NN Task

Rana Hassan Hosny
GIZA SYSTEMS | RANA.HOSNY@GIZASYSTEMS.ONMICROSOFT.COM

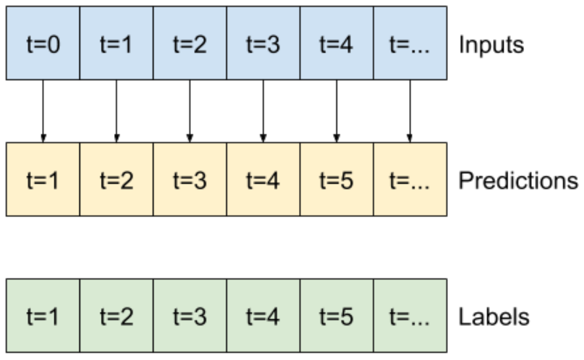
Table of Contents

1. Data Preprocessing:

- **Wind Velocity**: The minimum values in the wind velocity (wv (m/s)) and maximum wind velocity (max. wv (m/s)) columns contain -9999, likely indicating erroneous data. Since wind velocity should logically be non-negative, these values will be replaced with zero.
- **Extracting Time Features**: Convert the Date Time column from string format to seconds. To capture daily and yearly patterns, use sine and cosine transformations to create "Time of day" and "Time of year" signals, making the data more model friendly.
- **Normalize the Data**: Scaling features before training a neural network is essential. Here, normalization is performed by subtracting the mean and dividing by the standard deviation of each feature, calculated only on the training data to prevent data leakage into validation and test sets.
- **Split the Data:** The data will be split into training, validation, and test sets in a (70%, 20%, 10%) proportion. The data is not shuffled before splitting to
- **For easier model interpretation**, the wind direction and velocity columns are combined into a single wind vector.

2. Model architecture:

The architecture uses a wide_window defined by the WindowGenerator class, where the input sequence length is 24, and the label sequence is also 24. In this setup, the output is assumed to be affected only by the previous input. This means each predicted time step relies on the immediate past context, without considering longer historical dependencies. This setup may work well for capturing short-term dependencies in time series data, especially if the data exhibits such patterns.

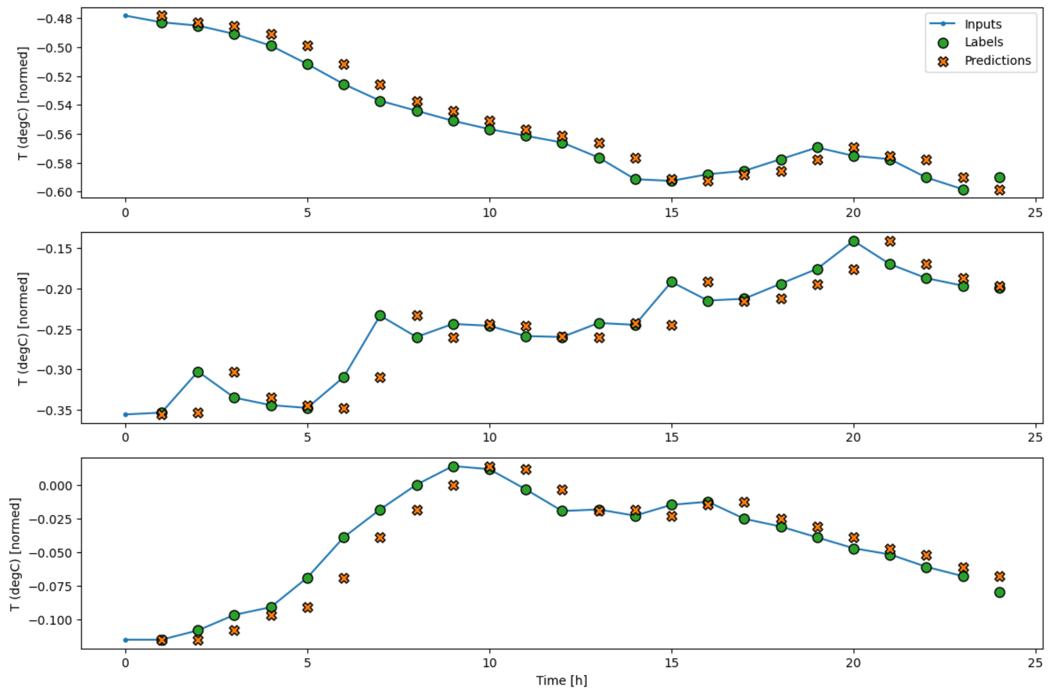


1. Baseline Model

The **Baseline Model** serves as a simple reference model, returning the input time-series as the output. This model assumes that the future value is identical to the most recent observation.

