Predictive Maintenance with the AGT-1500 Automotive Gasoline Turbine Engine

Christopher L. Kehl
Department of Computer Engineering and Computer Science
University of Louisville J.B Speed School of Engineering

Abstract

In 1965 the U.S. Army Tank and Automotive Command (TACOM) awarded the contract for development of the Automotive Gasoline Turbine Engine known as the AGT-1500 [1]. The AGT-1500 today still is driving force of the 70-ton M1 Abrams Tank. This engine was first battle proven in 1990 with the event of Operation Desert Storm and is still being used with the ongoing conflict in the Middle East.

The purpose of this experiment is to research new methodologies for fault detection in the AGT-1500. The current method is to wait for a fault to appear and then react to that fault. This type of methodology generally leaves the tank inoperable due to engine malfunctions. This experiment implements a technique known as predictive maintenance, which in terms uses algorithms applied to the engine parameters to predict the probability of an engine malfunction at a certain time in the future.

We use a deep-learning technique with Recurrent Neural Network (RNN) to predict future T7 temperatures from the automotive gasoline turbine engine (AGT-1500) used to power the M1 Abrams Main Battle Tank. The concept feeds the engine parameters obtained by the engine memory unit into a recurrent neural network (RNN) to predict the next T7 parameters that will be generated by the AGT-1500 engine. This experiment uses a simple LSTM model to predict the future T7 temperature with a 99% accuracy.

Keywords

EMU - Engine Memory Unit

INTRODUCTION

After World War II the military realized that the gas turbine engine with its lower weight, greater power, and mechanical simplicity, had a clearer advantage over the aircraft reciprocating engine [1]. With this realization the U.S military soon adopted the turbine jet turbine engine for use in all it's aircraft. In the early days just after World War II, the turbine engine began to replace steam powered applications used by the U.S Navy and commercial applications [1]. In addition, the turbine engine was introduced as replacements for steam powered pumping stations and used for power generation [1].

From this, trend the turbine engine with its light-weight application began to spark ideas for the use of these engines in armored land vehicles. In 1950 the Soviets began to experiment with turbine engines in armored tanks [1]. The Swedish adopted the turbine engine as the main powertrain in the early 1960's [1].

Around the same time the U.S Army was designing a replacement for it's M60 Patton Main Battle Tank (MBT). The newly designed prototype was known as the MBT-70, which was the foundation of the M1 Abrams Main Battle Tanks that the U.S. Army uses today [1]. In 1965 the U.S. Army Tank and Automotive Command (TACOM) awarded the contract for development of the AGT-1500 [1]. The AGT-1500 today still is driving force of the 70-ton M1 Abrams Tank. This engine was first battle proven in 1990 with the event of Operation Desert Storm and is still being used with the ongoing conflict in the Middle East.

The purpose of this experiment is research new methodologies for fault detection in the AGT-1500. The current method is to wait for a fault to appear and then react to that fault. This type of methodology generally leaves the tank inoperable due to an engine malfunction. This experiment implements a technique known as predictive maintenance, which in terms uses algorithms applied to the engine parameters to predict the probability of an engine malfunction at a certain time in the future.

This experiment uses deep learning techniques with Recurrent Neural Network (RNN) to predict future T7 temperature start sequences from an automotive gasoline turbine engine (AGT-1500) used to power the M1 Abrams Main Battle Tank. The concept feeds the engine parameters obtained by the engine memory unit into a recurrent neural network (RNN) to predict the next parameters that will be generated by the AGT-1500 engine. Once the RNN is trained and tested, the predicted parameters will be analyzed to determine if the next parameters generated will result in a healthy or unhealthy engine.

Future research with this application will be used to make a prediction on which component or components are the cause of any faulty engine parameters that are obtained. Currently, the data provided does not provide is not sufficient to predict individual component failure; however, the means to gather the data is obtainable, and plans for further research working with artificial neural networks in the realm of predictive maintenance will continue.

Case Study

Advances in technologies such as deep learning algorithms, advancement in the internet of things, and big data are allowing us to mine data out of devices that seemed impossible a decade ago. With the applications of new technologies predictive maintenance is now has the potential to become a reality [2].

Currently a fleet of vehicles in Gatineau, Canada is being used to conduct research by monitoring the fleet in real-time to predict their remaining useful life, which will help companies lower their fleet management costs by reducing their fleet's average vehicle downtime [2]. The approach that is being used is known as the Consensus self-organized models (COSMO) approach [2].

