` the Custom Transformer demonstrates a slight advantage, achieving a lower Mean Absolute Error (MAE) of 0.2849 compared to 0.2942 for the Vanilla Transformer. Furthermore, the coefficient of determination (R^2) is marginally higher for the Custom Transformer (0.7884 vs. 0.7834), suggesting that the modified attention mechanism enhances the model's ability to capture underlying patterns within the dataset.

The improvement in R^2, which measures the proportion of variance explained by the model, indicates that the custom architec-

ture learns the relationships among the input features effectively , leading to slightly better predictive performance.

Generalization Capability

An essential aspect of model evaluation is its ability to generalize well to unseen data. The Custom Transformer exhibits a marginally lower test loss (0.1949) compared to the Vanilla Transformer (0.1950), suggesting a slight enhancement in generalization. Although this difference is minimal, it implies that the modifications in the attention might contribute to better feature representation and learning. However, it is also possible that this observed improvement is not solely attributable to the architectural changes but is instead influenced by the applied feature engineering techniques.

Computational Efficiency

A notable trade-off between the two models is computational efficiency. The Custom Transformer requires a longer training duration (29 minutes 24 seconds) compared to the Vanilla Transformer (28 minutes 36 seconds). This increase in training time is likely due to the additional complexity introduced by the modified attention mechanism, which involves multiple matrix operations. Furthermore, the need to propagate additional information between the attention heads may contribute to the increased computational load. While this trade-off is relatively minor in an experimental setting, it could become significant in real-world applications where efficiency is a critical factor.

Discussion and Recommendations

From a performance standpoint, the Custom Transformer appears to offer a slight edge in accuracy and generalization. If the primary objective is to achieve the highest possible predictive accuracy, this architecture may be a preferable choice. However, if computational

efficiency is a primary concern, the Vanilla Transformer might be more suitable, as the observed performance gains from the Custom Transformer are relatively minor. Overall, the results indicate that the Custom Transformer shows slight advantages in terms of accuracy and generalization, these improvements come at the cost of increased training time. The choice of model should therefore be guided by the specific requirements of the application, balancing predictive performance and computational efficiency.

