Estimation of High Energy Steam Piping Degradation Using Hybrid Recurrent Neural Networks

JL van Niekerk[a], PS Heyns[a], MP Hindley[b], C Erasmus[c]

[a] *Center for Asset Integrity Management, University of Pretoria, Pretoria, South Africa*
[b] *Plant Engineering, Ultra Safe Nuclear, Lynnwood, Pretoria, South Africa*
[c] *Research Testing and Development, Eskom, Johannesburg, South Africa*

Abstract

The degradation of high energy piping systems is very complex to simulate due to the many variables that influence the useful lives of such systems. Estimation of the extent of degradation is however important in the maintenance planning process. In this research the use of data driven machine learning techniques to deal with this complex problem is investigated. A hybrid recurrent neural network is created that consists of a combined recurrent neural network and a feed forward neural network. The hybrid model is trained on historical data that has been captured over a six-year time period. The following variables related to piping system components are used as inputs to the machine learning model: operating temperature and pressure time history, distance to the closest anchor point, distances to neighbouring supports, elevation survey readings, as well as the last known creep damage measurements on the component. The model is created in Python using the Tensorflow library. A recurrent neural networks (RNN) is employed, namely the gated recurrent unit (GRU). The adaptive movement estimation optimization algorithm, called Adam, is used to optimize the machine learning model. The trained model is able to predict the degradation classification of a component with an accuracy of up to 92% on the training dataset and up to 55% on the validation data set. When using this model to predict components with high creep damage, more than 400 voids per mm^2 a hit rate of 25% is achieved. The current system employed at operating power stations shows a historic hit rate of 14%. This is a significant increase in performance and could be used to compile more efficient inspection plans. The model is successful in recognising patterns within the data and offers an automated way to parse large data sets that consist of a temporal and static data mixture simultaneously. Conventional data driven models are only able to look at either temporal data or static data. This suggests a generic approach to make objective decisions on similar complex data driven problems and its application is not limited to this particular problem. The methods applied in this research are expected to perform even better on problems where the frequency of data collection is higher than what is used in this research.

Keywords: Machine Learning, High Pressure Pipework, Recurrent Neural Network, LSTM, GRU, Condition Based Maintenance, Creep Degradation Estimation, Power Plant, Pipe Elevation Survey.

1. Introduction

The high pressure-steam pipework in a coal-fired power plant experiences fluctuating temperatures and pressure conditions during operation. Replacement of the system components is done on a preventative maintenance basis. The condition of the components are monitored during outages throughout their lives and replaced once significant creep damage or cracks are observed.

The steam piping systems in Figure 1 are supported from spring hangers, guides and rigid supports that are installed at strategically placed locations. The piping support system is designed to minimise the pipe stresses during operating conditions, due to the fact that the creep life and pipe stress have a direct negative correlation. Elevation surveys are used to see if the pipework is moving as per its original design, when the piping conditions are changed from atmospheric conditions to operating conditions.

Pipe supports that are not working within their intended design envelope usually cause increased stress on the piping system and directly influences the useful life of the pipework systems that are operating in the finite life temperature range.

The aim with this work is to see if any patterns can be recognized from historical data and to see if premature failures could be linked to the operating conditions (pressures and temperatures) as well as supports that are operating outside of its design intent. It has been shown that artificial neural networks (ANNs) can be used to make remaining useful life predictions for various applications where sufficient historical data is available. ANNs have been developed and proven to be successful in many diagnostic and prognostic applications as shown by Venkatesh and Rack (1999) and Tian (2012).

In the previous research conducted the data inputs are either temporal or static. In this research a hybrid network is implemented that can take both temporal and static data simultaneously as inputs. Due to the number of input features involved and the sensitivity of the remaining useful life to the input variables, the problem is regarded to be in a high-dimensional feature space. Using traditional remaining useful life methods to make very accurate predictions seems to be impossible. Using

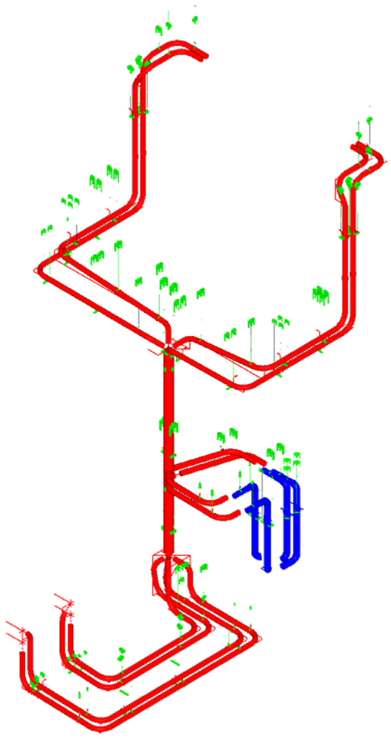

Figure 1: Main Steam And Cold Reheat Piping Systems

a hybrid recurrent neural network offers a possible way of generating predictive models for complex problems such as these.

1.1. Problem Statement

Replacing components at the end of its useful life is a very tedious process requiring a lot of time and resources. Predicting when components will need to be replaced is notoriously difficult in these systems. Currently there exists no reliable automated method that can be employed to predict the end of life of a component operating in such systems. Predictions are traditionally made by experienced system engineers who base their inspection and replacement plans on previous inspection results, basic creep life model estimations and past experience with these systems.

Due to the high number of variables present in piping systems, it is very difficult to make an accurate prediction as to when components will need to be inspected. These variables include: material strength scatter, welding techniques, post weld heat treatment, the geometry of the piping systems, fatigue, creep, corrosion, erosion, vibration, support effort, stress and temperature during operation, thermal expansion stress, etc.

A method is needed to parse both the temporal data (temperature and pressure sequences) and static data (collected during outages) and give an objective prediction of the expected damage of a piping component before the unit is shut down for component inspection. The prediction needs to be data driven and based on historical experience.

Improved remnant life estimates of high-energy piping systems will enhance life-cycle management of piping systems and pipe-replacement interventions can be planned accordingly.

1.2. Creep life of a component

Damage models are based on metallographic micro-void damage. During outages the metal surfaces of the high energy piping systems are replicated using a metallographic replica. This is inspected under a microscope with a magnification of 400 times. Voids are counted and quantified as voids per square millimetre. Creep damage models are usually specific to the type of weldment, base material and geometry.