The COSMO is a predictive maintenance system for a fleet of public transport buses, which attempts to diagnose faults that deviate from the rest of the bus fleet. The research is currently being conducted which implements the use of a Raspberry Pi, which acquired sensor data from a STO hybrid bus by reading from a J1939 network, the SLN was implemented using a laptop, and the RN was deployed using meshcentral.com [2]. The present work proposes a fleet-wide unsupervised dynamic sensor selection algorithm, which attempts to improve the sensor selection performed by the COSMO approach [2].

For our predictive maintenance model, the data that is collected from the AGT1500 is in the form of parameters stored in a device mounted onto the engine called the engine memory unit (EMU). Quarterly, the parameters are retrieved by field service representatives whom then label and analyze the data. The data used during this experiment contains 24 parameters, 5640 rows. The Date Time observation that we are going to use for simplicity for this experiment will be set up to record every 10 minutes. Our data will allow for you 6 observations per hour. Therefore, a single day will contain 144 (6x24) observations. The dataset will contain 39.2 days of data (5460 / 144).

To begin the experiment, I will use the parameters for the Date Time and the T7 temperature. By only predicting the T7 temperature allows for a simple exploration of predicting future parameters successfully for the initial phase of this experiment. Future experimentation will be performed that will allow for the prediction of all 22 parameters enabling an accurate diagnosis on how the engine will perform at a given point in the future.

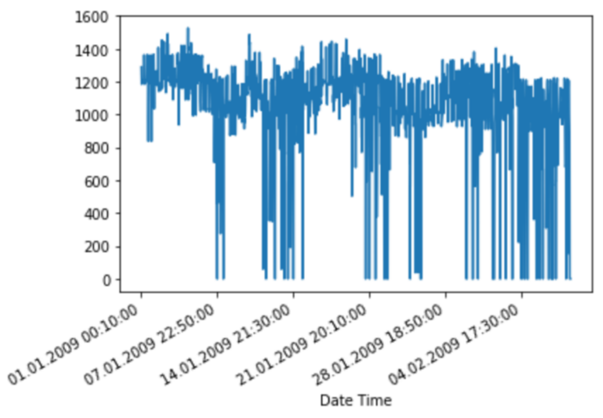

Figure 1. Plot of the T7

Figure 1. shows the Date Time and the T7 temperature. The T7 temperature is the temperature read by the T7 temperature thermal coupler which is positioned at the power turbine inlet. The T7 temperature is crucial in determining the scheduling of fuel to the digital electronics unit. In addition, the T7 temperature reading is an important parameter when performing engine diagnostic health checks, so it is an excellent starting point to predicting future readings. The range of the T7 temperature is between 0 and 1526.75 °F.

Once we have our Date Time and T7 temperature we will standardize the data with a scaling method of subtracting the mean and dividing by the standard deviation of each feature [3]. To begin work with our model we will create the data for the univariate model. Our model will be given the last 20 recorded temperature observations and will predict the temperature at the next time step.

Solution

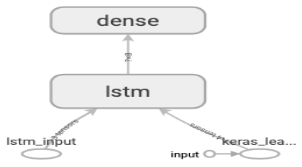

Figure 2. LSTM model

For this experiment we are using a deep-learning model to predict future T7 temperatures. As shown in figure 2, the model is composed of a Recurrent Neural Network (RNN). The RNN is a type of neural network well-suited to time series data [3]. RNNs process a time series step-by-step, maintaining an internal state summarizing the information they've seen so far [3]. An RNN is a type of neural network that has an internal loop. According to Chollet, "The state of the RNN is reset between processing two different, independent sequences so you still consider one sequence a single data point: a single input to the network [4]." He goes on to explain, "what changes is that this data point is no longer processed in a single step; rather, the network internally loops over sequence elements ", as shown in figure 3 [4].