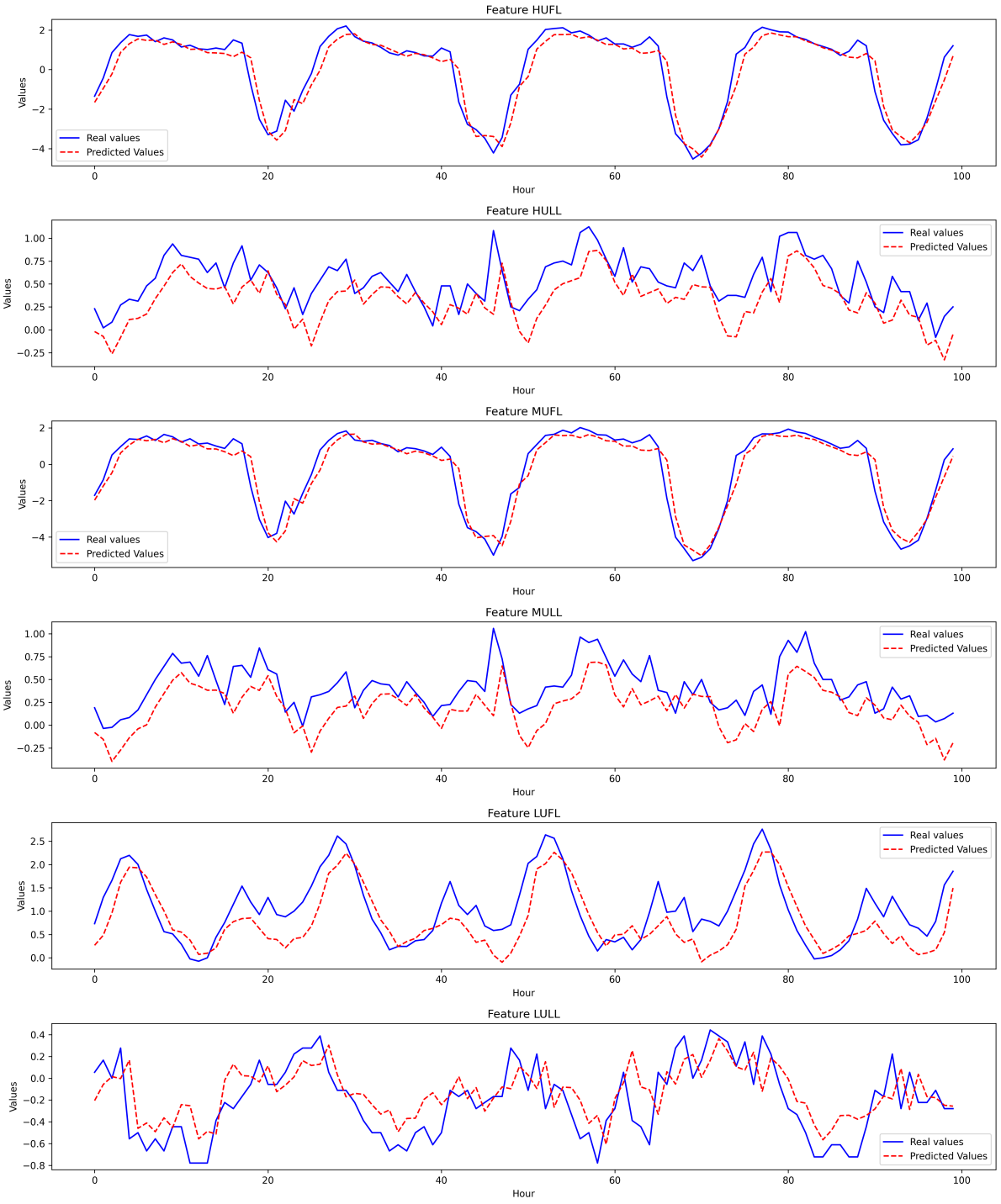

Figure 7.1: Vanilla's Prediction Visualization.

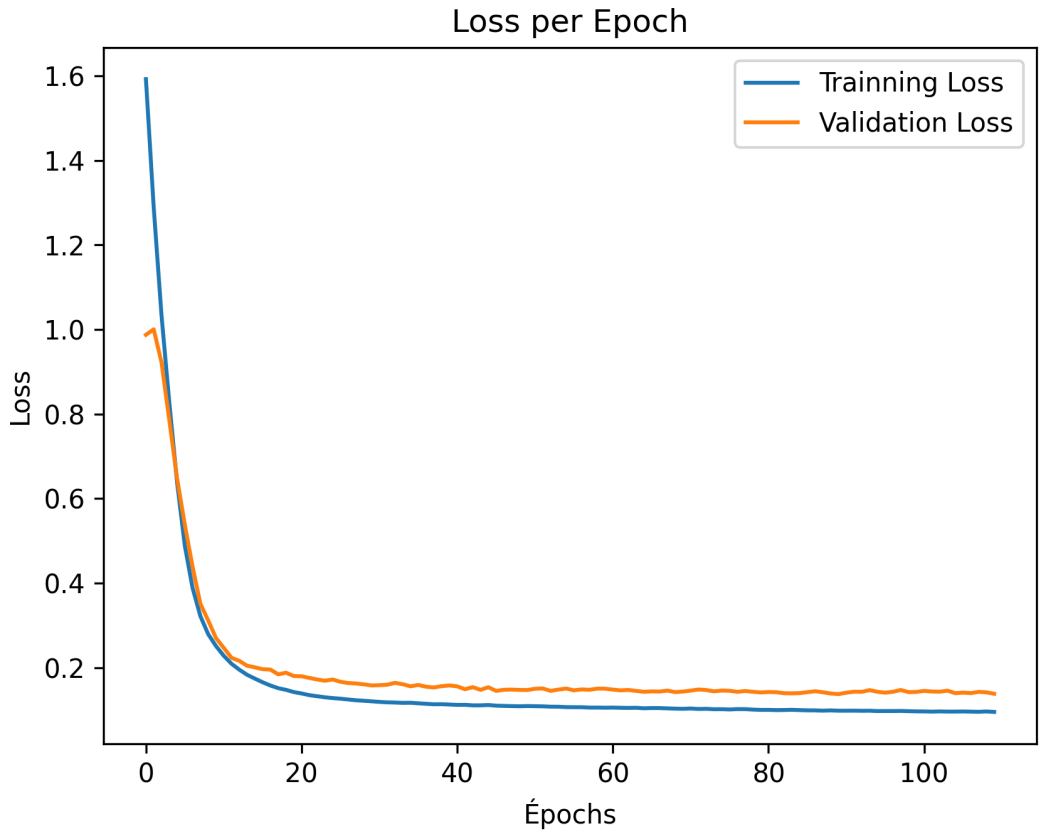

Figure 7.2: Vanilla's Training Curve.

The graphical visualization of the Vanilla Transformer in Fig. 7.1 struggles to effectively capture the intra-specific characteristics of the features. However, the HUFL and MUFL features exhibit a predicted curve that closely aligns with the actual values. Additionally, the learning curve in Fig. 7.2 consistently declines before stabilizing, indicating that the model is successfully learning and enhancing its ability to recognize general patterns within the data.

