**Detailed Report on Temperature Prediction Using IoT Sensor Data**

**Introduction**

This notebook aims to predict future temperature values using historical data collected from IoT temperature sensors. The model leverages a Long Short-Term Memory (LSTM) neural network, which is well-suited for time series forecasting due to its ability to capture temporal dependencies.

**Data Preparation**

1. Notebook Mode Selection:
   * We have created two modes for the notebook to run, one for Colab mode, that loads the dataset from google drive link.
   * The other one is to load the dataset from local system.
2. **Loading Data**:
   * The dataset is loaded from a CSV file containing temperature readings.
   * This step ensures that the data is readily available for preprocessing and model training.
3. **Removing Duplicates**:
   * Duplicate entries are removed to ensure unique timestamps.
   * This is crucial for maintaining the integrity of the time series data, as duplicates can distort the model’s understanding of temporal patterns.
4. **Handling Missing Values**:
   * Missing values are interpolated linearly.
   * Linear interpolation is chosen because it provides a simple yet effective way to estimate missing values based on surrounding data points, ensuring continuity in the time series.
5. **Normalization**:
   * Data is scaled to the range [0, 1] using MinMaxScaler.
   * **Reason**: Normalization helps in speeding up the convergence of the neural network and ensures that all features contribute equally to the learning process. MinMaxScaler is particularly useful for scaling time series data as it preserves the relationships between values.
6. Train-Test Splits
   * We have not split the data, because we want to have more data for training. Good approach is to use splits.

**Model Architecture**

1. **LSTM Layer**:
   * LSTM is chosen for its ability to remember long-term dependencies, making it ideal for time series data.
   * **Reason**: LSTM networks are designed to overcome the limitations of traditional RNNs by using gates to control the flow of information, which helps in capturing long-term dependencies and reducing the vanishing gradient problem.
2. **Dense Layer**:
   * A fully connected layer outputs the next 10 temperature values.
   * **Reason**: Dense layers are used to map the learned features from the LSTM layer to the desired output size, providing the final predictions.
3. **Reshape Layer**:
   * Reshapes the output to match the required format.
   * **Reason**: This ensures that the output is in the correct shape for evaluation and further processing.

**Training**

1. **Optimizer**: Adam
   * **Reason**: Adam optimizer is used for its efficiency and adaptive learning rate capabilities. It combines the advantages of two other popular optimizers—AdaGrad and RMSProp—making it well-suited for optimization.
2. **Loss Function**: Mean Squared Error (MSE)
   * MSE is used, to measure difference between the actual temperature and the predicted one.
   * **Reason**: MSE is used to measure the difference between predicted and actual values. It is chosen because it penalizes larger errors more than smaller ones, providing a clear indication of the model’s performance.
3. **ModelCheckpoint**:
   * Saves the best model based on training loss.
   * We can do it for validation, but we have not split the data, because we want to have more data for training. Good approach is to use splits.
   * **Reason**: This ensures that the best-performing model during training is saved, preventing overfitting and ensuring optimal performance on unseen data.

**Evaluation**

* The model is evaluated on the training data, and the Mean Squared Error is calculated to assess performance.
* **Reason**: Evaluating the model on training data helps in understanding how well the model has learned the patterns in the data. MSE provides a quantitative measure of the model’s accuracy.

**Experiments**

* We have performed 3 experiments as following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment** | **Model Details** | **Learning Rate** | **MSE** | **Findings** |
| Experiment\_1 | 1 LSTM Layer used | 0.01 | ***0.0185*** | Lr=0.01 gives better accuracy, thus we picked it for future results. |
| Experiment\_2 | Low Learning Rate | 0.001 | 0.0188 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment** | **Model Details** | **Learning Rate** | **MSE** | **Findings** |
| Experiment\_1 | 1 LSTM Layer used | 0.01 | 0.0185 | Gives Random Results |
| Experiment\_2 | 2 LSTM Layers used | 0.01 | ***0.0181*** | Gives better learning on epochs |
| Experiment\_3 | Transformer | 0.01 | 0.0185 | 2 LSTM Layers performs better |

\*With 20 epochs

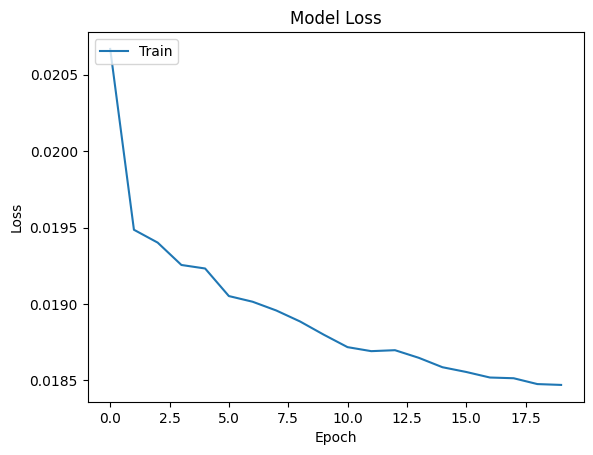


Figure: Training Loss Curve of Two layers LSTM model, on 20 epochs.

**Conclusion and Future Direction**

The notebook successfully demonstrates the use of LSTM for temperature prediction, with Adam optimizer and MSE loss function ensuring efficient training and accurate predictions. The preprocessing steps, including normalization and handling missing values, play a crucial role in preparing the data for effective modeling.

From the above experiments, we see that 2 layer LSTM performs better with learning rate 0.01.

In future, we can increase the model depth by adding more LSTM layers and we hope the model will perform better.