Predicting Machine Failures in Toothbrush Tufting Machines based on Temperature, Shock, and Vibrations Data

Bill Jiao

May 7, 2024

Abstract

This project proposes a machine-learning method to optimize dental industry manufacturing systems, specifically for toothbrush tufting machines. The project optimized a Long Short-Term Memory Model to generate machine states predictions based on vibrations, shock, and temperature data. To achieve improved performance, the project adjusted learning rates, training strategies, training loss functions (and weights). The model achieved a loss value of 1.1462 on the regular cross entropy loss function and a loss value of 0.1629 on the binary cross entropy function. On the testing set, the cross entropy loss trained model achieved a macro F1 score of 0.3342, a micro F1 score of 0.9081, and a weighted F1 score of 0.9383. This means that the model does a good job creating general predictions and correctly identifies normal operating states. However, the model does not do as well at failure state predictions.

Github: https://github.com/BillJiao/SP2024-DS598-Project.git

Introduction

Smart factories optimizes their manufacturing systems with existing data. In regards to this field, predictive maintenance, powered by smart sensors and data analytics, is another trend that minimizes downtime by predicting equipment failures before they occur. In the oral health industry, Toothbrush tufting machines are used to insert bristles into the toothbrush, forming the brush portion. However, because of their constant usage, many machines experience regular wear and tear, resulting in serious issues in their motor gear box. Currently, most factories rely on experienced engineers to conduct manual checks to ensure the reliability of machines. If user data is effectively utilized, however, we would not only be able to send alerts to engineers with real-time data but also generate machine failure predictions based on existing trends. To address this need, this paper will propose and train several established machine learning models on multivariate time-series data.

Related Work

The concept of predictive maintenance has been an active discussion with modern manufacturers. Therefore, after machine learning techniques became more widespread, many researchers tested a variety of models to optimize different aspects of manufacturing. For example, Jansen et al. [1] presented a workflow for cleaning, preprocessing, and using the data to train two models: one based on Convolutional Neural Networks and one based on Recurrent Neural Networks. The study covers their entire procedure, from data sampling to data processing and model training/evaluation. Other researchers suggested hybrid approaches. In their research, Hsu et al. [2] suggested a temporal convolution-based long short-term memory network, where temporal convolutional networks are used for feature extraction while LSTM and attention layers are used to learn temporal relationships among the features. Similarly, Wahid et al. [3] proposed a hybrid CNN - LSTM framework for multivariate time-series forecasting. In their system, the LSTM was used to analyze relationships among different time-series data variables through its memory function, and CNN was used to extract high-level features from the data.

Model Architecture

Long Short-Term Memory, or LSTM, networks were used in this study. The main architecture of this LSTM includes one input layer, one LSTM layer, one linear output layer, and a sigmoid function as the activation.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

LSTM is chosen here because like all recurrent neural networks, their memory abilities make them better suited for time-series data training. Moreover, LSTMs are designed to mitigate the vanishing gradient problem in traditional RNNs. Moreover, the incorporation of the sigmoid activation function will ensure that we are returning the predicted probabilities for each class.

To address the training curve shifting rapidly, a learning rate scheduler was added to the LSTM where the learning rate is cut from 0.0001 by 10 for every 200 epochs of training. This ensures that whenever the training process reaches a local minimum, the training process will slow down and carefully reach the real minimum loss.

The cross entropy loss function is used for the architecture because it effectively computes the loss for the probability predictions generated for each class by the LSTM. Moreover, the cross entropy loss function is particularly suited for our problem, which involves generating predictions for multiple classes.

As the dataset section will further discuss, the dataset is uneven in terms of classes, meaning that some non-Operating statuses will be more common than others. To address this issue, the cross entropy loss function is complemented with a tensor that weights each class based on the inverse of its prevalence in the dataset. This way, common classes will be weighed less and less common classes will be weighed more.

Datasets

The dataset is taken from a toothbrush manufacturing company on one of its toothbrush tufting machines. The data is taken over a 20-day period collected over 10-second intervals. Each data point contains 7 variables: temperature (degrees Celsius) and 3-dimensional (x, y, z) vRMS vibration velocity (m/s) and shock/noise frequency data (Hz) data. In total, there are 727,733 datapoints in the dataset. The set also includes 5 different types of machine states (Operating, Nozzle Cleaning + Needle Protection, Unknown, Bent Metal Wire, Turntable Fixture without Handle) set on the same time scale.