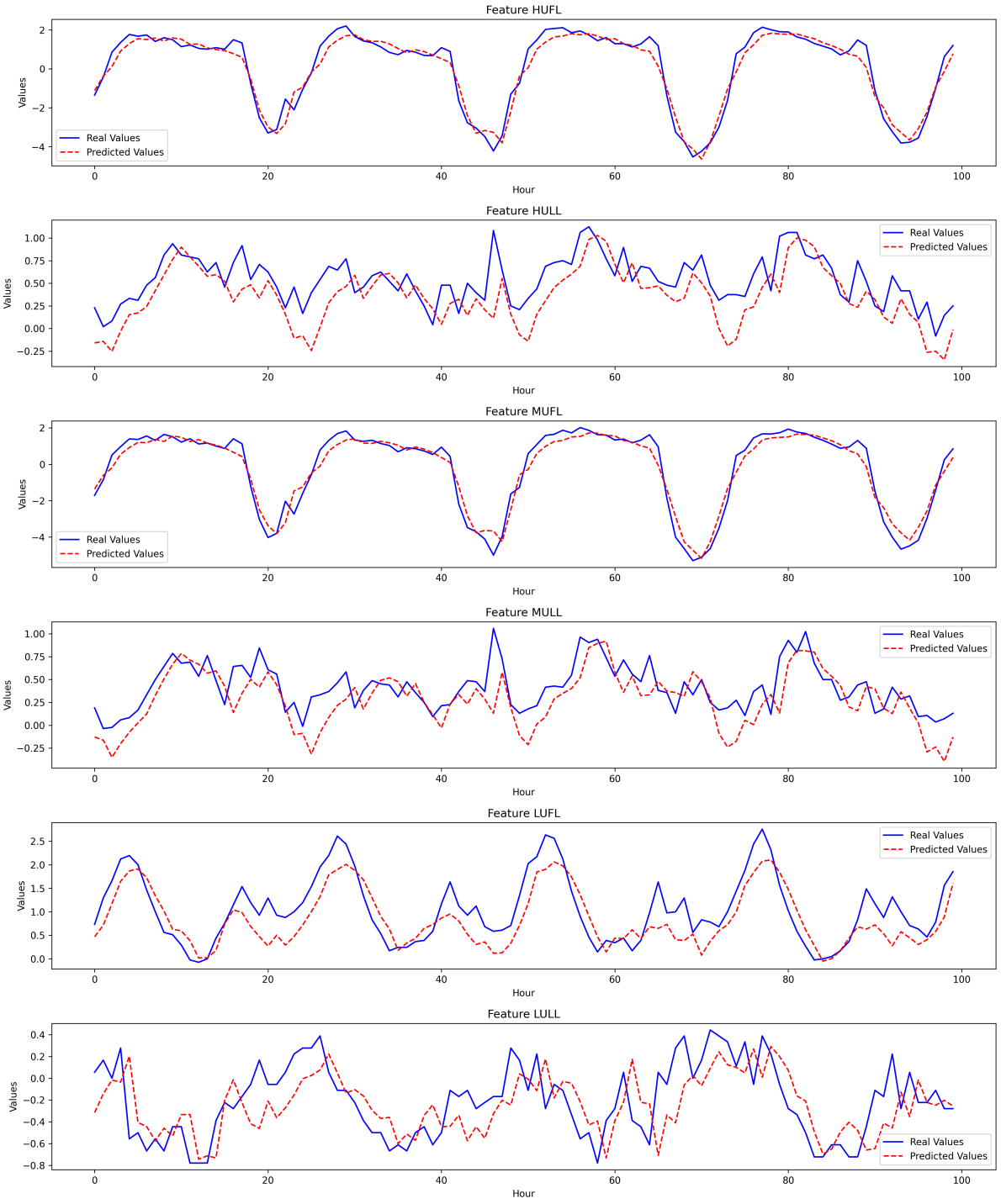

Figure 7.3: New Transformer Prediction Visualization.