Where the creep strain rate is less than $10^{-6}/h$ the assumption is made that the strain rate is directly proportional to the void formation rate (Van Zyl et al., 2005).

$$\frac{d\epsilon}{dt} = k\frac{dN}{dt} \tag{1}$$

The fraction of life used can be written as:

$$\frac{t}{t_r} = \left[1 - \left(1 - \frac{N}{N_f}\right)^{\lambda}\right] \tag{2}$$

where N is the current number of voids per square millimetre and N_f the number of voids per square millimetre at end of the component useful life. In practice this could evolve to a few thousand voids per square millimetre before micro crack formation and crack development occur. The material specific ductility parameter is λ and the life fraction consumed of a component is $\frac{t}{t_f}$.

It is important to note that there is a direct relation between the number of voids measured (N) and the consumed life fraction of a component ($\frac{t}{t_f}$). It is therefore possible and more convenient to train a machine learning model on the number of voids measured (N), and use the model to predict the future number of voids in a component. If the number of voids are known then the remaining life of the component can be calculated.

Equation 2 is a useful way to monitor the life fraction used of a component throughout its life, by taking metallographic replicas during outages. The life fraction used ($\frac{t}{t_r}$) can be employed as the regression or classification label ($\vec{y}$) to train a machine learning model. In this the number of creep voids is used as the label ($\vec{y}$). This makes the model less complex and besides normalizing the data, no post processing is required to determine the expected number of voids measured during an outages.

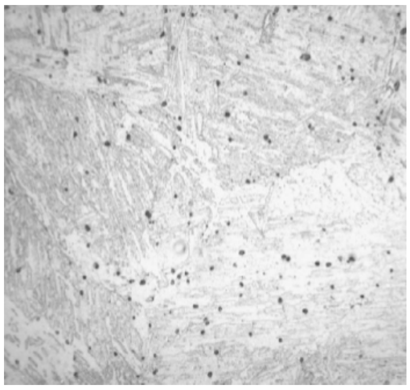

Figure 2: Creep void damage in X20CrMoV121 Main steam bend in 2003

Figure 2 shows an image of the material surface under a microscope using 400 times magnification. During outages the metallographic replicas, that could be likened to a fingerprint of the material structure, are sent to a laboratory where the number of voids per square millimetre are counted for each replica slide. This is recorded and used to label the condition of the components in this work.

2. Methodology

A feed forward neural network is combined with a recurrent neural network to form a hybrid neural network. The hybrid neural network has the advantage that both temporal as well as static data can be used as input to a single hybrid neural network model and optimization of both constituent networks can take place at the same time. This means that the model parameters should theoretically be optimized to account for interdependencies between the temporal and static data inputs. The hybrid network parameters are updated using the Adam-optimization algorithm that makes use of gradient descent back propagation and the loss, which is an error measure that measures the performance of the hybrid network.

The dataset used is also explained briefly in this section as well as the formatting of the dataset that is needed to feed the data into the network.

2.1. Feed Forward Neural Network

A feed forward neural network (FFNN) consists of an input layer, hidden layers and an output layer, where all connections are forward feeding and no recurrence or backward feeding is applied. The number of input, hidden and output layers can be customized to fit a specific problem. A simple FFNN that consists of the following is depicted in Figure 3:

- One input layer, with two input nodes and one biased parameter.
- One hidden layer, with three hidden nodes and one biased parameter.
- One output layer, with two output nodes.

The feed forward neural network is a powerful pattern recognition tool as explained by (Duda et al., 2001). There exist many permutations of the feed forward neural network. The hidden layers as well as the output layers have a non-zero bias nodes as an input that is connected to all nodes in the layer.

The activation functions can be any one of many functions. In this research softmax and tanh functions are used based on a trial and error.

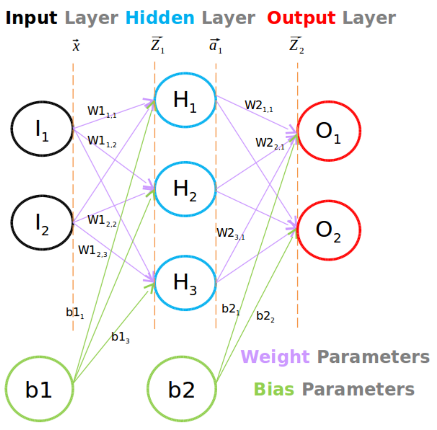

Figure 3: Feed forward neural network

Assuming the activation function that is used for forward propagation is tanh, then for a system with I input layers, N hidden layers and M output layers.

$$\vec{z}_1 = \vec{x}\vec{W}_1 + \vec{b}_1 \tag{3}$$
$$\vec{a}_1 = \tanh(\vec{z}_1) \tag{4}$$
$$\vec{z}_2 = \vec{a}_1\vec{W}_2 + \vec{b}_2 \tag{5}$$
$$\vec{y}\,' = \text{softmax}_c(\vec{z}_2) \tag{6}$$

where $\vec{W}_1 \in R^{I\times N}$, $\vec{b}_1 \in R^N$, $\vec{W}_2 \in R^{N\times M}$ and $\vec{b}_2 \in R^M$. The values of $\vec{b_n}$ are changed during training while b_1 and b_2 are constant values typically chosen as 1.

2.2. Recurrent Neural Network

An RNN is a network based on the principle of the feed forward network adapted for application on sequential data. The difference is that the output of step t is used as the input to step $t + 1$.

The cell takes an input x_t and outputs a value h_t and allows information obtained at time t to persist to time $t + 1$.

RNNs have proven to be ideal for sequential data tasks such as speech and handwriting recognition and generation (Chung et al., 2015).

One of the appealing features of RNNs is the idea that they might be able to connect previous information to the present, such as using previous data to give context to the data that is read at the current time step. This is useful in analysing temporal data such as temperature and pressure data.

2.3. Gated Recurrent Unit

Bengio, Simard and Frasconi (1994) investigated the problems with the RNN, and discovered that the importance of the current input on the RNN output quickly diminishes within a few steps.

The Gated Recurrent Unit or GRU was introduced by Cho et al. (2014). This is a simplified version of the LSTM (Hochreiter and Schmidhuber, 1997) that combines the input gate and the forget gate. The gated algorithm has gained preference from researchers, partly because the GRU trains in less time and generally can be trained using less data.

