NCA PRESENTATION

ELECTRICITY FORECASTING

D.Krishna Shandilya
Roll No. 2022BCS020

DATASET

This dataset captures time-series information related to electricity demand and generation across various utility zones and industrial consumers in Odisha. It includes demand values from four major distribution companies: TPCODL, TPWODL, TPNODL, and TPSODL, each representing different geographical regions (Central, Western, Northern, and Southern Odisha respectively). Alongside these, it records the total aggregated demand of the system, renewable power generation from solar sources, and grid frequency which reflects supply-demand balance. Additionally, the dataset contains drawl values—i.e., power consumption—by major industrial consumers like Vedanta, IMFA, and NALCO.

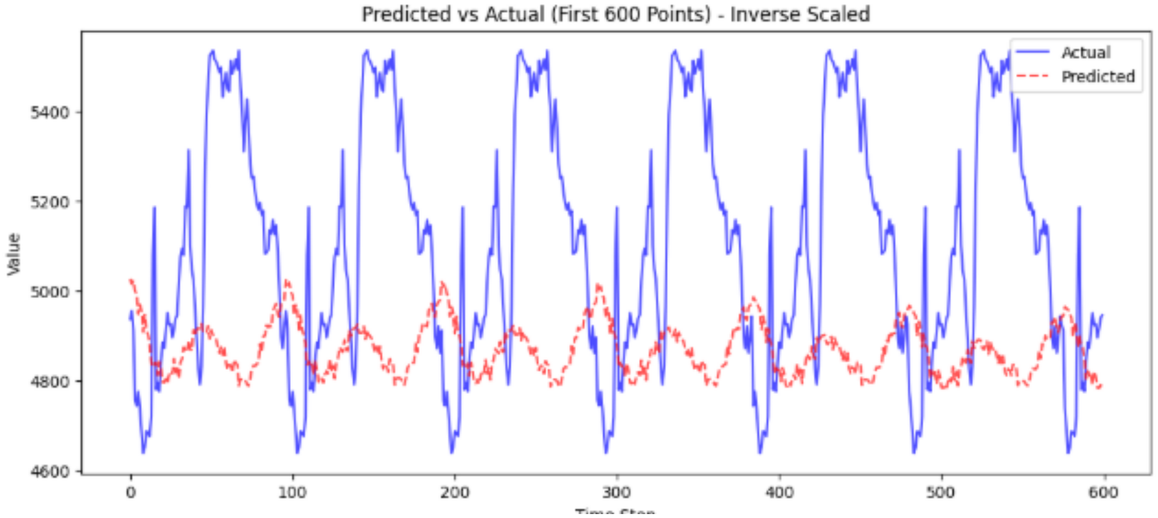
DISCUSSION

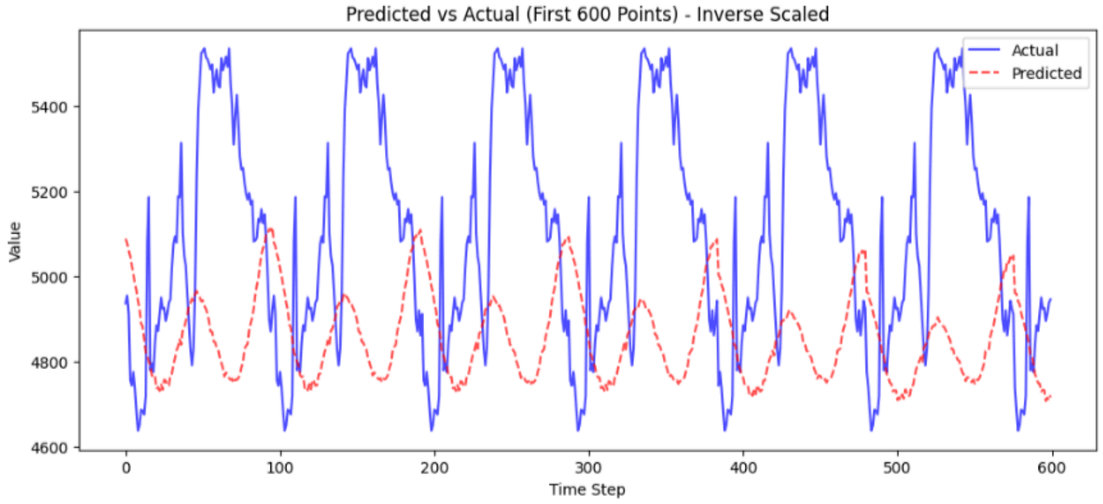
- **In this presentation, we explore various deep learning models used for electricity load forecasting. We begin with the LSTM (Long Short-Term Memory) network, and then implement the GRU (Gated Recurrent Unit), a simplified alternative to LSTM that offers comparable performance while being computationally more efficient.**