After two weeks of exploratory data analysis, I realized that there are many gaps within the data. Specifically, there is a large empty block of data where all values are nan between September 29, 2023 to October 4, 2023. This type of gaps also occur in many smaller intervals within the dataset. To address this issue, I divided the data into two parts from the beginning of the largest nan gap to the end of the gap. To address smaller nan values, I used a column mean for all missing numeric values and assigned the state "Unknown" to all missing machine states values. This way, my one dataset of 20 days is divided into two datasets of approximately one week each.

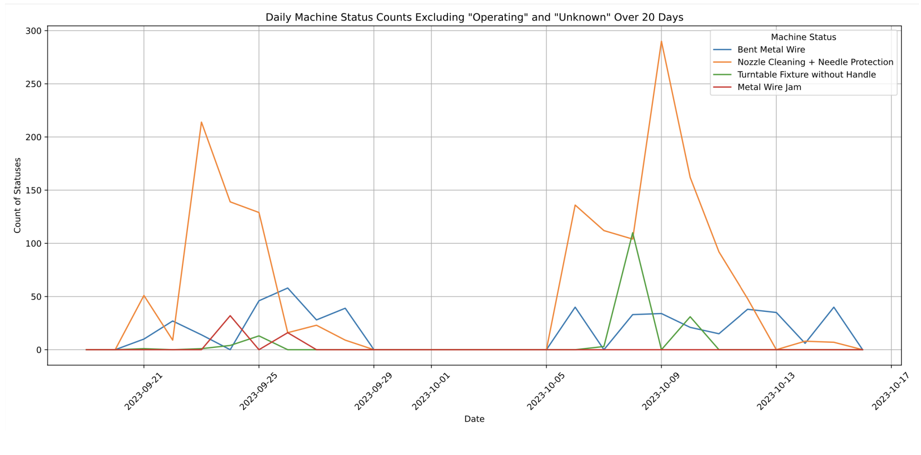

Figure 1: Failure States Distribution Across Entire Dataset

My analysis has also shown that among the two cleaned datasets, the failure counts of different types of machine failure seem to be roughly evenly distributed. Among the different types of failures, Nozzle Cleaning + Needle Protection was the most common, followed by Bent Metal Wire and Turntable Fixture without Handle. More specifically, the following is a machine state distribution of my second dataset, which was used in all training runs.

State	**Count**
Operating	85738
Unknown	2712
Nozzle Cleaning + Needle Protection	955
Bent Metal Wire	253
Turntable Fixture without Handle	144

Table 1: Counts of different machine states

For the first training run, I used randomly selected sequences. To account for the low distribution of failures among the two datasets, I chose random sequences of 100 data points (10s each) for my LSTM. This form of sequence selection resulted in good predictions for normally operating states, but failed at predicting non-operating states.

For the second training run, I divided the dataset into 2-hour sequences where each sequence ends with a failure state that is not unknown. This way, the sequences will more effectively target the non-operating states, which are what the model will be trying to predict.

Evaluation

To evaluate the training results, two loss functions: cross entropy loss and binary cross entropy loss were used. Binary cross entropy loss was used first to account for high loss values in the cross entropy loss function, and on the model trained with randomly selected 100-datapoint sequences, the loss value over 100 epochs was 0.1629. All other tests were conducted on the regular cross entropy loss function. Over 1000 epochs, the LSTM model trained on 2-hour sequences anchored with a failure state resulted in a cross entropy loss of 1.1462. Originally, the training curve had many random hikes whenever a local minimum was reached. Therefore, in following training runs, a learning rate scheduler was added to divide the learning rate by 10 from 0.0001 for every 200 epochs. This effectively addressed the problem of the loss hikes.

To evaluate the success of the model, I will use a F-1 score test to take into account both precision and recall. Previous tests done by professionals on the same dataset yielded a precision rate of around 80 percent, so if my test also takes into account the recall, I would expect a F1 score that is around 0.7.