The calculation sequence used in the GRU cell to get the cell output is:

$$\vec{z_t} = sigmoid(\vec{W_z} \cdot [\vec{h_{t-1}}, \vec{x_t}]) \quad (7)$$
$$\vec{r_t} = sigmoid(\vec{W_r} \cdot [\vec{h_{t-1}}, \vec{x_t}]) \quad (8)$$
$$\tilde{h}_t = \tanh(\vec{W} \cdot [\vec{r_t} \odot \vec{h_{t-1}}, \vec{x_t}]) \quad (9)$$
$$\vec{h_t} = (1 - \vec{z_t}) \odot \vec{h_{t-1}} + \vec{z_t} \odot \tilde{h}_t \quad (10)$$

2.3.1. GRU Back propagation

Due to the increased complexity of the GRU cell over the normal FFNN cell, the calculation of the gradient descent algorithm becomes much more involved. The complete derivation of back propagation algorithms is explained in (Hochreiter and Schmidhuber, 1997). An open source application program interface that can do automatic differentiation, Tensorflow, is used here to perform the computations.

2.4. Hybrid RNN

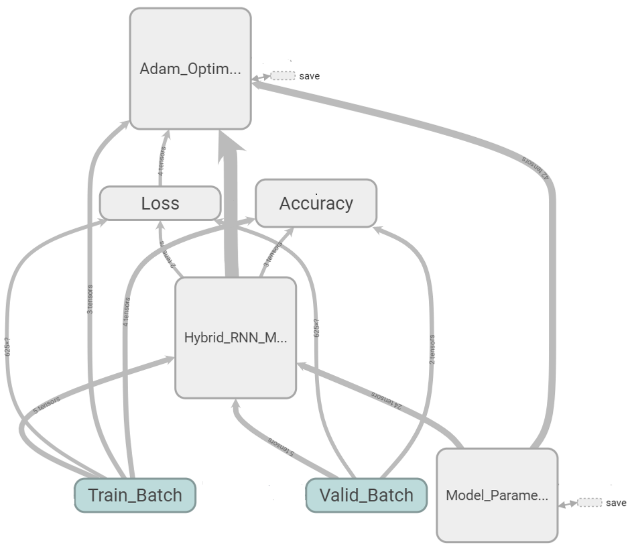

Figure 4: Tensorflow Graph

For the training and validation datasets four tensors are created [$\vec{x}$, y, seqlen, $\vec{components}$].

$\vec{x}$	The temperature and pressure input sequence. A two dimensional float tensor of shape [sequence length, 2].
y	The label, in this case the maximum number of voids experienced by a component, the tensor contains a single float value.
seqlen	The length of the sequence is saved in a tensor that contains a single value.
$\vec{components}$	The discrete information related to each of the components including elevation survey readings, last known maximum number of voids. A one dimensional float tensor of shape [number of features].

Tensorflow batches are created where the batch size = K. A batch is a collection of multiple inputs. A batch of size K contains the data for K components.

2.4.1. Training Sequence

In Figure 4 the training data batch is sent to the hybrid RNN prediction model. The model makes a prediction on each of the entries in the training batch. The loss is calculated by comparing the predicted labels to the actual labels. The loss, which is an error measure, is sent to the Adam-optimiser algorithm that calculates the gradients as per subsection 2.6. The Adam-optimizer algorithm determines the updates that need to be made to the model parameters and the model is updated accordingly. The model parameters are saved in order for it to be recalled during the testing and validation sequences.

It is important to note that the model is only trained using the training data. The validation data batch is only fed into the hybrid RNN prediction model and the accuracy of the predictions are calculated.

2.4.2. The Model Layout

The model layout is shown in Figure 5. The sequential data is sent through a GRU model to compress the information contained in the temperature and pressure sequence to a single value.

2.4.3. The Hybrid Recurrent Neural Network Forward Pass

The model hyper-parameters need to be decided to before the network weights are optimized. The hyper-parameters of the network was chosen based on more than 15 trail runs and changing the hyper-parameters manually to improve performance. Although the parameters of the hybrid RNN is optimised in this work, the hyper-parameters are not fully optimised and does leave room for improvement. The following hyper parameters were chosen for this research:

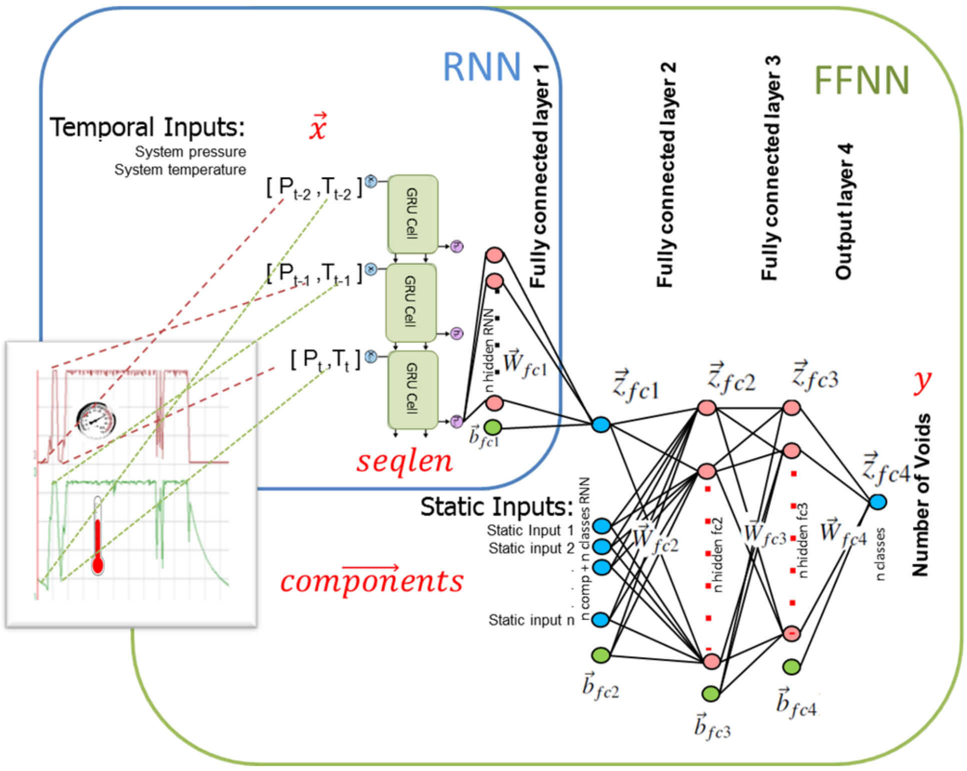

Figure 5: Hybrid RNN Model Layout

n hidden RNN	= 2	(RNN cell output size/ number of output hidden nodes)
n classes RNN	= 1	(Number of outputs of the RNN Network)
n comp	= 37	(Number of static component features)
n hidden fc2	= 256	(Number of hidden nodes in fully connected layer 2)
n hidden fc3	= 64	(Number of hidden nodes in fully connected layer 3)
n classes	= 1	(Number of output classes, this is 1 in case of regression)

