**Assignments 3 ADTA 5560 Recurrent Neural Networks for Sequence Data**

**Yog Chaudhary**

11727095

ADTA 5560 Recurrent Neural Networks for Sequence Data

Professor: Dr. Thuan L Nguyen

University of North Texas **Nov 06, 2024**

**2. PART I: Build, Train, and Test a Simple RNN on Sine Wave Data (70 Points)**

**Jupyter Notebook document Name:-** ADTA 5560 Assignment 3 - AI RNN Simple RNN with Sine Wave Data and Keras\_Yog [**http://localhost:8000/notebooks/AI%20RNN%20Simple%20RNN%20with%20Sine%20Wave%20Data%20and%20Keras%20I%20to%20V.ipynb**](http://localhost:8000/notebooks/AI%20RNN%20Simple%20RNN%20with%20Sine%20Wave%20Data%20and%20Keras%20I%20to%20V.ipynb)

**3. PART II: Write the Project Report (30 Points)**

Project Report: Building, Training, and Testing a Simple RNN on Sine Wave Data

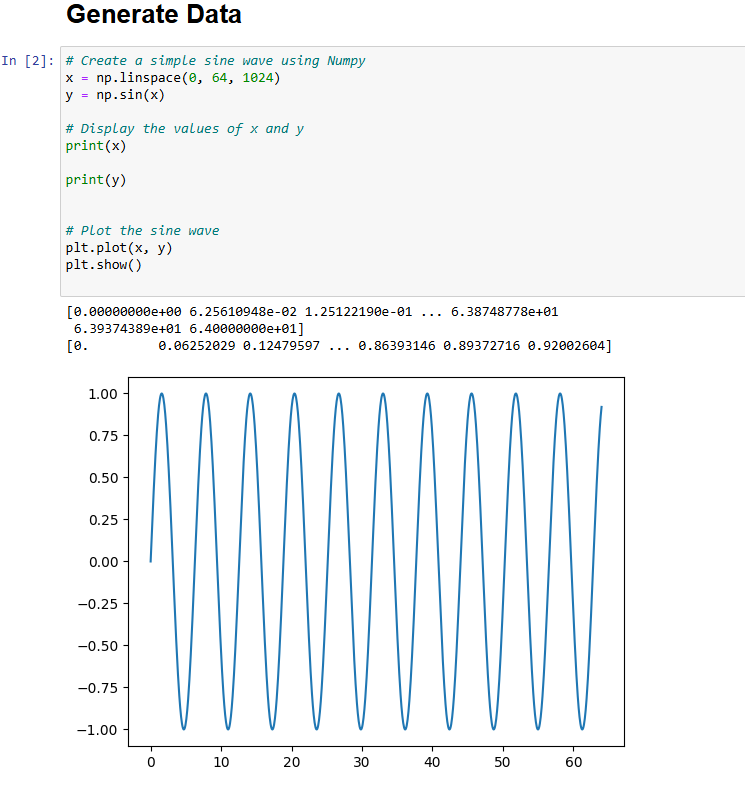
**Project Overview:**

In this Project work, we are going to build, train, and test a simple RNN model that can forecast future values of a sine wave. Sine waves are a very good example of sequence data, as they are periodic in nature, the values follow a very regular and predictable pattern making them suitable for testing an RNN's ability to learn time-series relationships

**Project Steps:**

**Step 1: Data Generation and Preprocessing**

The sine wave data was generated using NumPy, covering a range of 0 to 64 with 1024 data points. The data was split into a training set (80%) and a test set (20%), and then normalized between 0 and 1 using MinMaxScaler to improve model stability. A **TimeseriesGenerator** was used to create sequences of 50-time steps for training.



