REPORT ON TIME-SERIES DATA

NAME: VARSHITHA GUDURU

ID: 811289917

1. How to Apply RNNs to Time-Series Data?

Because recurrent neural networks (RNNs) can remember information from earlier steps through their hidden states, they are especially well-suited for time-series data.

Data preparation: In order to meet the required input shape of RNN layers, time-series data is first preprocessed, frequently by normalization, and then reshaped (usually a 3D tensor with dimensions: samples, timesteps, features).

Model Definition: To capture temporal dependencies, RNN layers are added, such as LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit). Time-series forecasting relies heavily on the model's ability to learn patterns over time, which these layers aid with.

Training Procedure: To teach the model the dependencies between time steps, it is trained using data point sequences.

2. How to Improve the Performance of the Network for Time-Series Data?

Specifically for time-series data, there are number of techniques to enhance RNN performance:

Layer Stacking: Complex temporal relationships can be captured by employing numerous RNN layers, such as stacked LSTMs.

Regularization: To avoid overfitting, dropout layers are included. This is especially crucial in recurrent layers, where overfitting is prone to happen.

Tuning Hyperparameters: Model performance can be greatly impacted by varying the learning rate, batch size, and number of units in each RNN layer.

Early Stopping: When performance no longer improves, training is stopped. This strategy keeps an eye on validation performance during training to keep the model from overfitting.

Bidirectional RNNs: By capturing patterns in both forward and backward orientations inside the sequence, bidirectional LSTM or GRU can occasionally enhance performance.

3. How to Apply Different Deep-Learning Layers to Time-Series Data?

To improve the model's capacity to learn from time-series data, the notebook illustrates the application of multiple layers:

Dense Layers: To map the final RNN output to the intended prediction dimension, dense (completely connected) layers are added after RNN layers.

Dropout Layers: These layers, which sit between dense and RNN layers, aid in network regularization.

Conv1D Layers: To capture local patterns across time steps for some time-series issues, 1D convolutional layers (Conv1D) can be added either before or after RNN layers. This frequently improves feature extraction before feeding into RNN layers.

MODEL	HIDDEN UNITS	TRAINING MAE	VALIDATION MAE	TEST MAE
Recurrent Layer	**32**	**7.0658**	**6.6502**	**6.64**
Recurrent Layer	**64**	**0.2610**	**0.2626**	**0.26**
LSTM	**16**	**0.2526**	**0.2573**	**0.25**
1D CNN + LSTM		**0.2573**	**0.2522**	**0.26**

CONCLUSION:

The LSTM model with 16 hidden units performs the best, as seen by the table, with the lowest test MAE of 0.25, demonstrating its ability to generalize to unknown data. A recurrent layer with 32 hidden units, on the other hand, produces significantly more errors on training, validation, and test datasets. Its test MAE of 6.64 indicates that this configuration is inadequate for the task. Performance is much enhanced when the recurrent layer is increased to 64 hidden units, lowering the test MAE to 0.26. Furthermore, there is no discernible improvement over employing the LSTM alone when a 1D CNN and LSTM are combined, as evidenced by the similar test MAE of 0.26. For this dataset, the LSTM model with 16 hidden units is the best option overall since it offers the optimum balance of accuracy and generalization.