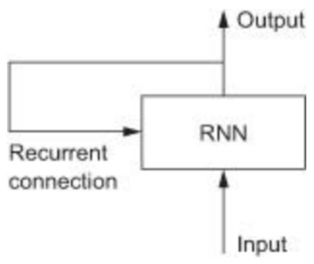

Figure 3. basic RNN network [4]

The model we will be using is a you use a specialized RNN layer called Long Short-Term Memory (LSTM). The LSTM layer is a variant of the Simple RNN layer [4]. The LSTM layer adds a way to carry information across many timesteps. Chollett makes the analogy of a conveyor belt running parallel to the sequence you're processing. The Information from the sequence can jump onto the conveyor belt at any point, be transported to a later timestep, and jump off, intact, when you need it [4]. Chollett further explains the LSTM saves information for later, thus preventing older signals from gradually vanishing during processing, in the case of the vanishing gradient.

To start our model, we develop a simple regression baseline model. Our baseline model consists of the last 20 recorded temperature observations [3]. The goal will be to learn to predict the temperature at the next time step. The baseline provides us with a beginning point to start with, this will be compared with to RNN to see if there is an improvement, or if it is more beneficial to use a simpler model such as the regression model.

Once our baseline model is built, we begin to develop our RNN. To begin we set a batch size of 10 and a buffer size of 100. We split the dataset using 80 % of our data for the training set and 20 % of the data for our validation set. For the architecture of our model we feed our data, which at this point are turned into tensors, into an 8-layer LSTM. The LSTM feeds its results into a single dense layer. The optimizer that is used is the adam optimizer and the metrics we evaluate the loss is the Mean Squared Error (MSE). We epochs to 15 and our evaluation interval to 200. In addition, we set our validation steps to 50.

The MSE assesses the quality of a predictor, or an estimator. In our case it's the quality of the predictor. To explain the MSE, we have a vector of predictions which is generated from a sample of *n* data points on all variables, and is the vector of observed values of the variable being predicted, with being the predicted values then the within-sample MSE of the predictor is computed as

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

We chose the adam optimizer because this method combines momentum and RMSprop (root mean squared prop). Adam is the acronym for adaptive moment estimation. The RMSprop smooths the gradient. Mathematically the RMS prop can be explained by first calculating the gradient.

$$S_{dw} + (1 - \beta_2)dw^2$$
$$S_{db} + (1 - \beta_2)db^2$$

Gradient in RMSprop

We add the weights and the bias matrix as:

$$w := w - \alpha \frac{dw}{\sqrt{S_{dw} + \epsilon}}$$

$$b := b - \alpha \frac{db}{\sqrt{S_{db} + \epsilon}}$$

Weight and bias update in RMSprop

In this formula we notice that beta2 is a new hyperparameter. Its epsilon is a very small value which prevents it from being divided by 0. We combine the momentum and RMSprop using adam. With the adam optimizer we introduce four hyperparameters:

- Learning rate alpha
- beta from momentum
- beta2 from RMSprop
- epsilon

Our advantage is that we will not have to tune the beta, beta2, and epsilon.

Results

For Figure 3. We plot our baseline by drawing a plot of the past 20 observations and plot our target destination to determine where we want to go.

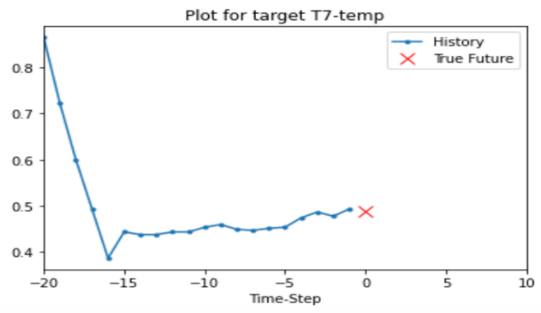

Figure 3. Target T7 Temperature

The plot shows the red X at time-step 0. This is our destination that we want to predict. The X is the actual true reading from the data.

After running our simple regression baseline predicting the next time-step we get the following plot.

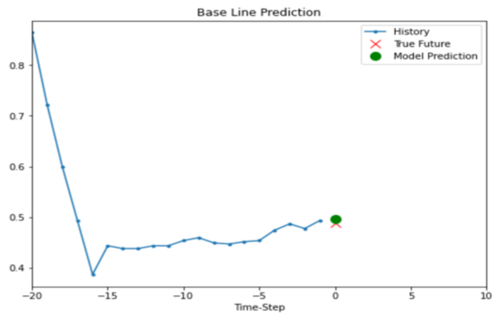

Figure 4. Predicted baseline T7 Temperature

We can tell from model prediction was 0.488 and our target is 0.493. The simple regression model leaves us with a 98% accuracy.