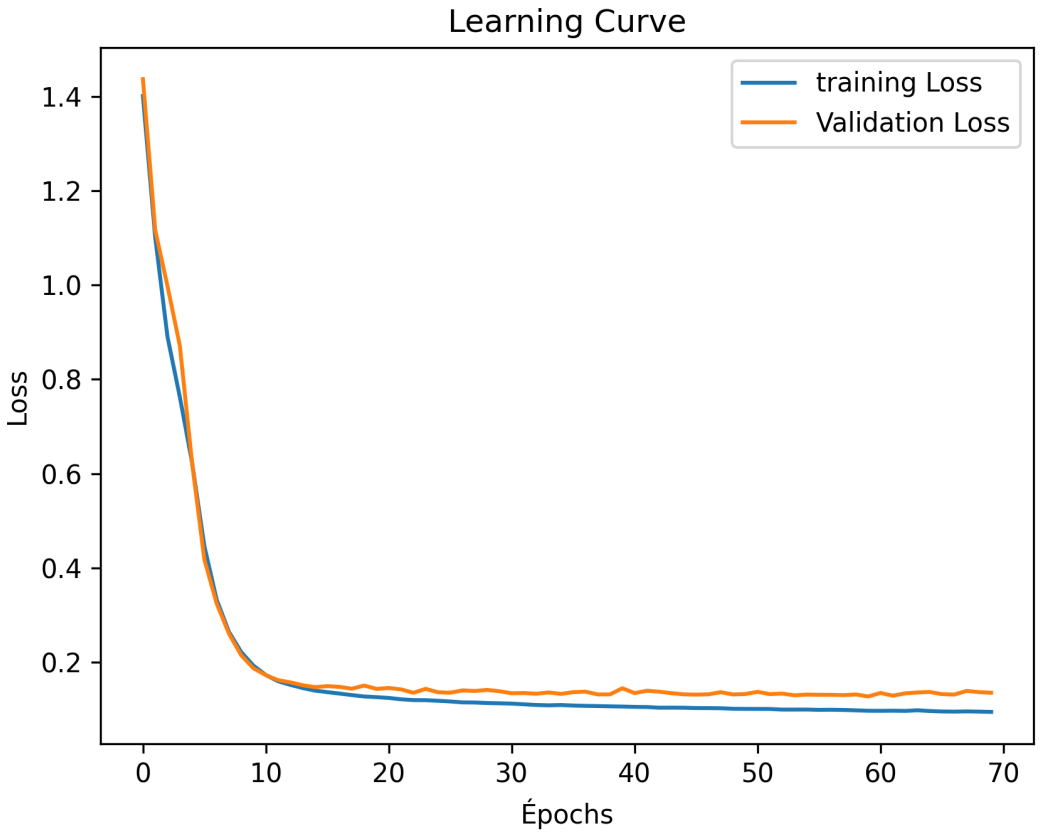

Figure 7.4: New Transformer Training Curve.

The graphical analysis of the new Transformer (Fig. 7.3) highlights its difficulty in effectively capturing the intra-specific characteristics of the features. However, there is a noticeable improvement in its ability to align predicted values more closely with actual ones. Additionally, the learning curve (Fig. 7.4) shows a slight crossover, indicating that the model is indeed learning. However, the presence of limited parameters restricts its capacity to fully grasp broader patterns, ultimately affecting its generalization ability.

7.2.2 Comparative Analysis of Model Performance on Different Sequence Lenght

Here, we examine a comprehensive evaluation of the Custom Transformer model's performance when trained with both short and long embedding sequences. The comparison is based on key performance indicators, including RMSE, MAE, R^2, test loss, and training time.Two training configurations were evaluated and the performance are summarize are in the tables bellow:

- A **12-hour sequence length** used to predict the values of the 13th hour sequence. (Table 7.2).
- A **47-hour sequence length** used to predict the values of the 48th hour sequence. (Table 7.3).

Table 7.2: Metric Values on Short Embedding New Transformer test

Metrics	**RMSE**	**MAE**	**R2**	**Test Loss**	**Training-Time(min)**
`Custom Transformer`	0.4539	0.3000	0.7802	0.2063	11mm 29s

Table 7.3: Metric Values on Long Embedding New Transformer test

Metrics	**RMSE**	**MAE**	**R2**	**Test Loss**	**Training-Time(min)**
`Custom Transformer`	0.4395	0.2911	0.7953	0.1936	29mm 55s

The results from both experiments allow us to assess how sequence length influences model accuracy, generalization, and computational efficiency.

7.2.3 Performance with Short Embedding Sequences

As illustrated in Table 7.2, the model trained on short embeddings demonstrates an RMSE of 0.4539 and an MAE of 0.3000. The R^2 score of 0.7802 suggests that the model captures a substantial portion of the dataset's variance. However, the test loss is relatively higher (0.2063), which may indicate a slightly weaker generalization ability compared to the long embedding variant. A

key advantage of this approach is the significantly reduced training time just 11 minutes and 29 seconds—making it a computationally efficient option.