- **Finally, we enhance the forecasting capability by integrating an Attention mechanism with the LSTM architecture. This allows the model to focus selectively on the most relevant time steps from the input sequence, thereby improving interpretability and prediction accuracy. Through comparative analysis, we evaluate the strengths and limitations of each model in the context of electricity demand forecasting.**

MODEL 1

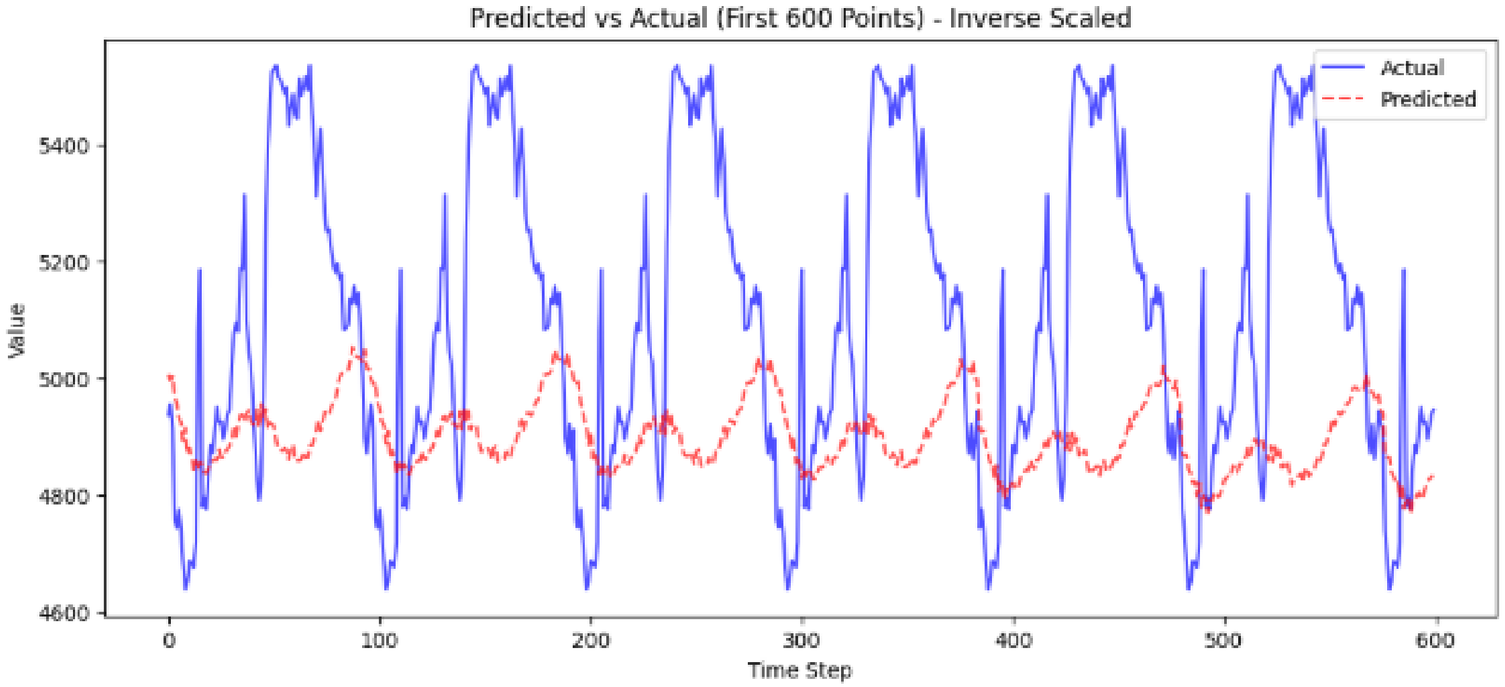
- The model is built using a stack of Long Short-Term Memory (LSTM) layers followed by fully connected (dense) layers.
- It takes as input a sequence of length 672 and is designed to predict the next 96 time steps. The LSTM component of the model consists of 3 layers, each with 256 hidden units. It uses a dropout of 0.2 .
- And then dense block begins with a 256-unit layer followed by ReLU activation and dropout for regularization. It is then followed by a 128-unit layer with ReLU, and finally a linear layer that outputs a 96 dimensional vector representing .
- The model uses the Mean Squared Error (MSE) loss function for optimization, as this is a regression task, and the optimizer used is Adam with a learning rate of 0.001, providing adaptive gradient-based updates during training.





