PREDICTIVE MAINTENANCE USING LSTM AND GRU ON NASA TURBOFAN ENGINE DATASET

Goutam Mukku

Andrew ID: gmukku

Sept 20th, 2024.

Introduction:

Remaining Useful Life (RUL) prediction is a crucial aspect of predictive maintenance. In this report,

We focus on using LSTM and GRU models to predict the RUL of engines based on sensor data. RUL prediction helps prevent unexpected failures and optimize maintenance scheduling by estimating how long an engine will continue to function before failing.

In this assignment, we use the FD001 dataset from the NASA CMAPSS Turbofan Engine dataset to predict RUL using two types of Recurrent Neural Networks (RNNs): Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU).

The objective is to compare the performance of LSTM and GRU models on the FD001 dataset. Additionally, we have applied both models to the more complex FD002 dataset as part of a bonus task and analyzed how the models handle the increased complexity.

Approach:

1) Data Acquisition and Preprocessing

Our approach is based on time-series modeling using LSTM and GRU architectures. The models are trained to learn from historical sensor data and predict the Remaining Useful Life (RUL) of engines.

We preprocess the data, engineer features, and split the data into sequences to allow the models to capture temporal dependencies.

We structured the data into 50-time step sequences, which allowed the models to track the operational behavior of the engines over time.

The goal was to predict whether an engine would fail within a certain time window (w1 = 30 cycles).

Data Cleaning:

1) Ensured no missing data exists in FD001 dataset. Filled the null values with the mean of the data in the column,

Feature Engineering:

2) Data labeling for RUL: We perform labeling for RUL column to perform binary classification and normalization.
3) Normalization: Performed Min Max scaling normalization to scale values between 0 and 1.
4) Time-series:
    a. Moving Average for sensor measurement with 3-cycle window to capture trends.
    b. Trend Features are computed to calculate difference between consecutive sensor readings
    c. Aggregation of sensor data is computed on training data to output the mean, median, standard deviation
    d. Dimensionality reduction using PCA to simplify the dataset and standardize the data.

2) Model Building

Model Development:

1) Generate Sequence for the LSTM and GRU with sequence length of 50 and create labels of the engine units, operational and sensor measurement as LSTM Input.
2) Parameters limitation to 10,000 is done and then run the LSTM model with inclusion of hyper parameters.
3) Saved the best model in the lifepath resulting in early stopping if the model doesn't improve its accuracy
4) Hyper parameters of epoch, delta and patience were changed to check the accuracy changes, resulting in an improvement of 0.4%
5) Perform the same process for GRU resulting in an improvement of 0.2%

Metric	LSTM Model	GRU Model	Comparison
Accuracy	98.16%	98.18%	GRU slightly higher
Precision	98.11%	98.82%	GRU has better precision
Recall	96.97%	97.68%	GRU has better recall

LSTM Model:

Long Short-Term Memory (LSTM) networks are well-suited for time-series prediction tasks because they are designed to capture long-term dependencies. The architecture for the LSTM model includes:

- **Two LSTM layers** with 2 units of 16 and 8 parameters each to capture temporal patterns.
- Dropout layers to prevent overfitting (with a rate of 20%).
- A final Dense layer for predicting the RUL and sigmoid activation function.
- The **Adam optimizer** and **Binary cross entropy** loss function were used for training.

GRU Model:

Gated Recurrent Units (GRU) are like LSTMs but have a simpler architecture, making them more computationally efficient. The GRU model follows the same architecture as the LSTM model, with:

- **Two GRU layers** with 20 and 10 units each.
- Dropout layers for regularization.
- The **Adam optimizer** and **Binary cross entropy** loss function were used for training.

Hyperparameter Tuning:

LSTM:

- Change in delta from 0.0001 to 0.005
- Change in Epoch from 50 to 30
- Change in patience from 10 to 5

GRU:

- Change in Epoch from 50 to 20
- Change in patience from 5 to 10

Advanced Feature Engineering:

1) Rolling Features
    a. Performed rolling statistics for Unit 5 and sensor measurement 7 and observed a decrease in sensor measurement over time in cycles.

b. Performed rolling statistics for Unit 3 and sensor measurement 4 and observed an increase in sensor measurement over time in cycles.

2) Time-based Features
    a. Performed time-based statistics for Time since the engine started and time until maintenance.

3) Model Evaluation and Visualization:

Model Accuracy over Epochs:

1) The model accuracy over epochs for both training and validation data signifies a steady increase.
2) The model loss over epochs for training and validation data observes a steady decline indicating the better performance of our model.

LSTM Model Performance:

Precision: 98.11% of the time, when the LSTM model predicted a positive case, it was correct.