Model Architecture:

- The Baseline Model only selects the label column from the input data, defined by label_index.



2. Dense Model

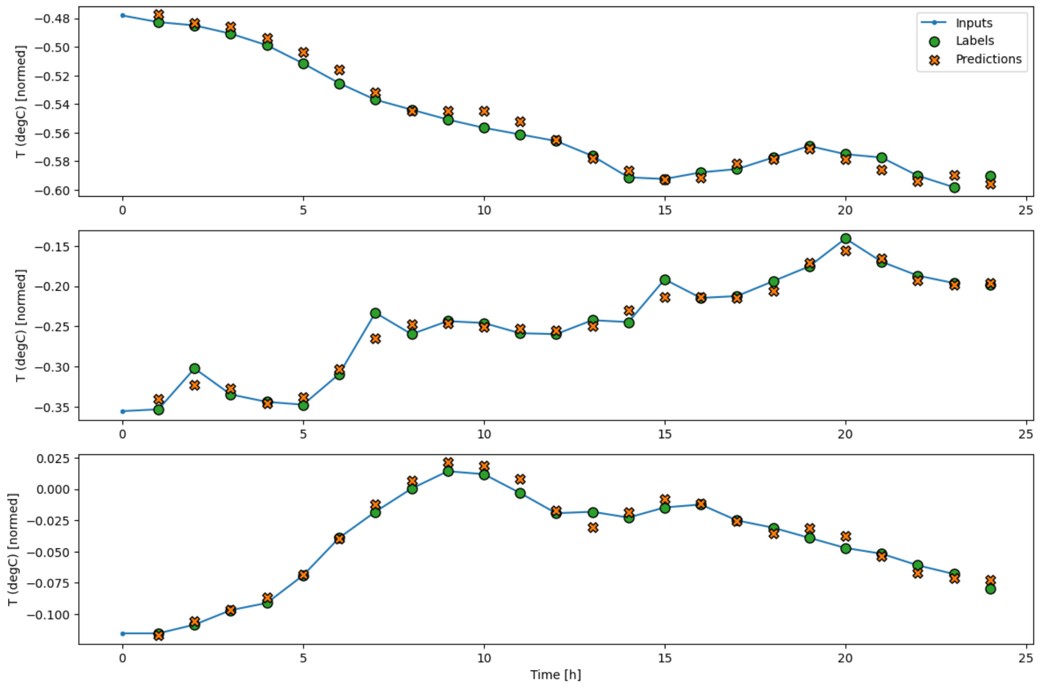
The **Dense Model** is a fully connected feedforward neural network designed to learn complex relationships in the data.

Model Architecture:

- The model consists of:
    - **Input Layer**: 24 * 19 (flattened input where 19 is the number of the features).
    - **Hidden Layers**: Two fully connected layers (fc1 and fc2) with ReLU activations.
    - **Output Layer**: A fully connected layer that predicts values over the required time steps (24 in this case).

Training Process:

- The model is trained using the **Adam optimizer** and the **MSE loss function**.
- The model is evaluated on the validation and test datasets using **MAE** as an additional metric.