7.2.4 Performance with Long Embedding Sequences

On the other hand, as shown in Table 7.3, the Custom Transformer trained with long embeddings achieves superior results across almost all metrics. The RMSE drops to 0.4395, while the MAE improves to 0.2911, reflecting enhanced predictive accuracy. Moreover, the R^2 value increases to 0.7953, indicating a better ability to explain variations within the dataset. Additionally, the test loss decreases to 0.1936, further supporting the idea that using longer embeddings improves generalization. However, this improvement comes at the cost of increased computational demand, with the training time extending to 29 minutes and 55 seconds.

7.2.5 Comparison and Insights

The results reveal a trade-off between accuracy and computational efficiency. While the long embedding model delivers better predictive performance, it requires considerably more training time. The improvements seen in the long embedding variant may stem from its ability to capture longer-range dependencies within the data. Furthermore, the feature engineering techniques employed—such as sinusoidal encoding for time-related variables and interaction terms like OT^2 and $OT \times LUFL$—likely contributed to the model's enhanced learning capabilities.

When choosing between short and long embeddings, the decision should be guided by the available computational resources and the specific accuracy requirements of the task. If speed and efficiency are the priority, the short embedding approach is a viable choice. However, for applications that demand greater precision and generalization, the long embedding configuration offers a dis-

tinct advantage.

Moving forward, several aspects warrant further investigation:

- Assessing the impact of embedding lengths beyond the ones tested in this study.
- Exploring the effect of increasing the number of attention heads (n_{head}) on model performance.
- Conducting an ablation study to isolate the contributions of individual feature engineering techniques and attention mechanisms.

These investigations could provide deeper insights into optimizing transformer-based models for time-series forecasting in energy management applications.