MODEL 2

- The model is built using a stack of GRU layers followed by fully connected (dense) layers.
- It takes as input a sequence of length 672 and is designed to predict the next 96 time steps. The GRU component of the model consists of 3 layers, each with 256 hidden units. It uses a dropout of 0.2 .
- And then dense block begins with a 256-unit layer followed by ReLU activation and dropout for regularization. It is then followed by a 128-unit layer with ReLU, and finally a linear layer that outputs a 96 dimensional vector representing .
- The model uses the Mean Squared Error (MSE) loss function for optimization, as this is a regression task, and the optimizer used is Adam with a learning rate of 0.001, providing adaptive gradient-based updates during training.



MODEL 3

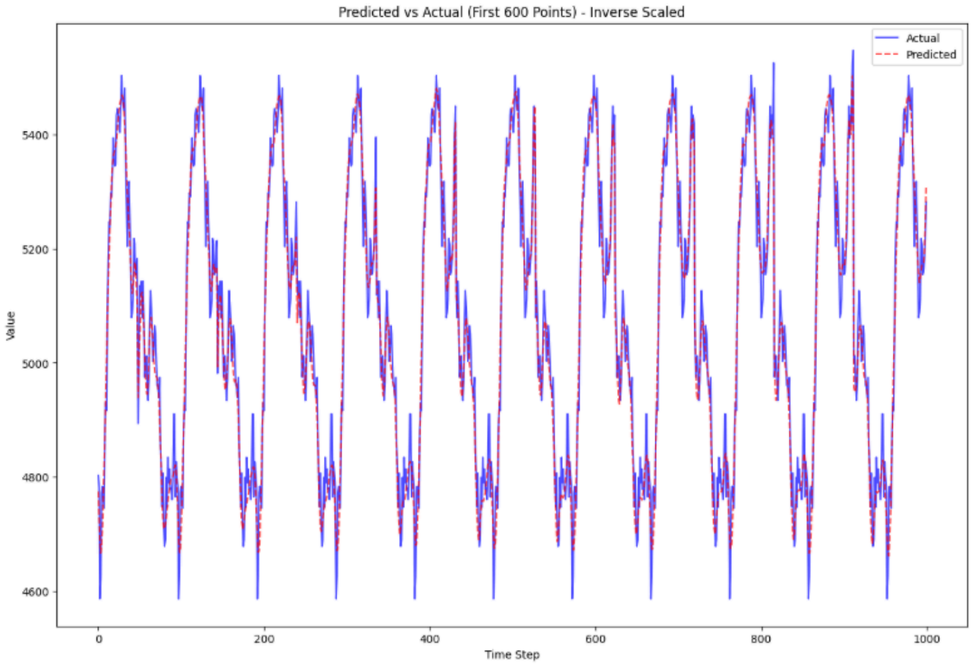
The model follows a sequence-to-one design with attention, where an encoder first processes the input sequence, and a decoder learns to selectively attend to relevant encoder outputs while generating the prediction.