**Step 2: Model Design and Compilation**

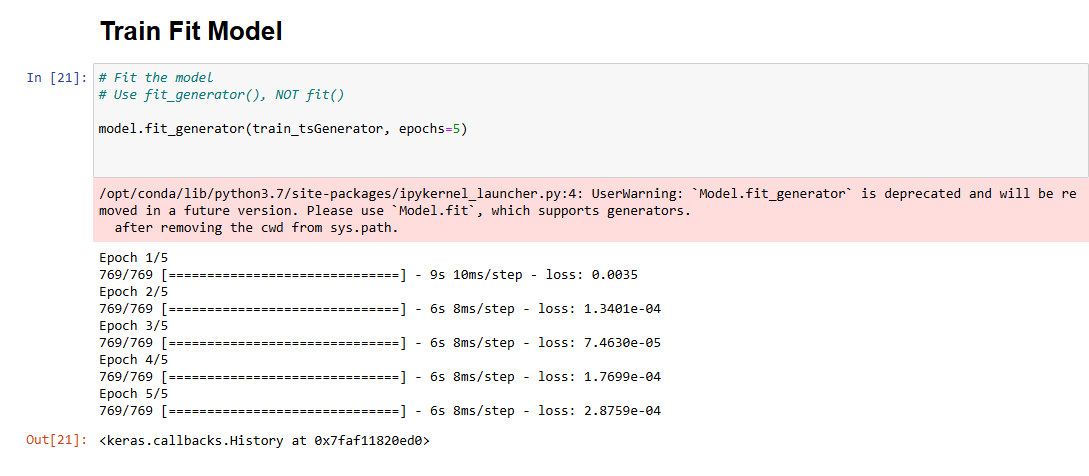
The model was built using Keras and consisted of a SimpleRNN layer with 100 units, followed by a Dense layer with one unit to predict the next value in the sequence. The model was compiled with the Adam optimizer and Mean Squared Error (MSE) as the loss function.

A screenshot of a computer program

Description automatically generated

**Step 3: Training the Model**

Training was conducted for 5 epochs, during which the model successfully minimized the loss, showing its ability to learn the sine wave pattern. The training used the fit() function with the data generated by the TimeseriesGenerator.



**Step 4: Evaluating the Model:**

The model's performance was evaluated using the test set, which had not been seen during training. An initial sequence of 50 points from the training set was used to seed the RNN, and predictions were made iteratively for the entire test set. The predicted values were then inverse transformed to their original scale. Visualization of the predictions against the actual sine wave data showed that the RNN was able to closely track the sine wave, especially in the initial parts of the sequence. However, there was a slight drift as the model continued to predict further, which is common in RNNs due to the accumulation of small errors.

**A screenshot of a computer program

Description automatically generated**

**Results and Visualization**

In order to check the accuracy of the model, the RNN predictions are plotted against real sine wave data. Key highlights from the results obtained are discussed here.

**Precision Sine:** At the start of the test set, the RNN was able to closely approximate the sine pattern. The prediction moved along the original curve, hence proving that this model can learn and generalize a pattern.

**Prediction Drift:** The model continued to make its predictions, which slightly drifted with each increase in distance into the test set. That is a well-demonstrated fact that RNNs have this problem because prediction happens sequentially, and a small error might snowball as prediction proceeds. The plotted prediction versus actual for the sine wave showed that the model learned the sine wave behavior underlaying the chart but with smaller errors on the longer-term prediction. These predictions captured the overall trend and cycle of the sine wave and, as such, were appropriate regarding the task at hand for the RNN.

A graph of a function

Description automatically generated with medium confidence

**In the last analysis,** the RNN model learned the cyclical behavior of the sine wave effectively and made accurate predictions, hence showing the strengths of RNNs in capturing temporal dependencies.

However, it was not without its limitations. for example, it entertained prediction drift in longer sequences, which might suggest that even more advanced architectures like LSTMs or GRUs would further improve performance. More epochs, hyperparameter tuning, or even regularization techniques to reduce overfitting could also be included in future enhancements. Overall, the project has been a successful demonstration of the simplicity of an RNN in the task of sequence prediction in consideration of its strengths and potential improvements.

On the whole, this was good practice in building and training an RNN on time series data to show strengths and weaknesses of the simplest RNN models for tasks of sequence predictions.

**Reference:**

**AI: RNN: Simple RNN with Sine Wave Data and Keras I**

**https://youtu.be/CcN7ucFEYIc**

**AI: RNN: Simple RNN with Sine Wave Data and Keras II**

**https://youtu.be/wFuByXsBCfQ**

**AI: RNN: Simple RNN with Sine Wave Data and Keras III**

**https://youtu.be/mD3GnwjYM\_s**

**AI: RNN: Simple RNN with Sine Wave Data and Keras IV**

**https://youtu.be/jP85vmnvQrA**

**AI: RNN: Simple RNN with Sine Wave Data and Keras V**

**https://youtu.be/hDrl5osf\_ew**

[**https://www.tensorflow.org/guide/keras/working\_with\_rnns**](https://www.tensorflow.org/guide/keras/working_with_rnns)

[**https://sams101.github.io/DataScience/keras/rnn/tensorflow/time\_series/python/2020/10/12/Keras\_RNN\_Sine.html**](https://sams101.github.io/DataScience/keras/rnn/tensorflow/time_series/python/2020/10/12/Keras_RNN_Sine.html)

[**https://machinelearningmastery.com/understanding-simple-recurrent-neural-networks-in-keras/**](https://machinelearningmastery.com/understanding-simple-recurrent-neural-networks-in-keras/)