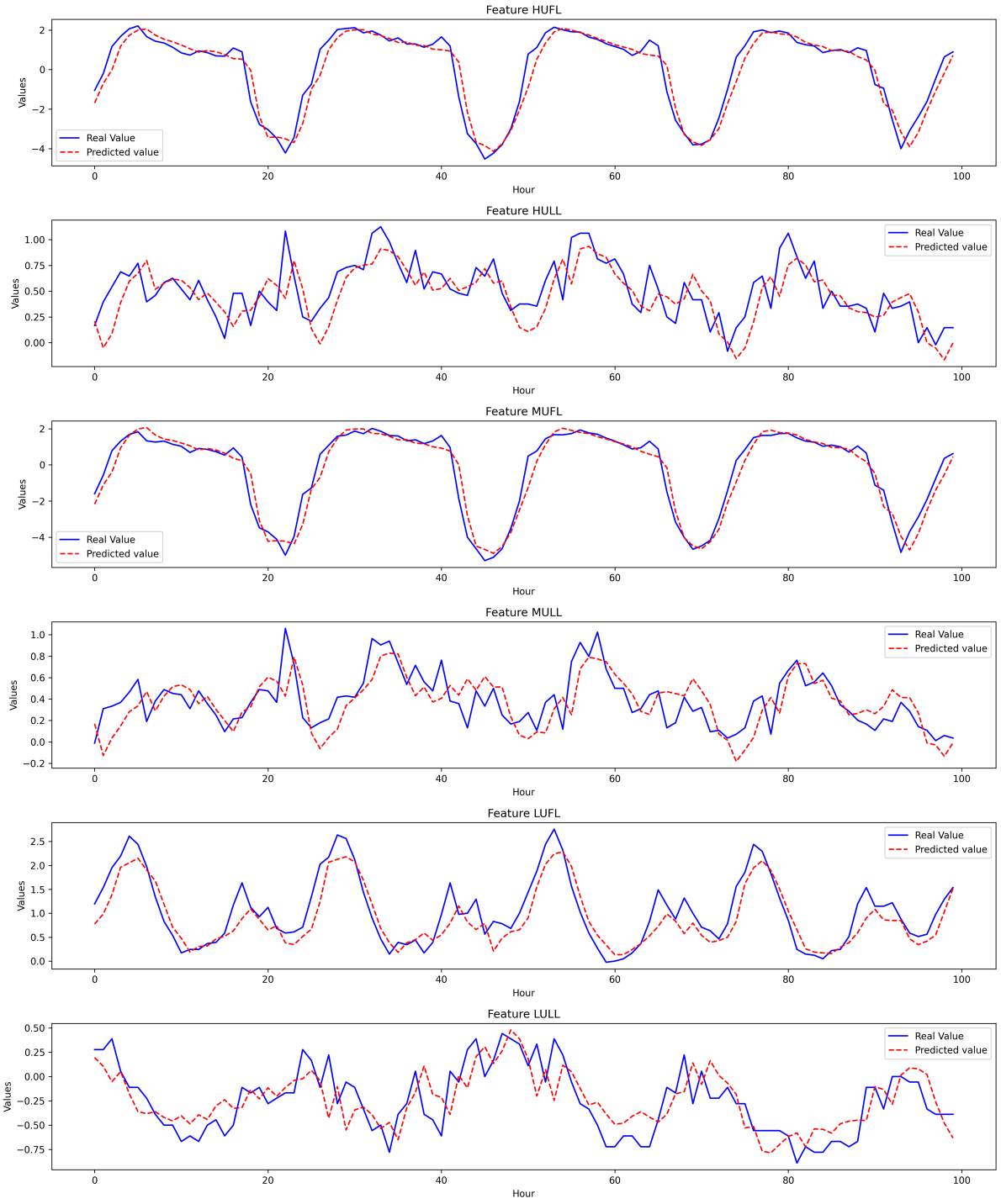

Figure 7.5: Long Sequence Training Prediction.

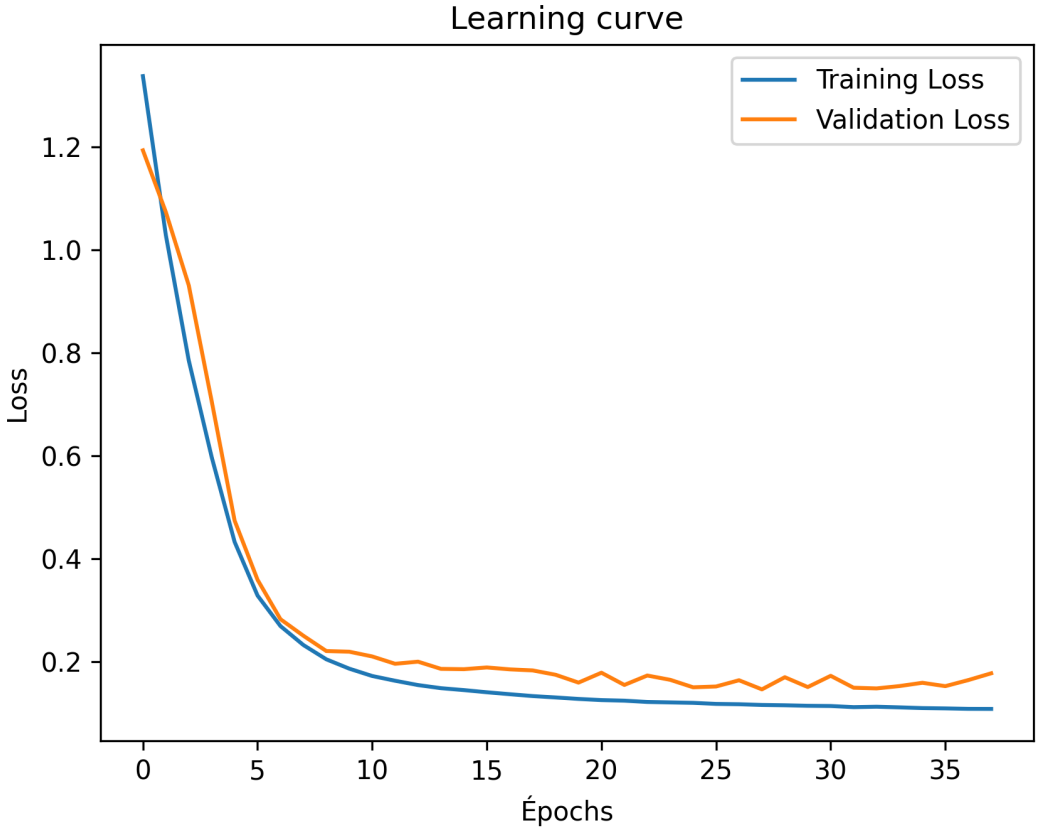

Figure 7.6: Long Sequence Training Curve.