An overall feed-forward pass function is created. This function takes the batch x, batch component, and batch sequence length values as an input. The built-in dynamic RNN function is used, which enables the use of varying length tensors in the same batch. The dynamic RNN uses the batch x values and the sequence length as an input, while the component vector is used together with the output of the RNN to form the input of the FFNN. The specified cell is an GRU cell and the outputs of the RNN are calculated as described in section 2.3.

$$\vec{output} = \text{dynamic\_rnn}(\vec{x}, \text{GRU cell}, \text{Seqlen}) \tag{11}$$

The RNN gives an $\vec{output}$ in the shape: [batch size, max sequence len, n hidden RNN]

The RNN output saves the output vector for each step of the RNN. However only the last output is of interest. The index of the last output of each of sequence in the batch needs to be saved and is calculated using:

$$\vec{index} = \text{range}(0, \text{batch size}) \times \max(\text{seqlen}) + (\vec{seqlen} \text{ -}1) \tag{12}$$

The index is a vector that contains the location of the final output for each of batch entry, in a flattened output vector. Only the output vectors at the index locations are saved in the output matrix, the rest are discarded. Each line in the output matrix contains the last RNN outputs of shape [n hidden RNN], of a batch entry. This means the output is in the shape, [batch size, n hidden RNN].

The final RNN output is calculated for each batch using:

$$\vec{z}_{fc1} = \tanh(\vec{output}[\vec{index}])\vec{W}_{fc1} + \vec{b}_{fc1} \tag{13}$$

where tanh is the activation function used. The RNN Output is concatenated with the static inputs (batch comp) to form a dense input to the next fully connected layer of shape [n classes RNN + n comp].

$$\vec{z}_{fc2} = \tanh(\vec{\text{dense input}})\vec{W}_{fc2} + \vec{b}_{fc2} \tag{14}$$

This is sent through more feed forward layers in the same fashion.

$$\vec{z}_{fc3} = \tanh(\vec{z}_{fc2})\vec{W}_{fc3} + \vec{b}_{fc3} \tag{15}$$

The final layer is sent through a sigmoid to ensure the result lies between 0 and 1.

$$\vec{y} = \vec{z}_{fc4} = sigmoid(\tanh(\vec{z}_{fc3})\vec{W}_{fc4} + \vec{b}_{fc4}) \tag{16}$$

2.5. Loss Function

The loss is an error measure used during optimization and is calculated per batch, hence the loss is:

$$E = \sqrt{\frac{\sum_i (\vec{y}_i - \vec{y}'_i)^2}{K}} \tag{17}$$

where y_i is the actual number of $\frac{10^3 \text{ voids}}{mm^2}$ of the i^{th} component in the data batch. y'_i is the predicted number of $\frac{10^3 \text{ voids}}{mm^2}$ for the i^{th} components in the data batch and K is the size of the data batch.

2.6. Back-Propagation Gradient Calculation

Back-propagation based algorithms are the most widely used algorithms for supervised learning with multi-layered feed forward networks. The basic idea of the back propagation learning algorithm is the repeated application of the chain rule to compute the influence of changes in the parameter values of the network on the value of the loss-function E (Mishra and Savarkar, 2012).

$$\frac{\delta E}{\delta \vec{W}_{ij}} = \frac{\delta E}{\delta \vec{a}_i}\frac{\delta \vec{a}_i}{\delta \vec{z}_i}\frac{\delta \vec{z}_i}{\delta \vec{W}_{ij}} \tag{18}$$

2.7. Optimization Algorithm

The optimisation algorithm uses the gradient calculated in subsection 2.6 to update the model parameters which should in theory reduce the loss calculated in subsection 2.5. The well known steepest descent algorithm is explained by Jacobs (1988). During preliminary research work it was found that this optimization algorithm does have some deficiencies when there exist multiple local minima. In this research the Adam-optimization algorithm , (Kingma and Ba, 2015) is used since a hybrid model is used, where the recurrent network receives data much more frequently than the feed forward neural network. This allows different learning rates for the various parameters of the hybrid network. This should theoretically address the problem where one of the networks in the system over-fits to the data before the other network has been trained properly. This has been applied successfully on speech modelling using recurrent neural networks (Chung et al., 2015).

Adam-optimization is different to the classical stochastic gradient descent in that a learning rate is maintained for each network parameter (weight). Each learning rate is separately adapted as learning unfolds.

Tensorflow's built in Adam-optimization feature was used to optimize the model.

The following hyper-parameters for the Adam-optimization algorithm that used in this research: η =0.001; β_1 = 0.9; β_2 = 0.999 and ϵ = 1e-08. the hyperparameters are the same as used in (Kingma and Ba, 2015).

2.8. Classification Accuracy

The model is set up to solve the regression problem and the accuracy of the model is an arbitrary value that is used to gauge how good the model predictions are. Four arbitrary classes are created based on the maximum void count of a component. Traditionally the components are placed in a class based on the hours of remaining life, however this is dependent on the component type and component material as well as the location of the void count. To simplify one would just want to categorize the components based on the number of voids as this is related to the remaining life of a component.

The components are considered to be completely creep exhausted once the void-count reaches 1000 voids per square millimetre.