3. LSTM Model

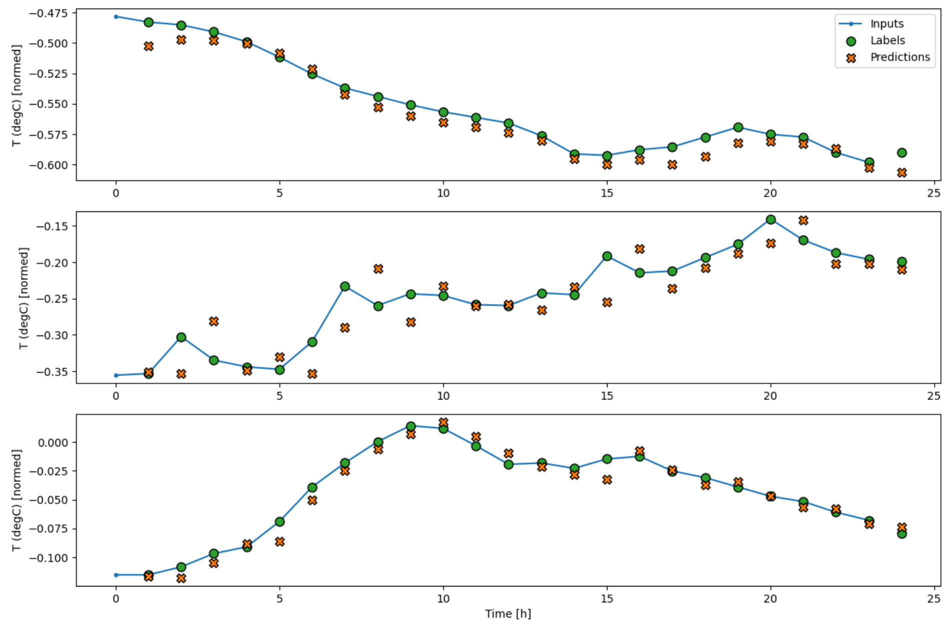
The **LSTM Model** is a type of recurrent neural network capable of capturing long-term dependencies in time-series data.

Model Architecture:

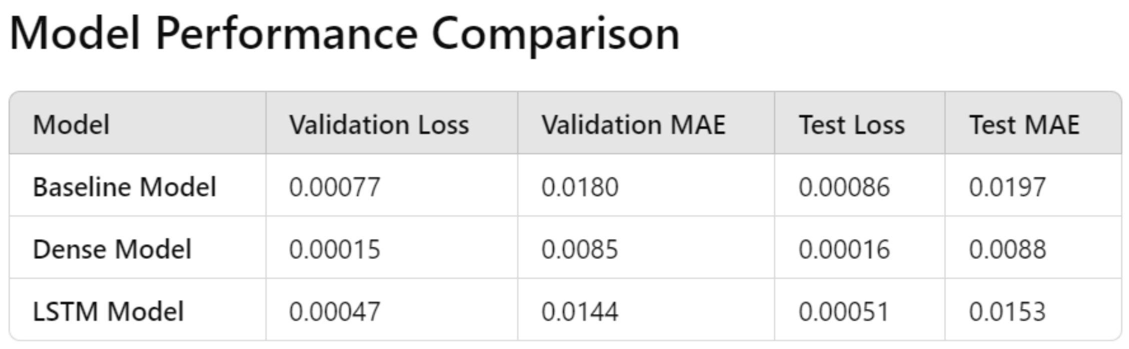
- The model uses an **LSTM layer** followed by a **fully connected layer** to output the predictions.
- **LSTM Layer**: This layer processes the sequential data and captures the temporal dependencies.
- **Output Layer**: The linear layer maps the LSTM output to the required time-series prediction.

Training Process:

- The model is trained in a similar manner to the Dense Model using the **Adam optimizer** and **MSE loss function**.

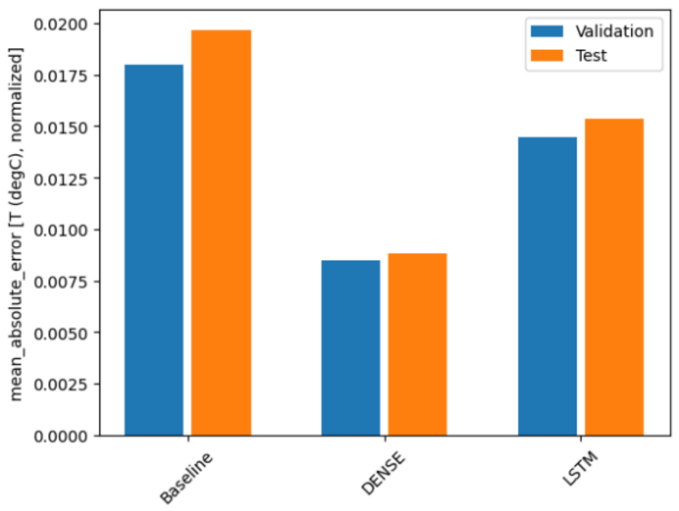


3. Testing and evaluation for one-shot models:



Model Performance Comparison

Model	Validation Loss	Validation MAE	Test Loss	Test MAE
Baseline Model	0.00077	0.0180	0.00086	0.0197
Dense Model	0.00015	0.0085	0.00016	0.0088
LSTM Model	0.00047	0.0144	0.00051	0.0153

