Modeling Climate Change Through DNN and LSTM

Raymond Jennings, Marwan Shouery and Kalima LNU

*[*[*rj07609p@pace.edu*](mailto:rj07609p@pace.edu)*,* [*ms96172p@pace.edu*](mailto:ms96172p@pace.edu)*,* [*klnu@pace.edu*](mailto:klnu@pace.edu)*]*

***Abstract*—Reputable data sources have shown that multiple factors are increasing both the air and the ocean temperatures which are contributing to significant climate change. Many of these factors are attributed to population- specifically overall population size, urban population size, educational level, life expectancy, poverty rate and population density.  Additional attributes for which there are data include land usage types such as agriculture, farming, and forestry as well as energy consumption of both renewable and non-renewable sources. Furthermore, greenhouse gas emissions, which may contain CO2, methane, and nitrous oxide are also contributing factors. These factors are all believed to play a significant role in climate change. Through this analysis, we demonstrate how all these attributes have direct correlations to the increase in the global temperature which is a primary contributing factor to climate change. The data sources are Our World In Data (https://ourworldindata.org) and The World Bank Group (https://www.worldbank.org). The data sets are labeled by year, attribute, and country. Deep Neural Network (DNN) and Long Short-Term Memory Network (LSTM) models were built using Tensorflow and Keras to model the dataset features. Furthermore, prediction visualization was used to model climate change based on the various data attributes within the primary countries of concern.**

***Index Terms*—LSTM, DNN, RNN, long short-term memory, deep neural network, recurrent neural network, climate change, climate change factors, temperature anomaly**

I. INTRODUCTION

Long short-term memory (LSTM) layers are used to create a model for climate change data. The data used is supplied by Our World In Data [1]. The initial and primary data source was used to build a univariate LSTM based model that can make single predictions given the time-based data. Because climate change is believed to be caused by multiple sources, we also integrate the climate change factors data from The World Bank [2] to create a multivariate LSTM model that augments the primary data source. The World Bank lists 28 datasets as belonging to the climate-change category.

The climate change factor datasets include:

1. Access to electricity (% of population)
2. Agricultural land (% of land area)
3. Arable land (% of land area)
4. CO2 emissions (metric tons per capita)
5. Cereal yield (kg per hectare)
6. Electric power consumption (kWh per capita)
7. Forest area (% of land area)
8. Land area where elevation is below 5 meters (% of total land area)
9. Mortality rate, under-5 (per 1,000 live births)
10. Population growth (annual %)
11. Population living in areas where elevation is below 5 meters (% of total population)
12. poverty headcount ratio at $1.90 a day (2011 PPP) (% of population),
13. Primary completion rate, total (% of relevant age group)
14. Renewable energy consumption (% of total final energy consumption)
15. Urban population
16. Agriculture, forestry, and fishing, value added (% of GDP)
17. CO2 emissions (kt)
18. Energy use (kg of oil equivalent per capita)
19. Forest area (sq. km)
20. Methane emissions (kt of CO2 equivalent)
21. Nitrous oxide emissions (thousand metric tons of CO2 equivalent)
22. Population in urban agglomerations of more than 1 million (% of total population)
23. Population, total
24. Prevalence of underweight, weight for age (% of children under 5)
25. Renewable electricity output (% of total electricity output)
26. School enrollment, primary and secondary (gross), gender parity index (GPI),
27. Total greenhouse gas emissions (kt of CO2 equivalent)
28. Urban population (% of total population).

LSTMs were first proposed by Hochreiter [3]. The LSTM has a schematic of:

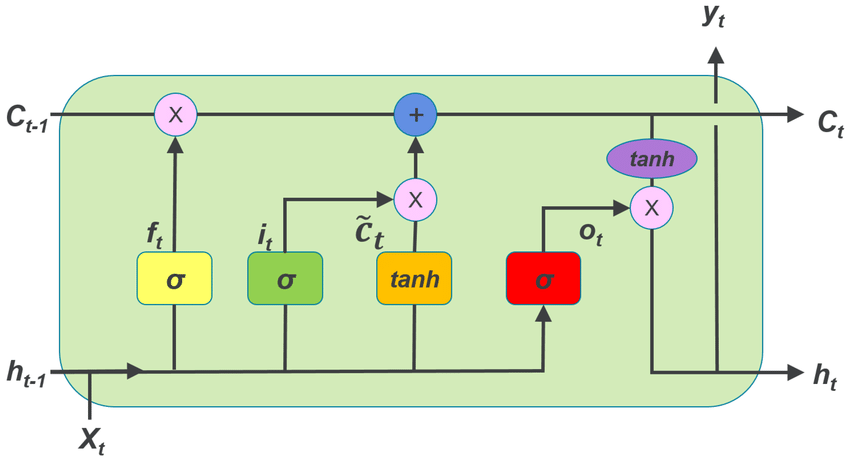


Fig 1. LSTM Schematic

Diagram

Description automatically generated

Fig 2. Vanilla RNN

LSTMs are an improvement over the Recurrent Neural Network model [4]. LSTM greatly improves on the traditional RNN by using gates to control the problem of vanishing gradient (and exploding gradient). A standard RNN consists of one layer with a *tanh* activation function. An activation function is used within a neural network to transform the data from one layer to another. LSTMs have a repeating module like RNNs but instead of one layer as in the RNN there are four layers. In Figure 1, the four layers are the four colored boxes, yellow *σ*, green *σ*, orange *tanh* and red *σ*. The line running across the top of the LSTM (*Ct-1* to *Ct*) is considered the “cell state.” The cell state runs across all of the LSTM cells and only minor changes are made to it (pink x, blue +). The four layers each contribute to either adding or removing state from an LSTM cell. The first layer (yellow *σ*) is called the *forget-gate*. The *forget-gate* outputs a value between 0 and 1. The next layer (green *σ*) is called the *input-gate* and its purpose is to decide which values will get updated. The next layer (orange *tanh*) creates new candidate values that will considered for updating the cell’s state. The new *cell-state* becomes the old *cell-state* multiplied by the *forget-gate* plus the scaled values computed by the second and third layers are added to the *cell-state*. The last step (and last layer) uses the *output-gate* (red *σ*.) The *cell-state* goes through a *tanh* function which scales the values to be between 0 and 1 and is then multiplied by the output value from the *output-gate*. There are variations on the actual implementation, gates and steps of an LSTM cell depending on who developed it.

The *vanishing gradient problem* is a common problem with RNNs. The vanishing gradient problem is encountered when an update happens to a RNNs weights. The vanishing aspect occurs when the update is proportional to the partial derivative of the error function causing the weight to remain unchanged. A frequent cause of the vanishing gradient problem is the *tanh* function which results in values in the range between 0 and 1 and backpropagation (backward propagation of errors) computes the gradients via the chain-rule. When these small numbers are used in the beginning layers of an RNN then the training process either fails or takes a very long time to converge. Various techniques have been proposed to overcome the vanishing gradient problem including avoiding using a gradient based activation function.

The number of hidden layers within an artificial neural network ANN [X] model typically defines whether a model is a deep neural network DNN [X], although there is debate on exactly how many layers constitutes a DNN this writer will use the common definition that a model must have at least two hidden layers between the input and output layers. Taking the argument further, an LSTM cell consists of 4 layers. Does that mean that any model with an LSTM layer is by definition a DNN? One may say that all of our models fall into the broad DNN definition.

II. LITERATURE REVIEW

Zhang [4] used an LSTM model for the use of predicting sea surface temperature on the coastal seas of China. They focused on the specific coastal areas given that the coastal areas have greater fluctuation in temperatures as opposed to areas further out in the ocean. In [5], [6], [7] the authors use LSTM to predict wind speed. The authors in [5] used both LSTM and bidirectional LSTM models to experiment and evaluate the prediction of stochastic wind speed to be used in a wind farm for generating electricity. In [6], the authors develop a Fuzzy-Rough-Set LSTM to predict the short-term prediction of the wind speed, again for the benefit of optimizing the generation of electricity.

