LSTM approach for forecasting Monthly Mean Total Sunspot Number

Swastik Pramanik

December 2024

Contents

---

1 Abstract

This report tries to explore the use of Long-Short-Term Memory (LSTM) Neural Networks to predict sunspot activity based on a dataset of Monthly Mean Total Sunspot Numbers from "January 1749 to August 2017". The extensive temporal coverage of the dataset and the cyclical patterns make it an ideal candidate for time series forecasting. The LSTM model demonstrated its ability to capture the periodic nature of sunspot activity and provided reasonable accuracy in predicting future values.

2 Introduction

Sunspots, dark regions on the surface of the Sun, serve as indicators of solar activity and follow an approximately 11-year cycle. Understanding and predicting sunspot activity has significant implications for space weather forecasting and its impact on satellite operations, communication systems, and power grids.

This report used the publicly available dataset of Monthly Mean Total Sunspot Numbers, spanning from January 1749 to August 2017, from Kaggle to develop and evaluate a long-short-term memory (LSTM) neural network for time series forecasting.

LSTM models were specifically chosen for this task due to their ability to model temporal dependencies and capture long-term patterns in sequential data. Unlike traditional neural networks, LSTMs can retain

information over longer periods using memory cells and gating mechanisms, making them particularly suited for cyclical and periodic time-series data such as sunspot activity.

2.1 Dataset Overview

The dataset contains monthly mean total sunspot numbers over a period of nearly 268 years. Key features include:

- **Date:** Observation taken at the end of each Month from 1749 to 2017.
- **Sunspot:** Monthly mean total sunspot count.

Given the long time span and cyclical nature of sunspot data, the dataset provides an excellent basis for time-series analysis and prediction.

3 Methodology

3.1 Overview

This report proposes a LSTM framework for Sunspots time-series prediction. The framework is illustrated in Fig below.

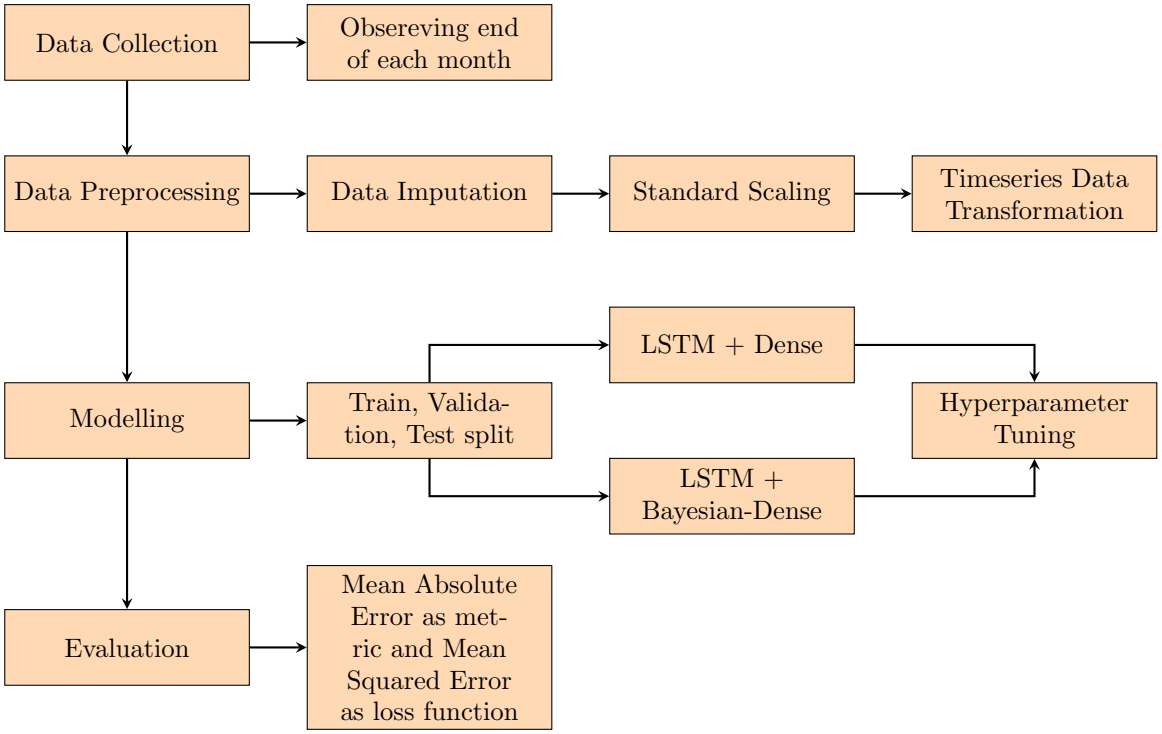

Figure 1: Visualizing Methodology

3.2 Data Preprocessing

The data set was first loaded and checked whether it had any missing values or not. The data set apparently had no missing values.