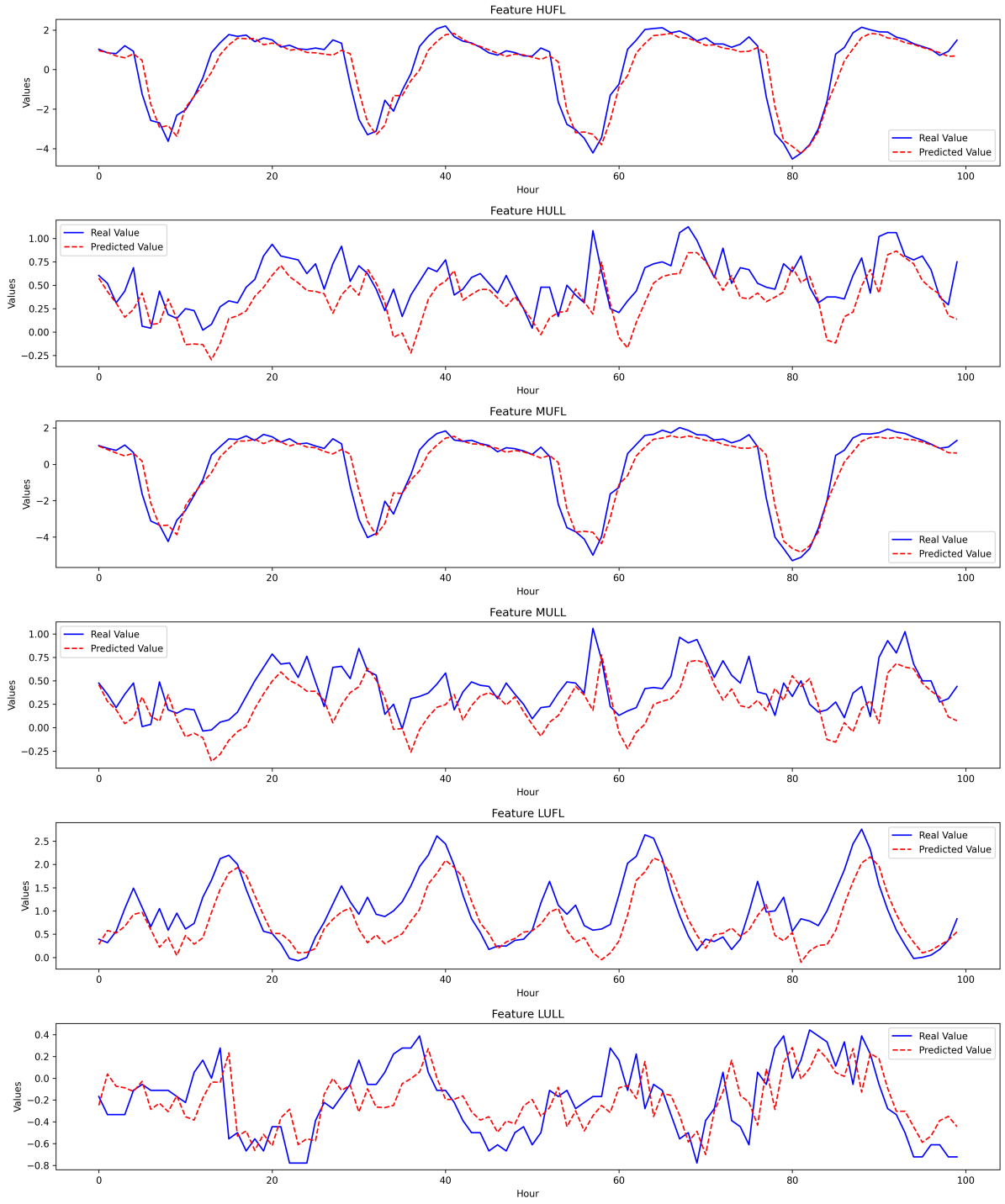

Figure 7.7: Short Sequence Training Prediction.