III. METHODOLOGY

*Exploratory Data Analysis and Data Preprocessing:*

The primary data source [1] is given in the date range of January 15, 1880 to November 11, 2021 (as of the time of this writing.) Within this dataset the “World” entity values were used and were assigned to the the temperature anomaly column as our univariate data source. This dataset states that the temperature anomaly values are: *The combined land-surface air and sea-surface water temperature anomaly is given as the deviation from the 1951 – 1980 mean.* Based on this description of the data there has already been some pre-processing of the dataset. The date values are given as the 15th of each month from 1880 to 2021. Because the data had been pre-processed, we chose to use Scikit-Learn’s [4] *MinMaxScalar* to normalize the data between 0 and 1. Although the Keras [9] / Tensorflow [8] LSTM could handle the temperature anomaly data as is, the *MinMaxScaler* was used since it would be required once all of the datasets filled out the multivariate model.

The climate change risk data sets were given in the time range of 1960 to 2020 with one data value per year. The datasets are broken down by country, region, or world. The *World* column was used so that it would be in line with climate change temperature data from [1]. In order for the multivariate data to be considered to be valid it needs to be converted into a W x H *NumPy* [10] array. The first problem to address was that two data sources had different year ranges (1880 to 2021) and (1960 to 2020). The goal was to keep the entire to keep the long time series temperature anomalies so empty rows for the climate change factors were added so that each dataset had the same number of rows. The second problem was that the climate change factors data was yearly instead of monthly like the temperature change data. To resolve this every row of the climate change factors dataset was duplicated 12 times, taking a yearly data value and replicating as a monthly data point. The climate change factor data have slower changing values and would be difficult to measure every month. For example, mortality rate would not change greatly from month to month and would also be difficult to get an accurate value if physical surveys were needed. All of this data pre-processing was done so that the two datasets, built as *Pandas* [11] data-frames could be joined doing an inner-merge operation.

Once the dataset was run through the *MinMaxScalar*, the columns were normalized. The larger valued columns would not dominate the model’s fitting step. The climate change data had many missing data points, particularly within the early years. *NaN* values need to be removed before feeding the data to a *Keras* LSTM model, otherwise a single *NaN* value in the validation data will cause the validation loss and validation accuracy to not compute and be *NaN* for every epoch during the training phase. To work around this limitation, every *NaN* value could have been set to 0 as indication of missing data but then the model would not be aware of the data was missing or just a value of 0. The approach used was to set every *NaN* value to -1. Because the data was previously processed with the *MinMaxScaler* within the range of (0, 1) there could not possibly be a valid value of -1 within the dataset. *MinMaxScaler* will leave *NaN* values as is. With the -1 values in place, the model could now make use of the Keras *Masking* layer to ignore all values of -1 during the training process.

***Model Analysis:***

Four univariate models were built. The univariate models took the temperature anomaly dataset as input, trained on the data, and made one future prediction. The first model comprised of a Vanilla LSTM comprising of one LSTM layer and a single Dense output layer. The second model created was a stacked LSTM model where the sequences from the first LSTM layer are fed directly to the second layer with again a single Dense output layer. The third model used a Bidirectional layer. The *Bidirectional* layer takes the LSTM layer as input. The values of both directions are concatenated before passing them to the next layer. Again, a single node Dense layer was used for the output. The fourth model was a stacked bidirectional model, effectively merging the concepts of Model 2 and Model 3 together. All four models used Keras’ *EarlyStopping* callback to terminate the training period when the loss monitor of mean-square-error did not increase within 10 epochs.

Table 1. The Univariate Models Evaluation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Training | | Testing | |  |
| Model | MAE | MSE | MAE | MSE | R2 |
| *1* | 0.0400 | 0.0027 | 0.0506 | 0.0039 | 0.7095 |
| *2* | 0.0430 | 0.0031 | 0.0493 | 0.0037 | 0.7338 |
| *3* | 0.0401 | 0.0027 | 0.0481 | 0.0036 | 0.7325 |
| *4* | 0.0398 | 0.0027 | 0.0480 | 0.0035 | 0.7362 |