Encoder

- **The Encoder is a standard LSTM that processes the past T time steps of the input sequence.**
- **It takes the raw inputs and feeds them directly to the LSTM,**
- **This captures the temporal dynamics of the input and compresses them into hidden representations used by the decoder.**

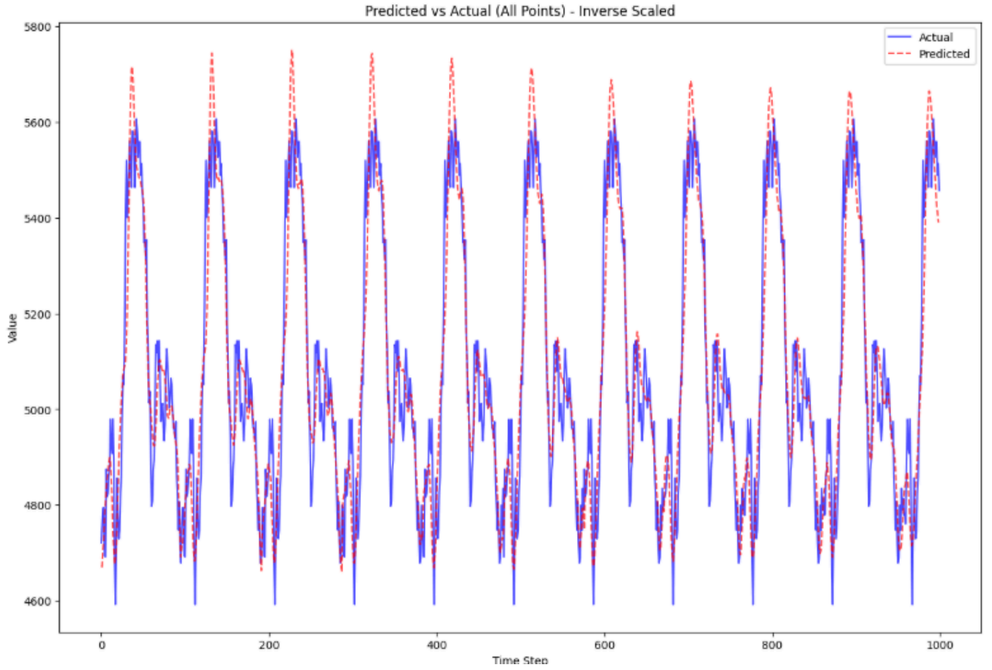
Decoder with Attention

- **The Decoder is responsible for generating the forecast of length output_size (future time steps) based on the encoder output and previous target values.**
- **It computes attention weights over the encoder outputs using a custom attention network. This helps focus on relevant encoder outputs instead of treating all equally.**
- **Attention vector is formed by concatenating decoder hidden state d_n, cell state c_n, and encoder outputs.**
- **It passes through two linear layers with tanh activation, followed by a softmax to get attention scores (β).**
- **Applies the attention weights to form a context vector via a weighted sum over encoder outputs.**
- **Concatenates the context vector with the previous true target value y_prev[:, t] and feeds it into the decoder LSTM.**
- **Updates the decoder hidden and cell states.**
- **After processing the entire target input sequence, it:**
- **Concatenates the final decoder hidden state with the final context vector.**
- **Passes this through a fully connected layer to output the final prediction vector of shape (batch_size, output_size)**



Mse Loss: 1.19e-4

MAPE Loss: 3.15%



Mse Loss: 3e-4

MAPE Loss: 3.87%

THANK YOU