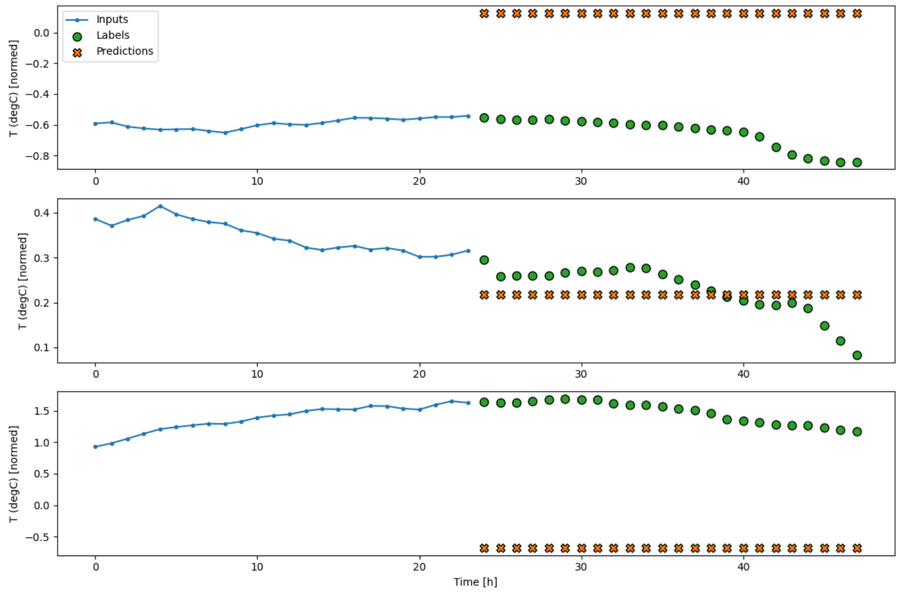


The coming architecture:

uses a multi_window defined by the WindowGenerator class, where the input sequence length (input_width) is 24, but the label sequence length (label_width) and the shift (shift) are both set to OUT_STEPS, which is 24 in this case. This configuration implies that the model predicts the next 24-time steps based on the past 24 time steps.

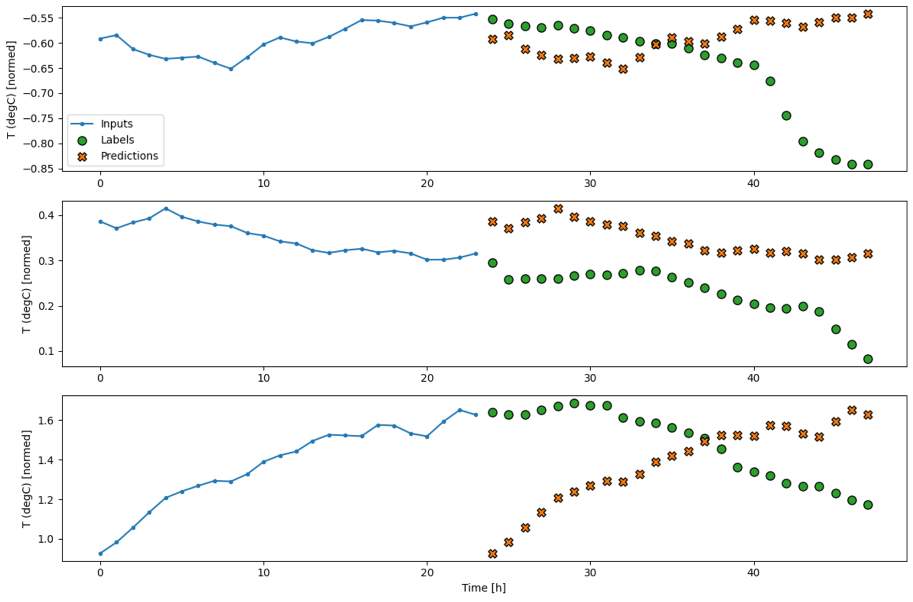
MultiStepLastBaseline:

- This model assumes that future values can be predicted by repeating the last observed value from the input sequence.
- **Input**: The model takes the last time step from the input data and repeats it for the entire prediction horizon (i.e., OUT_STEPS).
- **Output**: The output is a sequence where each value is the same as the last value from the input, repeated for the required number of future time steps.