Chart, line chart

Description automatically generated

Fig 3. Model 1 Loss Plot

Chart, scatter chart

Description automatically generated

Fig 4. Model 1 Train/Test Plot

Chart, line chart

Description automatically generated

Fig 5. Model 2 Loss Plot

Chart, scatter chart

Description automatically generated

Fig 6. Model 2 Train/Test Plot

Chart, line chart

Description automatically generated

Fig 7. Model 3 Loss Plot

Chart, scatter chart

Description automatically generated

Fig 8. Model 3 Train/Test Plot

Chart, line chart

Description automatically generated

Fig 8. Model 4 Loss Plot

Chart, scatter chart

Description automatically generated

Fig 9. Model 4 Train/Test Plot

IV. MODEL EVALUATION RESULTS

The multivariate model performed well. The final DNN model consisted of a Masking layer masking out the missing values from the climate change factors datasets by marking them with a -1 value. The next layers within the model were three *Bidirectional* LSTM layers. Two *Dropout* [12] layers were added in between the three *Bidirectional* LSTM layers. Each of the Dropout layers used a dropout rate of 10 percent. The output layer consisted of a single cell *Dense* layer. In order for this model to train properly the key was to use the Dropout layers. Without the Dropout layers the test validation was fell off at the tail end. In typical model building there was a lot of trial and error to find the best working model. The Adam optimizer was used simply due to its all-purpose general excellent behavior. We used all default activation functions. Although we normalized the data values to be between 0 and 1 which would have made the Relu activation function a good candidate, we kept the default LSTM activation functions with the goal to see how a traditional LSTM layer would operate. One of the hyper-parameters that took some test and evaluation was the learning rate for the Adam optimizer. A learning rate of 0.00003 was found to give the best results without overfitting or underfitting. Early stopping was used again and was extremely useful by allowing the model to train until its patience level was reached. The environment used was an Apple Macbook Pro, 2.4 GHz 8-Core Intel Core i9 with 64GB of RAM. Tensorflow and Keras version 2.7.0 and Python 3.8.12.

Chart, line chart

Description automatically generated

Fig. 10. Multivariate Model Loss Plot

**Chart, scatter chart

Description automatically generated**

Fig 11. Multivariate Train/Test Plot

V. CONCLUSIONS

We created an LSTM model to predict the temperature change anomaly values as provided by [1] and [2]. We formulated techniques to working with the different data sizes and missing data limitations. By using Keras LSTM layer we were able to create a model that will make a single prediction on a monthly time granularity. We started with a univariate model that took as input the single data feature of the temperature anomaly. Different variations of the model were created including LSTM, Stacked LSTM, Bidirectional LSTM, and Stacked Bidirectional LSTM. We then created a multivariate model that took the same temperature anomaly feature plus an additional 30 features relating to causes of climate change. The univariate model used a masking layer to ignore missing data values, a bidirectional LSTM layer, a dropout layer and a dense output layer which outputs a single feature value.

REFERENCES

[1] Our Word In Data, <https://ourworldindata.org/>

[2] The World Bank, <https://data.worldbank.org/topic/climate-change>

[3] Sepp Hochreiter and Jürgen Schmidhuber, Neural Computation, Volume 9, Issue 8, pp 1735–1780, November 1997.

[4] Recurrent Neural Networks, https://en.wikipedia.org/wiki/Recurrent\_neural\_network

[4] Scikit-Learn: https://scikit-learn.org/

[5] Q. Zhang, H. Wang, J. Dong, G. Zhong and X. Sun, "Prediction of Sea Surface Temperature Using Long Short-Term Memory," in IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 10, pp. 1745-1749, Oct. 2017, doi: 10.1109/LGRS.2017.2733548.

[6] W. Yao, P. Huang and Z. Jia, "Multidimensional LSTM Networks to Predict Wind Speed," 2018 37th Chinese Control Conference (CCC), 2018, pp. 7493-7497, doi: 10.23919/ChiCC.2018.8484017.

[7] J. Li, D. Geng, P. Zhang, X. Meng, Z. Liang and G. Fan, "Ultra-Short Term Wind Power Forecasting Based on LSTM Neural Network," 2019 IEEE 3rd International Electrical and Energy Conference (CIEEC), 2019, pp. 1815-1818, doi: 10.1109/CIEEC47146.2019.CIEEC-2019625.

[8] Tensor Flow, <https://www.tensorflow.org/>

[9] Keras, <https://keras.io/>

[10] NumPy, https://numpy.org/

[11] Pandas, <https://pandas.pydata.org>

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdino, “Dropout: A Simple Way to Prevent Neural Network from Overfitting”, Journal of Machine Learning Research, June 2014, pp. 1929-1958