ROUTE.AI Documentation

June 2023

1 Introduction

In the world of data analysis and predictive modeling, the ability to accurately forecast future events based on historical patterns is a valuable skill. One such application lies in analyzing route data, where understanding the movement and behavior of objects or individuals over time is crucial. In recent years, advancements in deep learning techniques, such as Long Short-Term Memory (LSTM) networks, have revolutionized the field of time series forecasting. In this project, we aim to leverage LSTM networks to predict the location of the next time steps in routes of different shapes using available datasets.

2 Model Architecture

The model architecture consists of an LSTM layer followed by a Dense layer. Following is the breakdown of the architecture and an explanation of each component.

2.1 LSTM Layer

- The LSTM layer is a type of recurrent neural network (RNN) layer that is well-suited for processing sequential data.
- The layer is added to the model using the add method of the Sequential model.
- The LSTM layer has 64 units, which determines the dimensionality of the output space.
- The input_shape parameter is set to (25, 4), indicating that each input sequence has a length of 25 time steps and each time step has 4 features.

2.2 Dense Layer

- The Dense layer is a fully connected layer that follows the LSTM layer.
- It consists of 4 units, specifying the dimensionality of the output space.

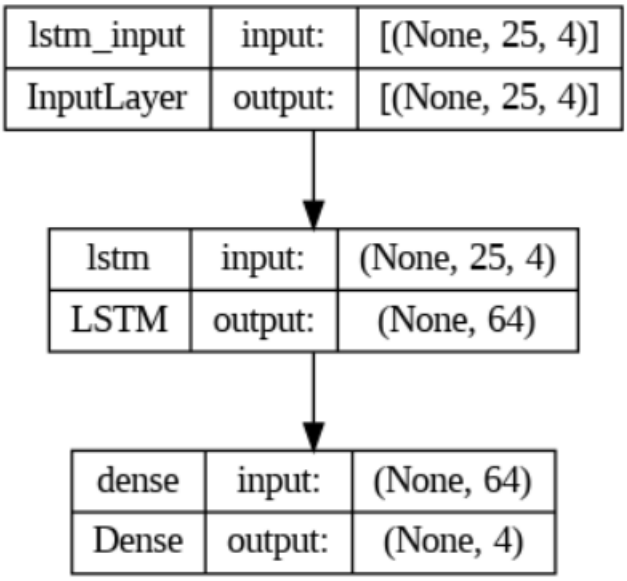

Figure 1: Model Architecture

- This layer provides the final output of the model.

2.3 Model Compilation

- The model is compiled using the compile method of the Sequential model.
- The loss function is set to 'mse', which stands for mean squared error. It is commonly used for regression problems.
- The optimizer is set to 'Nadam', which is an extension of the Adam optimizer. Nadam combines the benefits of Adam and Nesterov momentum.

Overall, this architecture takes sequences of length 25 with 4 features at each time step as input. It processes the input sequences through an LSTM layer with 64 units, followed by a Dense layer with 4 units. The model is trained to minimize the mean squared error loss using the Nadam optimizer.

3 Why This Model ?

LSTMs are particularly effective when dealing with time series data or sequences where capturing dependencies over longer time intervals is crucial. The LSTM layer in the architecture has 64 units, which determines the dimensionality of the output space. More units allow the layer to capture more complex patterns

in the data. Also, compared the MSE with different values for number of LSTM units and chose 64 finally.

Also here, 4 features are being predicted which is data related to the columns `[Longitude, Latitude, Heading, Average Speed]`, the input given will be a sequence of past 25 steps. So, the input_shape to the model is defined as (25,4).

The Nadam optimizer is utilized, as it combines adaptive learning rates and momentum to efficiently update the model's weights during training.

MSE is commonly used as a loss function where the goal is to predict continuous values. MSE penalizes large errors more significantly than small errors due to the squaring operation. LSTM models rely on the ability to calculate gradients to update the model parameters during training. The differentiability of MSE enables the use of gradient descent optimization algorithms to efficiently search for optimal model parameters that minimize the loss.

4 Working Of Code

- The necessary libraries are imported, including numpy, pandas, MinMaxScaler from sklearn.preprocessing, and Sequential, LSTM, and Dense from tensorflow.keras.
- The dataset is loaded from the CSV files named 'circle_train.csv', 'square_train.csv', 'star_train.csv', and 'real_train.csv' using pandas' read_csv function (by merging all the datasets mentioned). These CSV files should contain columns for 'Longitude', 'Latitude', 'Heading', 'Average Speed', and 'Position Date Time'. The 'Position Date Time' column is dropped since it don't affect the predictions of longitude, latitude.
- The data is normalized using the MinMaxScaler from sklearn.preprocessing. This ensures that all the features are within the range of 0 and 1, which can help the LSTM model to learn more effectively.
- The input sequences (X) and corresponding output sequences (y) are prepared using a sliding window approach. The window_size variable determines the number of previous time steps to be considered as input to predict the next time step. For each iteration, a window of window_size data points is selected, and the next data point is used as the output. This process continues until the end of the dataset.
- The input sequences (X) and output sequences (y) are converted into numpy arrays.
- The data is split into training and validation sets using an 80:20 split, where 80% of the data is used for training and 20% is used for validation.

- The LSTM model is built using the Sequential API from Keras. It consists of a single LSTM layer with 64 units, which takes the input shape of (window_size, 4) since we have four features. The LSTM layer is followed by a Dense layer with 4 units, representing the output coordinates. Here, the window_size value is taken as 25.
- After training, the model is ready to make predictions. The validation set is used to evaluate the model's performance. The mean squared error loss is computed.

5 Training The Model

The data used to train is the a merged data from of all the train csv files provided, all are merged and given as input to model as it can understand the different patterns and works on different patterns in the test data.

Used MinMaxScaler to scale input data for model as it provides normalization, aids gradient descent optimization, ensures optimal activation function performance, and establishes consistent preprocessing.

The training dataset is split into 80% for training and 20% for validation. The data is trained for 10 epochs, and during the validation process, the loss value obtained was approximately 0.0172. For every test CSV file, both Mean Square Error (MSE) and validation loss are calculated.

A window size of 25 is chosen to create input data sequences. This value is used because in the test data, there are 50 rows of data and we need to predict the next 25 values. To create these 25 time sequences, it is necessary to have a window size of 25.

6 Team Members

- Kavali Sri Vyshnavi Devi - 200010023
- Manche Pavanitha - 200010027
- Tella Rajashekhar Reddy - 200030058