Recall: The model correctly identified 96.97% of the actual positive cases.

GRU Model Performance:

Precision: 98.82% of the time, when the GRU model predicted a positive case, it was correct.

Recall: The model correctly identified 97.68% of the actual positive cases.

RUL Prediction:

- Both models perform well in predicting the Remaining Useful Life (RUL), but the GRU model tracks the actual RUL more closely, particularly in regions where the LSTM underperforms (sample indices 20-40 and 60-80).
- MSE Comparison:

The GRU model's MSE is significantly lower (≈ 0.024) compared to the LSTM model's MSE (≈ 0.042). This shows that the GRU model is more accurate in minimizing large errors in predictions.

- MAE Comparison:

The GRU model's MAE is also lower (≈ 0.045) compared to the LSTM model's MAE (≈ 0.07), further demonstrating the superior predictive performance of GRU by reducing the average prediction error.

4) Conclusion and Future directions:

GRU outperforms LSTM: Based on both visual comparison and quantitative metrics (MSE and MAE), the GRU model demonstrates superior performance in predicting RUL. It has lower prediction errors across both metrics, indicating that it provides more accurate and reliable predictions.

Future Directions:

- Use of Bi-directional LSTMs:

Bi-directional LSTMs process input sequences in both forward and backward directions, allowing them to capture more context from both past and future time steps. This could improve model performance by leveraging both past and future information to predict RUL more accurately.

- Ensemble Models:

Ensemble learning involves combining multiple models like LSTM, GRU, and possibly other models like CNNs to improve predictive performance. By averaging or voting on predictions from multiple models, ensemble methods can reduce variance and improve generalization. Stacking: A form of ensemble learning where different models are trained, and their predictions are used as input to a higher-level model, could further improve prediction accuracy and robustness.

- Incorporating Attention Mechanisms:

Attention mechanisms allow the model to focus on the most important parts of the input sequence. This could help the model weigh certain time steps more heavily when predicting RUL, improving interpretability and performance, especially in complex time-series datasets.

Appendix: Use of Generative AI

1) Code Assistance in Advanced feature engineering error resolution
2) Conceptual understanding of hyper parameters and assistance in interpretation of Evaluation metrics.
3) Report Drafting assistance
4) Calculation of parameters to limit to 10,000

We need to fill in the "Your code here" section by adding LSTM layers and Dropout layers, ensuring the total parameters are under 10,000.

Calculating Parameters for LSTM Layers

- **Parameters in an LSTM layer** are calculated as:

$$\text{Parameters} = 4 \times \text{units} \times (\text{units} + \text{input\_dim} + 1)$$

  - The factor of 4 accounts for the input gate, forget gate, cell state, and output gate.

Choosing the Number of Units

Let's choose the number of units such that the total parameters are under 10,000. Let's assume:

- **First LSTM Layer (`units_1`)**: 16 units
- **Second LSTM Layer (`units_2`)**: 8 units

Calculating Total Parameters

1. **First LSTM Layer Parameters:**

$$\text{params\_LSTM1} = 4 \times \text{units\_1} \times (\text{units\_1} + \text{nb\_features} + 1)$$

2. **Second LSTM Layer Parameters:**

$$\text{params\_LSTM2} = 4 \times \text{its\_2} \times (\text{units\_2} + \text{units\_1} + 1)$$

3. **Dense Layer Parameters:**

Calculating Total Parameters

1. **First LSTM Layer Parameters:**

$$\text{params\_LSTM1} = 4 \times \text{units\_1} \times (\text{units\_1} + \text{nb\_features} + 1)$$

2. **Second LSTM Layer Parameters:**

$$\text{params\_LSTM2} = 4 \times \text{units\_2} \times (\text{units\_2} + \text{units\_1} + 1)$$

3. **Dense Layer Parameters:**

$$\text{params\_Dense} = \text{units\_2} \times \text{nb\_out} + \text{nb\_out}$$

4. **Total Parameters:**

$$\text{total\_params} = \text{params\_LST} \downarrow \text{1} + \text{params\_LSTM2} + \text{params\_Dense}$$

The error you're seeing (`ValueError: Data cardinality is ambiguous. Make sure all arrays contain the same number of samples. 'x' sizes: 100, 'y' sizes: 93`) is caused by a mismatch between the number of samples in your input data (`x`) and the number of labels (`y`). Specifically:

- Your `seq_array_test_last` (the input data) contains 100 sequences.
- Your `label_array_test_last` (the labels) contains only 93 labels.

Causes:

This mismatch likely occurred because some of the engines in the test set have fewer than `sequence_length` timesteps, and the sequence padding logic you applied may not have been applied uniformly. There might also be a diffe↓ .e in how you're selecting sequences versus labels.