RepeatBaseline:

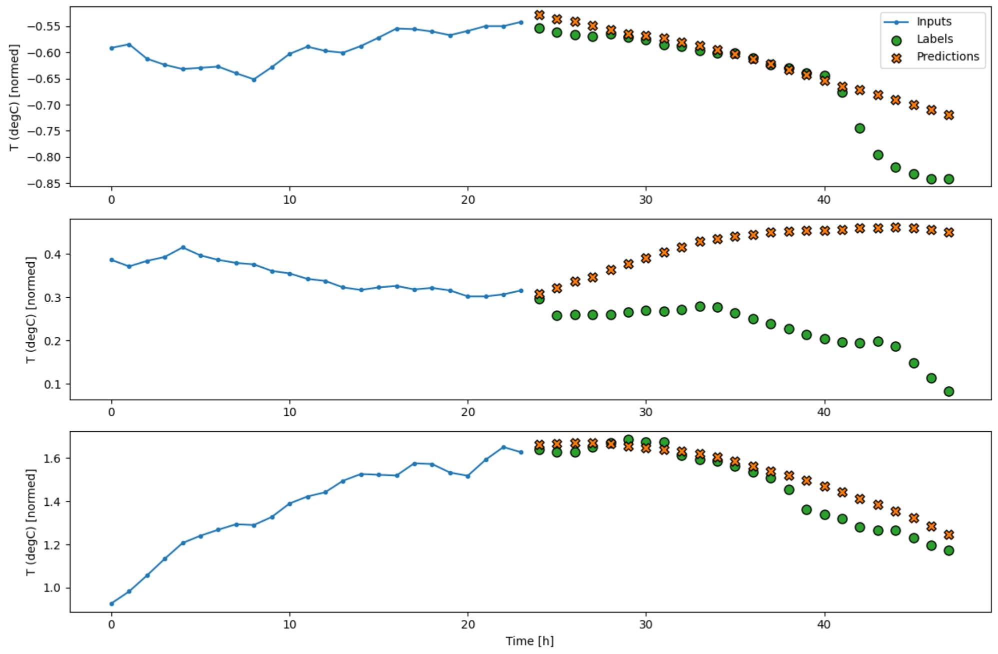
- This model assumes that future values are based on a repeated pattern, using the last 24 time steps from the second feature in the input sequence.
- **Input**: The model repeats the last 24 time steps for each output step in the forecast.
- **Output**: Similar to **MultiStepLastBaseline**, but the values are repeated from the last 24 time steps instead of just the last single timestep.



MultiLSTMModel:

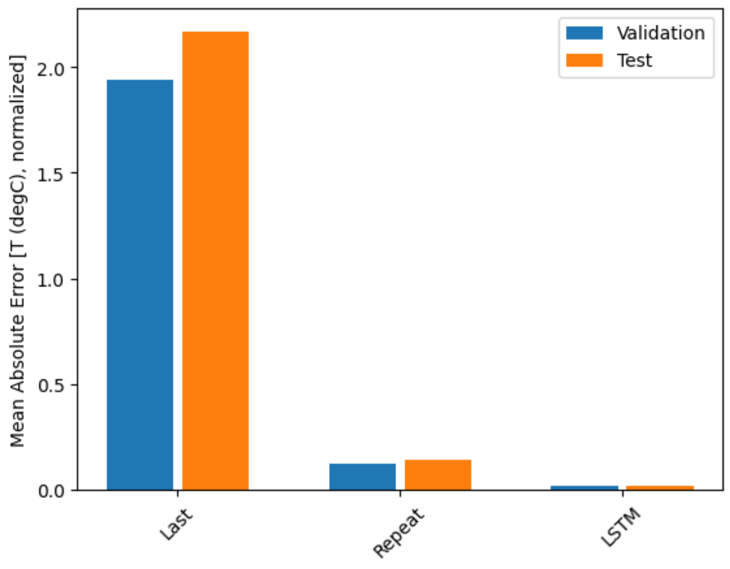
This model consists of the following layers:

1. **LSTM Layer**:
   - The core component of the model is the **Long Short-Term Memory (LSTM)** layer, which is responsible for processing sequential data and capturing temporal dependencies.
   - This layer takes input sequences and learns the relationships between time steps over longer sequences. It is configured with:
     - **Input size** of **19**, representing the number of features at each time step.
     - **Hidden size** of **32**, which determines the number of units in the LSTM that store the learned representation of temporal dependencies.
2. **Dense (Fully Connected) Layer**:
   - After the LSTM processes the input sequence, the output is passed through a **fully connected (dense)** layer.
   - This layer transforms the LSTM output into the desired shape, outputting predictions for the future time steps (specified by `out_steps`).
   - The dense layer has an output size of **`out_steps * 1`**, meaning it produces one prediction per time step.
3. **Output Shape**:
   - The final output shape is **[batch_size, out_steps, 1]**, where each prediction corresponds to a future time step in the forecast. This shape is achieved by reshaping the dense layer's output.



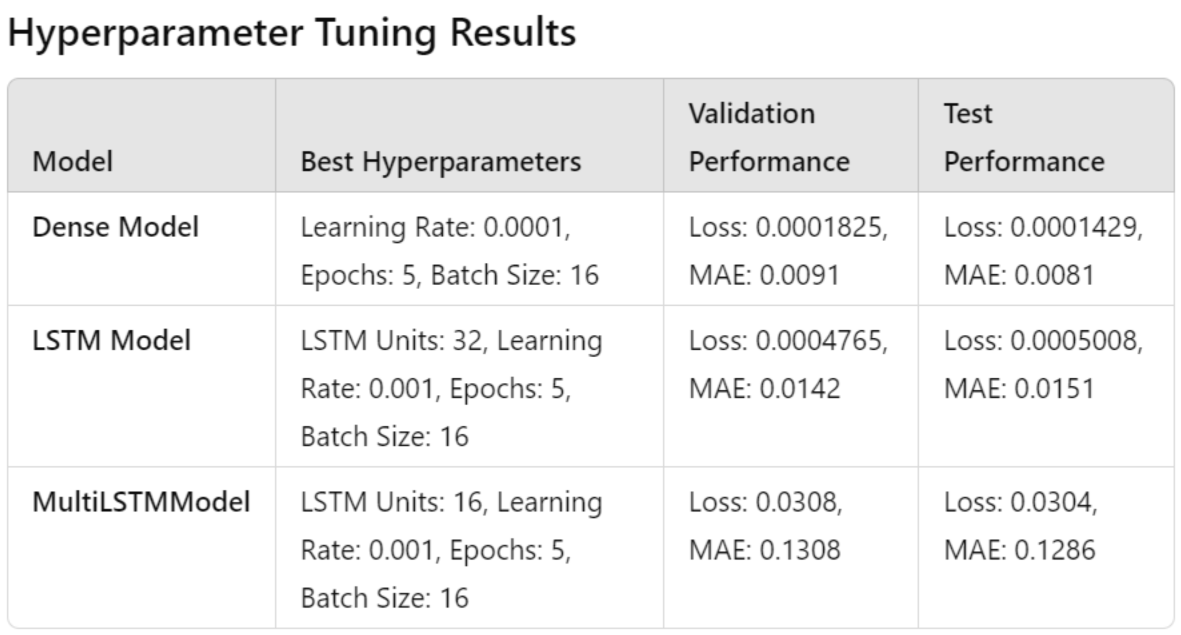
4. Testing and evaluation for multi-shot models:

Model	Validation MSE	Validation MAE	Test MSE	Test MAE
MultiStepLastBaseline	1.94	1.14	2.17	1.16
RepeatBaseline	0.12	0.26	0.14	0.29
MultiLSTMModel	0.014	0.08	0.013	0.08



5. Fine Tuning Part:

For dense, lstm and MultiLSTMModel:



Hyperparameter Tuning Results

Model	Best Hyperparameters	Validation Performance	Test Performance
Dense Model	Learning Rate: 0.0001, Epochs: 5, Batch Size: 16	Loss: 0.0001825, MAE: 0.0091	Loss: 0.0001429, MAE: 0.0081
LSTM Model	LSTM Units: 32, Learning Rate: 0.001, Epochs: 5, Batch Size: 16	Loss: 0.0004765, MAE: 0.0142	Loss: 0.0005008, MAE: 0.0151
MultiLSTMModel	LSTM Units: 16, Learning Rate: 0.001, Epochs: 5, Batch Size: 16	Loss: 0.0308, MAE: 0.1308	Loss: 0.0304, MAE: 0.1286