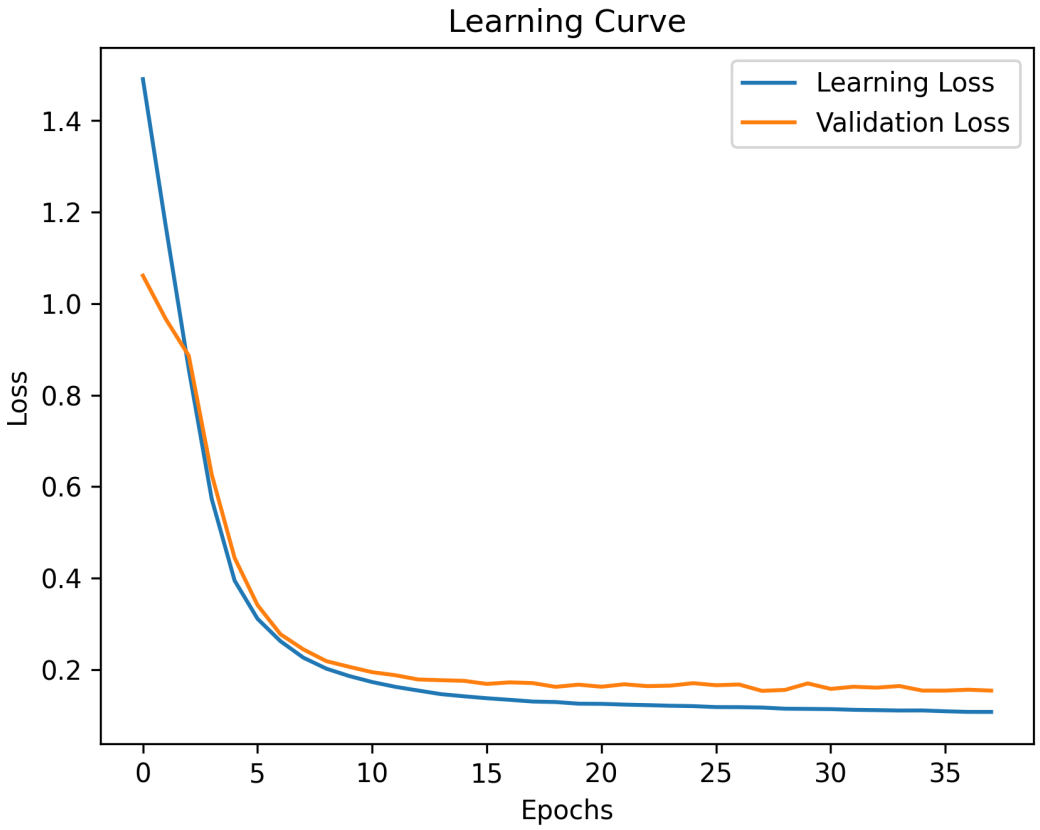

Figure 7.8: Short Sequence Training Curve.

The prediction plots for the new Transformer model (Fig.7.5) on long sequences and (Fig.7.7) short sequences show notable differences between long and short sequences. This is likely due to the fact that fluctuations in this time series data are particularly prominent on a daily scale. Consequently, the transformer's attention mechanism seems to effectively capture significant variations between days, depending on the sequence length.

However, the difference in slope between the learning loss and validation loss in (Fig.7.6) for the long sequence, compared to (Fig.7.8) for the short sequence, suggests that in both cases, the training and validation losses do not fully converge. Moreover, the curves do not follow a strictly decreasing trend and exhibit irregular fluctuations, underscoring the difficulties of training long sequences with the current Transformer architecture. These in-

consistencies point to a potential issue of overfitting during long-sequence training. The difficulties encountered in training long sequences may stem from suboptimal choices of hyperparameters in relation to the training environment, making optimization more complex. Further fine-tuning of these parameters could be necessary to enhance model stability and overall performance.

8 Conclusion

The development of a specialized attention mechanism tailored for multivariate time series forecasting aims to improve the learning capabilities of Transformers when applied to the energy monitoring forecasting. This study has shown that refining the conventional attention framework—by incorporating a cross-attention mechanism that separately processes temporal features and energy-related variables enhances the model's ability to learn intricate dependencies while reducing challenges such as pattern repetition and overfitting. A comparative analysis between the standard Transformer and the proposed custom architecture indicates that, although both models demonstrate comparable generalization capabilities, the custom approach delivers a modest improvement in predictive accuracy and feature representation, albeit at the cost of greater computational demand. Furthermore, experiments involving embedding dimensions and sequence lengths highlight the inherent trade-offs between computational efficiency and forecasting precision, revealing that longer sequences facilitate improved long-term pattern recognition while introducing optimization complexities.

Looking ahead, several promising avenues could further refine Transformer-based forecasting for multivariate time series datafor the instance, Brown et al. [6] highlighted how Transformers can be adapted beyond NLP into time series and forecasting tasks. Future research could explore more adaptable attention mechanisms that can dynamically adjust to varying sequence lengths

while preserving computational efficiency. Additionally, incorporating external contextual factors—such as weather patterns, economic trends, or anomaly detection techniques—could enhance the model's predictive capabilities. Another compelling direction involves developing hybrid architectures that integrate Transformer models with convolutional or recurrent layers to better capture both localized fluctuations and hierarchical temporal structures. Advancing these methodologies could extend the applicability of this approach beyond energy management to other domains requiring high-precision forecasting, including financial markets, healthcare analytics, and industrial process optimization.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] T. Liu, P. Vincent, and A. Courville, "Hierarchical recurrent attention networks," in *Advances in neural information processing systems*, vol. 31, 2018.

[3] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *AAAI Conference on Artificial Intelligence*, 2021.

[4] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *arXiv preprint arXiv:1609.03499*, 2016.

[5] Y. Wu, S. He, Y. Liang, and L. Song, "Deep transformer models for time-series forecasting," *arXiv preprint arXiv:2001.08317*, 2020.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[8] C. K. Park, H.-K. Lee, and S.-I. Cho, "Predicting electricity demand using time-series transformer models," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2201–2211, 2020.

[9] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.

[10] R. Wen, K. Boukas, and T.-Y. Liu, "A multi-horizon quantile recurrent forecaster," in *arXiv preprint arXiv:1711.11053*, 2017.