The sunspot numbers were normalized to transform the distribution to have a mean of 0 and standard deviation of 1. This is done from the equation $z = \frac{x-\mu}{\sigma}$, where x represents the original feature value, μ is the mean of the feature, σ is the standard deviation, and z is the standardized feature value. This was done using StandardScalar() in python.

This is to ensure that all the values are on same scale, preventing any single value from dominating the learning process due to its larger magnitude.

The data was split into training, validation, and testing subsets in a 72%, 8%, and 20% ratio, respectively. Sliding window sequences were created with a lookback window of 7 days, allowing the model to consider the past week's data to predict the subsequent day's sunspot activity.

3.3 Modelling

The modelling is done in 3 stages training, validation, and testing. The training set comprised of data from 1749-01-31 to 1944-03-31, the validation had data from 1944-04-30 to 1966-08-31, and the testing data from 1966-09-30 to 2021-01-31. The raw data is pre-processed, standardized, and then used to build the LSTM model.

3.3.1 LSTM+Dense

The time-series dataset was used to predict the occurrence on the 8th day using the previous 7 days data using a model that uses LSTM layer and a fully connected dense layer. LSTM was designed to process sequences of data and improved upon traditional RNN by using memory cells that can store information in memory for long series and a set of gates to control the flow of this memory information. These innovations allow LSTM to learn longer-term dependencies in sequential data.

The model was trained using an input shape of (1,7) where each sequence consisted of previous 7 days Sunspot numbers. The LSTM layer processes the input sequentially, updating its cell state and hidden state at each timestep. This allows the model to retain relevant information from previous days and discard irrelevant details through forget gates. This gave an output of 64 nodes which was then passed to a Dense layer with 64 units. Finally, the Dense output layer with 1 unit predicted the sunspot number for the next day. The model was compiled with the Mean Squared Error (MSE) loss function and optimized using the Adam optimizer. Mean Absolute Error (MAE) was used as a metric to quantify accuracy.

The model was trained over multiple epochs, with early stopping applied to prevent overfitting.

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 64)	20,224
dense (Dense)	(None, 64)	4,160
dense_1 (Dense)	(None, 1)	65

Total params: 24,449 (95.50 KB)

Trainable params: 24,449 (95.50 KB)

Non-trainable params: 0 (0.00 B)

Figure 2: LSTM+Dense Model

3.3.2 LSTM + Bayesian - Dense

In Bayesian Approximation Approach we use **probabilistic layers** which lets gather uncertainty in our prediction. These layers are layers in which weights are not deterministic rather they are random variables (normally distributed) sampled differently on each feedforward step of the network . When the layer is called (i.e., on each feedforward step), it samples a weight from the distribution and runs normally. On the backpropagation step, for each trainable random variable, the gradients are the partial derivative from the Loss in relative the mean and standard deviation.

This model uses a 'DenseVaraitional' layer from 'tensorflow_probability'. This layer uses variational inference to fit a "surrogate" posterior to the distribution over both the kernel matrix and the bias terms which are otherwise used in a manner similar to specify weights during training in gradient descent in Dense layers.

The layer fits the "weight posterior" :

$$\begin{aligned}[K, b] &\sim \text{Prior}() \\ M &= \text{matmul}(X, K) + b \\ Y &\sim \text{Likelihood}(M)\end{aligned}$$

Here we define the prior weights and biases to be normal distribution $w \sim \mathcal{N}(0, 1)$. The posterior is learned during the training process which represents the updated belief about the weights after observing the data, $w \sim \mathcal{N}(\mu, \sigma^2)$. It can be modelled as

$$\begin{aligned}p(w) &= \prod_i (w_i|0, 1), && \text{prior} \\ q(w) &= \prod_i (w_i|\mu_i, \sigma_i^2), && \text{posterior} \\ \text{KL}(q(w)||p(w)) &= \int q(w) \log \frac{q(w)}{p(w)} dw, && \text{KL}\end{aligned}$$

After calculating posterior weights Kullback-Leibler (KL) Divergence is used to regularize the posterior $q(w)$ to stay close to the prior $p(w)$. This model trains by minimizing ELBO loss function which is

$$\mathcal{L} = \text{KL}(q(w)||p(w)) - \mathbb{E}_{q(w)}(\log p(y|X, w))$$

for Gaussian like distribution. Minimizing $-$ELBO optimizes the variational posterior $q(z)$ to match the true posterior $p(z|x)$. In Gaussian likelihood $p(y|X, w)$ models the probability of the observed target y given the inputs X and weights w. PDF (probability density function) of Gaussian is

$$\begin{aligned}p(y|X, w) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \hat{y})^2}{2\sigma^2}\right) \\ -\log p(y|X, w) &= \frac{1}{2}\log(2\pi\sigma^2) + \frac{(y - \hat{y})^2}{2\sigma^2}\end{aligned}$$