Hence, four classes are created:

Table 1: Classification labels for accuracy calculation

Class	**Description**	**$\vec{y}$ Encoding**
Class1	Void Count 750+	[1,0,0,0]
Class2	Void Count 500 to 750	[0,1,0,0]
Class3	Void Count 250 to 500	[0,0,1,0]
Class4	Void Count 0 to 250	[0,0,0,1]

$$Accuracy = \frac{\sum_{i=1}^{K} \delta(Class(\vec{y}'_i), Class(\vec{y}_i))}{K} \tag{19}$$

The δ function is 1 if the constituents of the function variables are equal to one another. $Class(\vec{y}')$ is the predicted class, $Class(\vec{y})$ is the actual class and K is the batch size.

2.9. The Data Set

The dataset comprises of information collected from a coal-fired power station from 2011 to 2017, for the main steam piping systems of five of the operating units. The data consists of the temperatures and pressures that the piping system experience during operation, elevation survey data, metallographic inspection results, pipe stress analysis results as well as constant component properties. In cases where the temporal data was not available the assumption is made that the unit was on outage during this time and the components were not incurring any creep damage during this time.

2.9.1. Defining the Components

In this study components are defined as circumferential butt welds, fillet welds or bends. Past experience indicates that these are the areas where excessive damage are most likely to occur. Figure 6 illustrates the component numbers for leg RA11 of the main steam piping system.

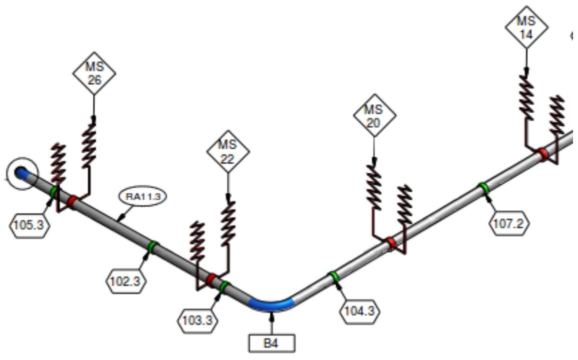

Figure 6: RA 11 Main Steam Components

A Python dictionary was created that contains all the information of a component that is assumed constant throughout its operating life.

Some information such as text, cannot be fed into a mathematical machine learning model. For instance, the component type is described by a string value "Buttweld". In order to discriminate between component types a one hot encoding vector was generated. In this case there exist three different component types thus an encoding vector could be [1, 0, 0] which indicates that the component is of the first type. Thus, only one of the values in the vector is 1, the rest are 0.

The Python dictionary captures the following information for each of the components in the piping system:

Component Name	-	The unique name of the component that corresponds to similar components on other operating units
Component Type	-	One hot encoding (buttweld, filletweld or bend)
ID	-	Pipe internal diameter
WT	-	Pipe wall thickness
Stress	-	Code stress under design conditions
EP1	-	Elevation Point 1 (the closest support)
EP1 dist	-	Horizontal distance along pipe to Elevation Point 1
EP2	-	Elevation point 2 (the second closest support)
EP2 dist	-	Horizontal distance along pipe to Elevation Point 2
EP3	-	Elevation point 3 (the third closest support)
EP3 dist	-	Horizontal distance along pipe to elevation point 3
Dist Anchor	-	Distance to an anchor point
Repl Months	-	The number of months in operation
OTN	-	This denotes if the weld was new material to old material or new material to new material at the time of installation

2.9.2. *Automated Data Reading*

An outage dictionary is manually created that contains the start and end date of the unit operation. The outage dictionary also lists the location of the associated files that contain the data that was measured during each of the listed outages. A script is used to iterate through each entry in the outage dictionary. The corresponding operational temperature and pressure data signals are read from the data frame that contains all the pressure and temperature data related to each of the outages. This is saved in an operational data pickle (Opp data.pickle).

Table 2: Opp Pickle extract

Date Time Index	Pressure [MPa]	Average Temp [° C]
12/02/06 00:00	16.667	543.66325
12/02/06 01:00	16.654	544.24925
12/02/06 02:00	16.667	544.83525
12/02/06 03:00	16.661	544.359
12/02/06 04:00	16.673	546.0075
12/02/06 05:00	16.685	547.5825
12/02/06 06:00	16.679	549.5605
12/02/06 07:00	16.673	544.79875
...	...	...

The replica data is read from the corresponding outage report. A data frame is generated that contains a list of all components inspected during this outage. The replica results of each of the components are saved in the data frame as well as the last known replica results of the components.

The properties of each of these components are read from the component database. Elevation survey results for the closest three supports of each component are read from the elevation data-frames. The component data-frame is saved in a component pickle (Component.pickle).

Table 3: Component pickle extract

Component Name	RA11.Blr.exit	RA11.101.1	RA11.106.1	RA11.107.1	RA11.108.1
Current Max Voids	50	2300	1100	2200	1700
Previous Max Voids	80	70	60	80	140
Buttweld	1	1	1	1	1
Bend	0	0	0	0	0
Fillet	0	0	0	0	0
ID	250	250	250	250	250
WT	31	31	31	31	31
SIF	69382	69382	65329	39910	51347
EP1 dist	2587	2000	-510	-4460	-6960
EP2 dist	8097	7337	4907	270	-1230
EP3 dist	10494	9734	7307	3357	1857
Dist Anchor	0	-760	-2510	-5472	-7972
Repl Hrs	368	44	44	368	368
OTN	0	1	1	0	0
Design Travel 1	-104	-104	-104	-104	-104
Hanger Load 1	14.5	14.5	14.5	14.5	14.5
Hot 1.1.1	-0.093	-0.093	-0.093	-0.093	-0.093
Hot 1.1.2	-0.092	-0.092	-0.092	-0.092	-0.092
Hot 1.2.1	-0.1	-0.1	-0.1	-0.1	-0.1
Hot 1.2.2	-0.101	-0.101	-0.101	-0.101	-0.101
Hot 1.3.1	-0.092	-0.092	-0.092	-0.092	-0.092
Hot 1.3.2	-0.095	-0.095	-0.095	-0.095	-0.095
Design Travel 2	-94	-94	-94	-94	-94
Hanger Load 2	13.4	13.4	13.4	13.4	13.4
Hot 2.1.1	-0.086	-0.086	-0.086	-0.086	-0.086
Hot 2.1.2	-0.078	-0.078	-0.078	-0.078	-0.078
Hot 2.2.1	-0.125	-0.125	-0.125	-0.125	-0.125
Hot 2.2.2	-0.104	-0.104	-0.104	-0.104	-0.104
Hot 2.3.1	-0.098	-0.098	-0.098	-0.098	-0.098
Hot 2.3.2	-0.081	-0.081	-0.081	-0.081	-0.081
Design Travel 3	-135	-135	-135	-135	-135
Hanger Load 3	60.4	60.4	60.4	60.4	60.4
Hot 3.1.1	-0.111	-0.111	-0.111	-0.111	-0.111
Hot 3.1.2	-0.109	-0.109	-0.109	-0.109	-0.109
Hot 3.2.1	-0.175	-0.175	-0.175	-0.175	-0.175
Hot 3.2.2	-0.143	-0.143	-0.143	-0.143	-0.143
Hot 3.3.1	-0.136	-0.136	-0.136	-0.136	-0.136
Hot 3.3.2	-0.127	-0.127	-0.127	-0.127	-0.127