For the randomly chosen 100-datapoints sequences, after around 800 training epochs of my

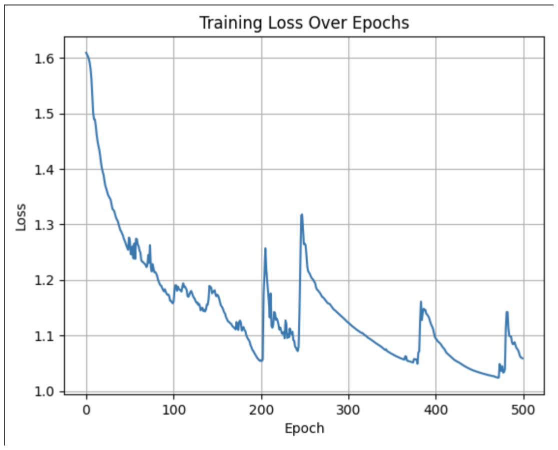

Figure 2: 500 Epoch Training Without Learning Rate Scheduler

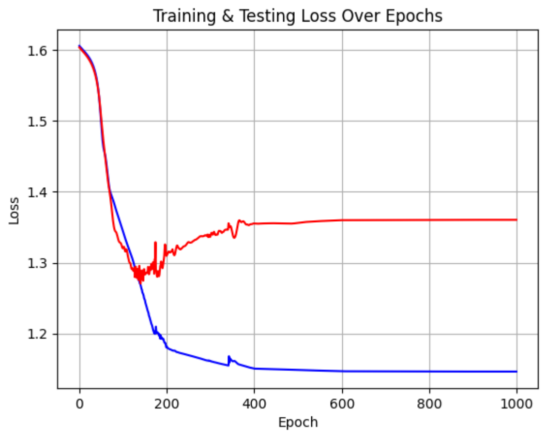

Figure 3: 1000 Epoch Training With Learning Rate Scheduler

LSTM with cross entropy loss, adam optimizer with learning rate set to 0.0001, the model was evaluated on three different F1 score metrics. The model scored a 0.6446 in the micro F1, which accounts for all true and false predictions, 0.1656 in macro F1, which treats all classes as independent, and 0.9811 on the weighted F1, which weights each class based on their prevalence. This means that although the LSTM is able to very accurately predict when machine operations (which is the most common class) happens, it is not as good as predicting rare cases of failures, since it scored a low macro F1 score.

For my 2-hour sequences that end with failure states, I did two training runs, one with 500 epochs without a learning rate scheduler, and one with 1000 epochs with the learning rate scheduler. For the higher epoch training run with 1000 epochs, the model scored a 0.3342 on the macro F1, 0.9081 on the micro F1, and 0.9383 on the weighted F1. The rise in macro and micro F1 score means that the adjustment with sequence selection was able to improve the model prediction accuracy on non-operating states. However, this comes at the cost of more inaccuracies when predicting operating states, which can be shown in the lower weighted F1.

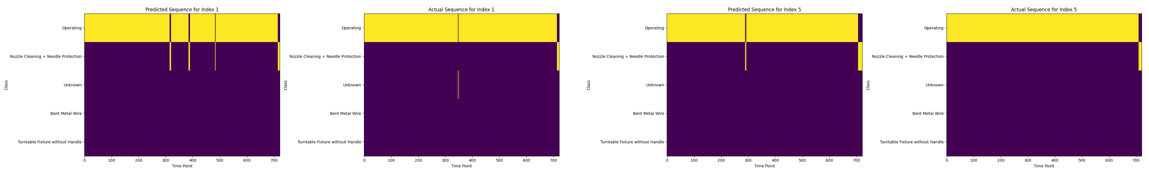

Figure 4: Model Prediction Results on Testing Set Sequences (Y-axis is machine status, X-axis is time, left image is predicted, right is true)

Conclusion

Overall, I learned a lot about the implementation of an LSTM, data preprocessing, and data visualization from this project. At the beginning of the project, I had high expectations and assumed the most time consuming part of the project to be finding effective model architectures and training the model. However, understanding the data, addressing issues, and creating different sequences for my LSTM actually required more of my time than building the model itself. Especially because I am working with real data, immersing myself in the data and gaining an understanding of how the features related to the machine states benefited my model training immensely as well. Next steps for this project would include improving the model complexity to achieve improved training and testing results, experimenting with different architectures, and potentially using a larger dataset to improve the model performance.

References

[1] Femke Jansen. *Predicting machine failures from industrial time series data.* IEEE, 2018.

[2] Chia-Yu Hsu. *Temporal Convolution-Based Long-Short Term Memory Network With Attention Mechanism for Remaining Useful Life Prediction.* IEEE, 2022.

[3] Abdul Wahid. *Prediction of Machine Failure in Industry 4.0: A Hybrid CNN-LSTM Framework.* Applied Sciences, 2022.