When running our simple LSTM model with 15 epochs we have an end MAE loss of 0.2050 and a val_loss of 0. 3627. Figure 5 shows a graphic representation of the loss function.

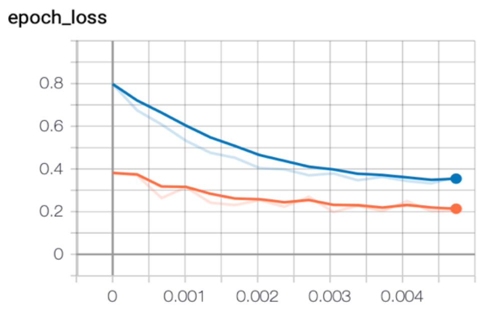

Figure 5. Simple LSTM model epoch loss

Figure 5. Epoch Loss simple_lstm_model

When we plot the results of the simple lstm model we set the model to perform 3 takes. Figure 6 - 8 shows the results for our simple lstm model. The first and second take produces much greater results than our baseline (99%); however, our third take does not perform well.

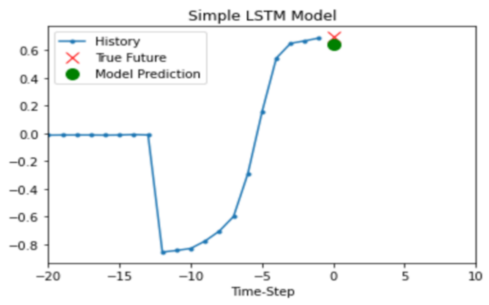

Figure 6. Simple LSTM model predicted take 1

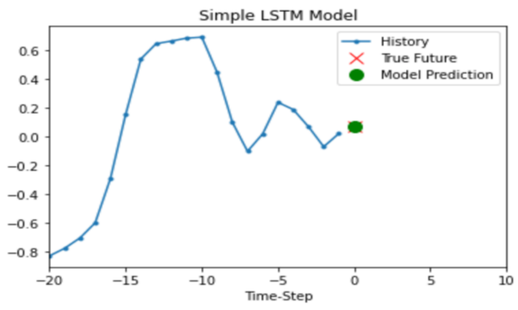

Figure 7. Simple LSTM model predicted take 2

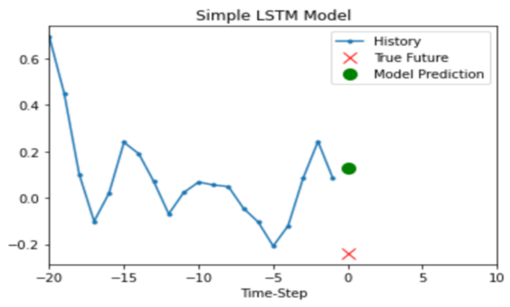

Figure 8. Simple LSTM model predicted take 3

Conclusions

The baseline model using a basic regression model resulted in an accuracy of 98%, our simple LSTM model produced results that were 99%. Further research is being conducted using multivariate samples of the current data used for the simple lstm model. For this model we are stacking a 32-layer LSTM with a 16-layer LSTM followed with a 72-layer Dense layer. The multivariate multi LSTM model currently not producing significant results. With further research and testing the model will be able to accurately predict multiple AGT-1500 parameters.

REFERANCES

[1] Honeywell, *AGT1500 Familiarization Course Study Guide,* Phoenix, AZ: Honeywell, 2018.

[2] P. Killeen, "uO Research," 17 01 2020. [Online]. Available: https://ruor.uottawa.ca/handle/10393/40086.

[3] Martín Abadi. et, al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 01 04 2020. [Online]. Available: https://www.tensorflow.org/. [Accessed 20 04 2020].

[4] F. Chollet, Deep Learning with Python, Shelter Island: Manning Publications, 2018.

[5] alisina, "Kaggle," 16 February 2020. [Online]. Available: https://www.kaggle.com/alisina/tanks-images-cmpt-726-assignment-2-fall-2019/discussion/130851.

[6] Dasgupta, "Time-Series Analysis Using Recurrent Neural Networks in Tensorflow," 28 June 2018. [Online]. Available: https://medium.com/themlblog/time-series-analysis-using-recurrent-neural-networks-in-tensorflow-2a0478b00be7.