The automated data reading programs creates two pickle files for each recorded outage, that will be used to train the model.

1. Opp-data.pickle

2. Component.pickle

The Opp-data pickle creates the temporal data for the outage. The Component pickle contains the static data of the components that have been inspected during the current outage. Static data is data that is assumed to stay constant throughout the duration of the operation from the previous outage to the current outage.

2.10. *Normalizing*

The data is normalized using min-max scaling. If a data point T lies between T_{min} and T_{max} the normalized value x will be in the range [0,1]. This makes the neural network computationally more stable, meaning that features such as stress in the 10^5 [kPa] ranges can be used in the same model as the elevation survey features in the 10^{-3} [m] ranges.

$$x_i = \frac{T_i - T_{min}}{T_{max} - T_{min}} \tag{20}$$

The following T_{min} and T_{max} values are used to normalize the training data (Feature $\in [T_{min}, T_{max}]$):

Temperature	$\in$ [0,600]
Pressure	$\in$ [0,17]
Buttweld	$\in$ [0,1]
Bend	$\in$ [0,1]
Filletweld	$\in$ [0,1]
Pipe ID	$\in$ [0,500]
Pipe Stress	$\in$ [20000,170000]
Maximum Voids	$\in$ [0,1000]
EP1	$\in$ [-19554, 5802]
EP2	$\in$ [-9832, 10917]
EP3	$\in$ [-3824,14317]
Distance to Anchor	$\in$ [-31714, 27568]
Months in operation	$\in$ [0,369]
Old to new	$\in$ [0,1]
Support design travel	$\in$ [-154,168]
Support design load	$\in$ [0,156.8]
Support movement	$\in$ [-0.154,0.168]

2.11. *Running the Model*

The model runs the complete training dataset *num epochs* times through the training sequence as explained in subsection 2.4. The model processes one batch at a time where a batch contains the specified *batch size* number of components.

After each training batch sent to the optimization sequence, the updated model is validated with a validation batch. Thus for each training epoch the batch loss and batch accuracy for each of the training, validation data sets are recorded and plotted.

3. RNN Hybrid Network Results

The Hybrid RNN network design as described in subsection 2.4 is used with the input data collected as described in subsection 2.9. The results listed in this section were run on a Windows based operating system using Python 3.5 with the Tensorflow 1.2 library. The following hardware setup was used to give the reported results:

Table 4: Hardware Setup

CPU	**GPU**	**System Memory**	**HardDrive**
Intel Core i5 @3.30GHz	NVIDIA GeForce GTX 1060 6GB, 1280 CUDA Cores	20GB @1600MHz	240GB SSD, 545 mb/s sequential read speed

Some of the model parameters were kept constant for all the runs in order to keep consistency between different runs as well as keeping the degree of complexity of the problem lower. The model was trained using the following fixed hyper-parameters, where hyper-parameters refers to parameters that are set manually and are not changed during optimization of the model:

Table 5: Constant Model Hyper Parameters

Sub-sample method	**Learning rate**	**n classes RNN**	**Loss**	**Component features**
max	0.001	1	RMS	37

3.1. *Best Run*

Using the built in Adam-optimizer function in Tensorflow alleviates the problem where the optimization algorithm gets stuck in a local minimum point as the model is able to fit the training data very well. It was found that using this approach the model is able to obtain a very high training accuracy of 92%. Meaning it is very good at classifying a set of input data that it has seen before. However when it is provided a new set of input data, it only achieves a classification accuracy of 55%. This could mean that the training dataset is not large enough, or there is insufficient information contained in the input data or that there is a very low correlation between the measured inputs and the damage of a component.

The runs performed have the same hyper parameters as noted in Table 6.

Table 6: Best Run Hyper Parameters

Run Number	**Sample Rate**	**Number Epochs**	**Batch Size**	**RNN Hidden Nodes**	**Layer 2 Hidden Nodes**	**Layer 3 Hidden Nodes**	**RNN Cell**	**Algorithm**
1	1H	2145	256	2	256	64	GRU	Adam

The results are listed in Table 7.

Table 7: Best Run Results

Run number	Loss train	Loss valid	Accuracy train	Accuracy valid	Running time [h]
1	0.1921	0.3948	92%	55%	26.25