Taking $\sigma = 1$ giving $-\log p(y|X, w) = \frac{1}{2}\log(2\pi) + \frac{(y-\hat{y})^2}{2}$, ignoring the constant term, minimizing the negative log-likelihood is equivalent to minimizing the Mean Squared Error (MSE). So, we can assume MSE $= -\log p(y|X, w)$. Thus the loss function becomes

$$\mathcal{L} = \text{KL}(q(w)||p(w)) + \text{MSE}$$

which is less computational for the model. The model architecture is:

- **LSTM layer:** This is a standard LSTM layer that processes sequential data. The output has 64 units.
- **DenseVariational Layer:** This is a Bayesian dense layer with 64 units. Each weight and bias in this layer is treated as a distribution rather than a fixed value.
- **DistributionLambda Layer:** The DistributionLambda layer transforms the output into a distribution: $t \sim \mathcal{N}(\text{loc}(t), \text{scale} = 1)$ This assumes the outputs follow a normal distribution with learned means and a fixed scale of 1.
- **Additional Dense Layers:** A dense layer with 64 units and ReLU activation. A final dense layer with 1 output.

3.3.3 Hyper-parameter tuning

Hyper-parameter tuning is conducted with trial and error during the training. In the experiments, adam optimizer with a learning rate of 0.001 was used for training the LSTM models, and the mean squared error was used as the loss function.

After that, the models with the selected hyper-parameters were used to forecast the Sunspots number in the testing stage. The model's accuracy was verified by comparing the measured data with real data using Mean Absolute Error (MAE).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                20224

 dense_variational (DenseVa  (None, 64)                12480
 riational)

 distribution_lambda (Distr  ((None, 64),              0
 ibutionLambda)               (None, 64))

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 36929 (144.25 KB)
Trainable params: 36929 (144.25 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3: LSTM + Bayesian-Dense Model

4 Evaluation Metric

The trained model was used to predict the test set. The predicted values in test set were compared against the actual values with Mean Absolute Error (MAE) as the metric. The formula is

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$$

where, y_i is actual value for the i-th observation, $\hat{y_i}$ is calculated value for the i-th observation, and n is total number of observations.

5 Result

The LSTM+Dense model successfully captured the cyclical nature of sunspot activity, as reflected in the overall trend of predictions. The model was trained multiple times (100 times precisely) to predict the future values and the average MAE on the test set turned out to be aprrox. 88.41024017333984 while for the LSTM+Bayesian-Dense MAE was approx. 84.49547370910645

While the model performed well in predicting the general trend of sunspot activity, deviations were observed during periods of rapid change. Also the Bayesian model did perform as well as (a bit better) the normal LSTM model according to MAE metric but it was able to capture the uncertainity and the validation loss using MSE was less in Bayesian model.