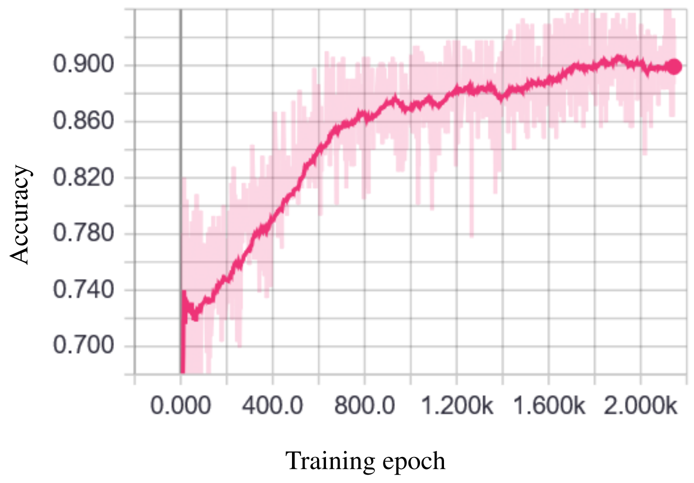

Figure 7: Training accuracy

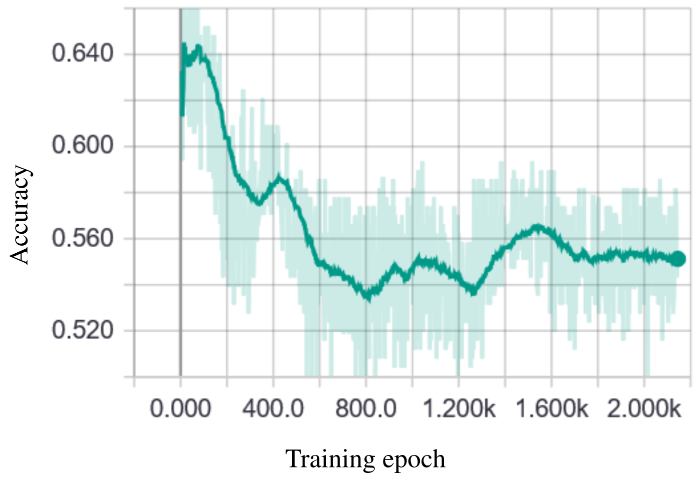

Figure 8: Validation accuracy

Figure 7 and Figure 8 illustrate the training curves for the training and validation sets. It is clear that the training accuracy trends upward while the validation accuracy does not follow any trend. The loss curves Figure 9 and Figure 10 show a clear downward trend in the training dataset but no real trend in the validation set. The lowest validation loss is around epoch 260, at this stage the model predicts almost all voids to be zero, this may seem to be a good solution as most of the components inspected has got low void counts. This is however not a good solution as a model that predicts zero output is not useful. Letting the model overfit to the training data yields a better result in finding high damaged components. Keep in mind that the accuracy in this case is an arbitrary measure and a decreasing accuracy does not necessarily mean that the model becomes worse at making predictions.

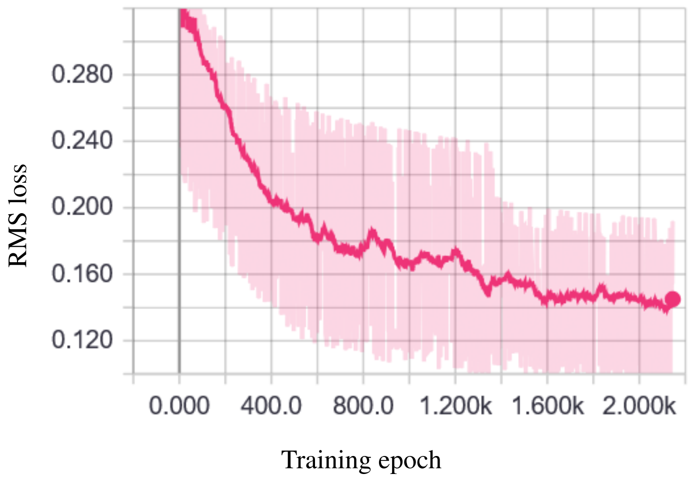

Figure 9: Training loss

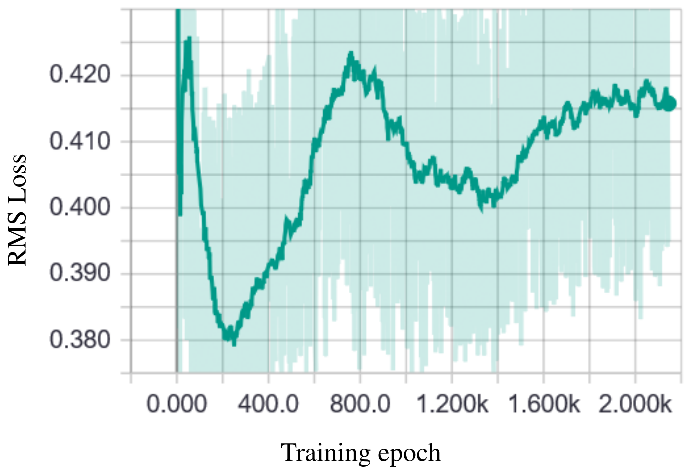

Figure 10: Validation loss

Figure 11 and Figure 12 show the predicted number of voids compared to the actual measured number of voids. The black dashed line indicates an ideal line where the predicted number of voids would be exactly the same as the measured number of voids. From this figure it is clear that given the complexity of the problem the model fits the training data very well. However, when the model is given a new validation dataset the fit is not as good.

The model over fits to the training data as it is clear that the validation loss does not monotonically decrease. However, in the experience of the author the model becomes better at identifying high damage components if it is allowed to over-fit to the training data. If this is not done the model predictions seems to be bias to zero. Using this approach will not help in predicting components with high damage.

There is a significant scatter in the predicted vs. measured number of voids. This could be partially due to a weakness of the proposed model as it has only been trained on a finite number of examples. However, given the complexity of the problem at hand a significant scatter is to be expected. Doing a sensitivity analysis using PD 6525 (BS, 1990) it can be seen with a temperature variance of $\pm 5°C$, pressure variance of $\pm 10\%$ and a material property variance of 15%. It can bee seen that the estimated theoretical ISO Mean Life can be anything between 667 thousand hours and 4.4 million hours. This is a significant scatter which excludes the influence of varying stress due to stress concentrations and support efforts in the components.

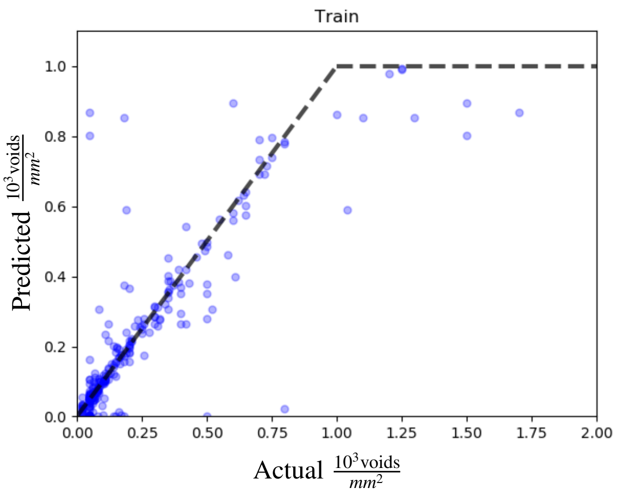

Figure 11: Training data fit

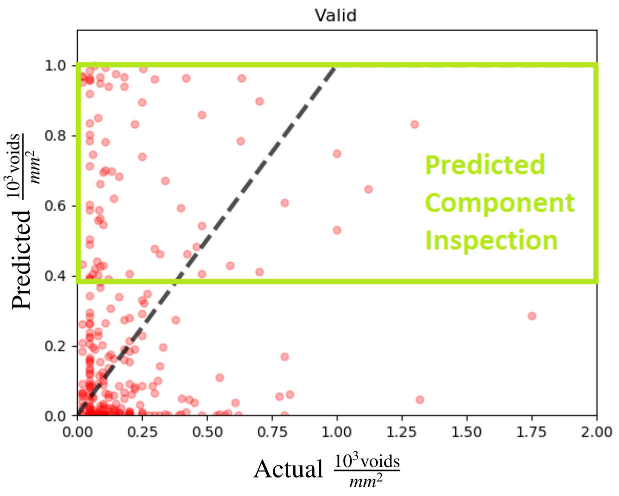

Figure 12: Validation data fit

For the given problem the components with higher damage classes are of more interest than those with little to no creep damage. One could hypothetically decide that only components with very high void counts above 400 voids per square millimetre are of interest. In this case, based inspections by system engineers over a 6 year period, 14% of components will fall within the criteria. Inspecting component based on the predictions of the hybrid RNN a hit-rate of 25% will be achieved, which is a significant improvement of the current inspection plan performance.

3.2. Discussion

3.2.1. Network Layout

It was found that a RNN with output size of 32 and a fully connected feed forwards network with a (38-256-64-1) configuration is stable enough to run on a 6GB GPU without causing out of memory errors. When increasing the network size, additional steps should be taken to ensure the GPU memory is not overfilled. It was also found that when the model complexity is increased there is no noticeable increase in prediction accuracy.

3.2.2. Optimization Algorithm

In general the both the gradient descent and adaptive gradient descent algorithms struggle to fit the training data. The Adam-optimizer is able to fit to the training data with a very low loss. This is a good sign that the model is able to pick up patters within the data. The validation error is higher than that of the training error. This is mostly because the training dataset is not large enough or the underlying patterns in the training data are very weak.

4. Conclusion

Historically temporal/sequential data sequences such as temperature-time data sequences have proven to be difficult to analyse and recognize patterns. The GRU cell based recurrent neural network (RNN) has however proved to be able to perform pattern recognition successfully on these sequences. The flexibility that this hybrid RNN model offers makes it appealing for application to real world data.

In this work it is shown that the hybrid RNN is able to recognize patterns within high energy steam piping data, by using the temperature and pressure time histories as well as previous damage and elevation survey data, as input. The model is able to fit training data that represents creep damage in terms of number of voids in a component with relatively high accuracy. Despite the good accuracies achieved on the training dataset the model however still struggles with the unseen data in the validation data set.

The lower level of success with the unseen data must however be seen in perspective. A comparison to results obtained by experienced system engineers over a period of six years, indicate that the hybrid RNN system is still capable of identifying creep damaged components better than the engineers. The hybrid RNN based system can automatically parse large and very complex data sets that consist of temporal and static data. Conventional data driven models are only able to look at either temporal data or static data. Compiling inspection scopes manually is a very tedious task for engineers. Using such a system may therefore be expected to very significantly reduce the manpower and time required by creating a detailed inspection list before an outage. It may also be expected that the model should get better at predictions as the size of the training dataset grows.

A caveat based on the application of the model is that due to the nature of the number of iterations and amount of linear algebra equations that are being performed it is very difficult to trace and ensure that there is no errors in the computations or input data. Another caveat is that the model will not be able to predict all of the components which are high risk, hence a full inspection shall still be required to ensure all high risk components are found.

This tool should also be used validate that all inspection plans include all the possible high risk class 1 and class 2 components.

The approach proposed in this research is expected to do well with similar problems where sufficient historical data is available and where creating a physics based model is too complex. The hybrid RNN is scalable and configurable to work with most types of data driven problems. The model is fully scalable meaning that if the training dataset size increases there will be no difference in the model except that more training iterations might be needed to optimise the model. If the model is to be adapted to track more input features, the number of input nodes can simply be adjusted.

Bengio, Y., Simard, P. and Frasconi, P. (1994), 'Learning long-term dependencies with gradient descent is difficult', *IEEE Transactions on Neural Networks* **5**(2), 157–166.

BS (1990), PD 6525 : Elevated temperature properties for steels for pressure purposes; Part 1 - stress rupture properties, Technical report, British Standards Institution.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014), 'Learning phrase representations using RNN encoder-decoder for statistical machine translation', *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* pp. 1724–1734.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. and Bengio, Y. (2015), 'A recurrent latent variable model for sequential data', *Advances in Neural Information Processing Systems 28 (NIPS 2015)* p. 8.

Duda, R. O., Hart, P. E. and Stork, D. G. (2001), *Pattern classification*, 2 edn, John Willey and Sons, New York.

Hochreiter, S. and Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Jacobs, R. A. (1988), 'Increased rates of convergence through learning rate adaptation', *Neural Networks* **1**(4), 295–307.

Kingma, D. P. and Ba, J. L. (2015), 'Adam: a method for stocastic optimization', *ICLR Conference* pp. 1–15.

Mishra, S. and Savarkar, S. (2012), 'Image compression using neural network', *Proceedings of International Conference and Workshop on Emerging Trends in Technology (ICWET)* **3**(2), 18–21.

Tian, Z. (2012), 'An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring', *Journal of Intelligent Manufacturing* **23**(2), 227–237.

Van Zyl, F. H., Von dem Bongart, G., Bezuidenhout, M. E. J., Doubell, P., Havinga, F. C., Pegler, D. A. H., Newby, M. and Smit, W. (2005), 'Life assessment and creep damage monitoring of high temperature pressure components in South Africa's power plant', *ECCC Creep Conference* (September), 934–945.

Venkatesh, V. and Rack, H. (1999), 'A neural network approach to elevated temperature creepfatigue life prediction', *International Journal of Fatigue* **21**(3), 225–234.