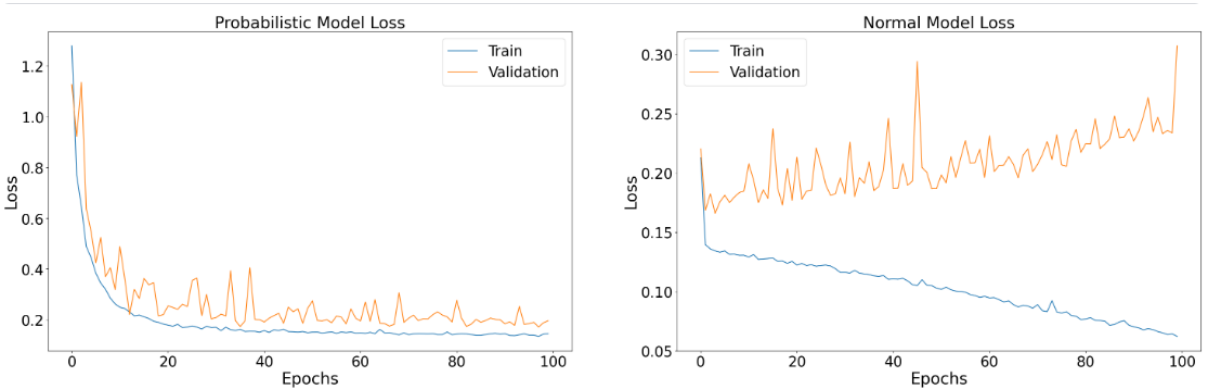

Figure 4: Shows MSE loss of models

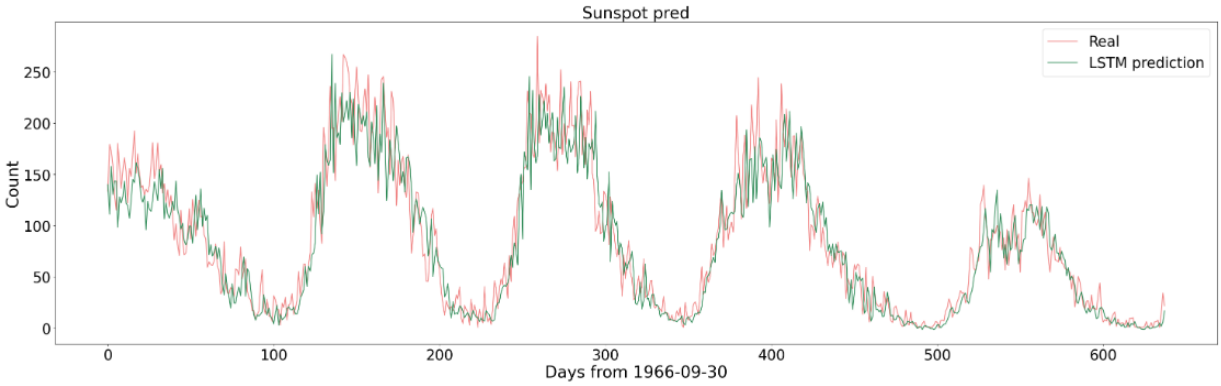

Figure 5: Showing Prediction and real values of Test data in LSTM + Dense model

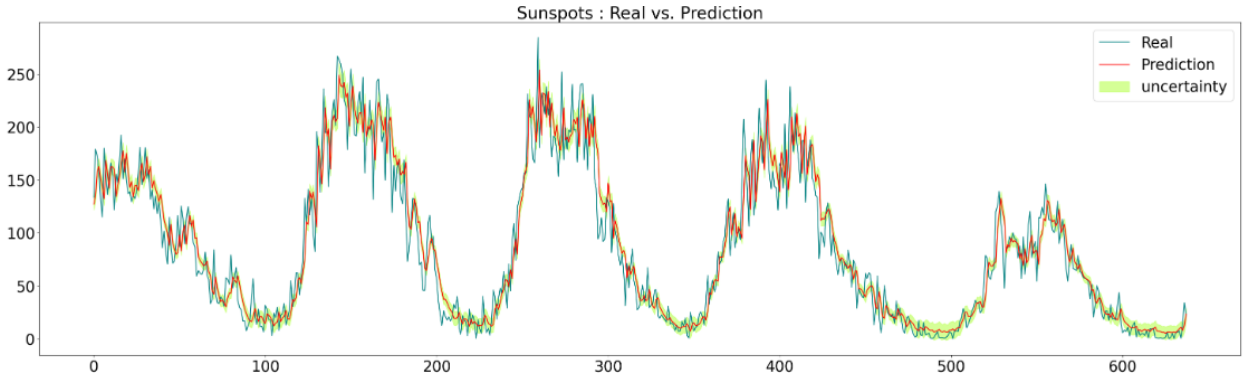

Figure 6: Showing Prediction and real values of Test data in LSTM + Bayesian-Dense model

6 Discussion

The LSTM model demonstrated its ability to model the temporal dependencies and cyclical patterns inherent in the sunspot dataset. However, some challenges remain:

- Amplitude Variations: The model's predictions for peak and trough values were not always accurate. This limitation could be addressed by experimenting with advanced architectures such as bidirectional LSTMs or attention mechanisms.
- Long-term Dependencies: The fixed lookback window may limit the model's ability to capture long-term dependencies. Incorporating features such as Fourier transforms or wavelet decompositions could enhance the model's understanding of cyclical patterns.
- External Factors: Sunspot activity is influenced by complex solar phenomena that may not be fully captured in the dataset. Including additional features or data sources (e.g., solar irradiance) could improve predictions.

7 Conclusion

The study on sunspot analysis demonstrates the application of deep learning techniques and time series analysis for understanding and predicting solar activity patterns. The evaluation results, MSE and MAE, provide insights into the model's predictive accuracy. Neural Networks do work by performing feedforward and outputing logits, which can be used as predictions or interpreted as wished. Despite being very accurate and able to model (with the right hyperparameters) any kind of mathematical function, NNs do operate with any kind of data.

Bayesian Neural Networks may be more reliable than the deterministic ones because they have an uncertainity in the weights and can avoid overfitting. For sensible decision taking, it may be the more "ethical approach", due to the possibility of avoiding taking risky decision which the model is not trained to predict. On the other side, Bayesian Networks do last more to train and to start having some good accuracy on which we can rely on

References

1. Bayesian Deep Learning - Exploration - Kaggle
2. An Approximate Bayesian Long Short-Term Memory Algorithm for Outlier Detection
3. Uncertainty in DL
4. Recurrent neural network trained with approximate Bayesian computation