Project 9: Structural Health Monitoring

This manuscript (permalink) was automatically generated from dbudzik/project9_SHM@3d4470e on December 6, 2020.

Authors

- **David Budzik**
  · dbudzik
- **Claudia Mederos**
  · cr25
- **Bolaji Lawal**
  · bolajilawal
- **Abdullah Assaf**
  · aboo12

1. Abstract

Civil infrastructure all around is subjected to the challenges posed by aging, deterioration, and extreme events. In recent years, structural health monitoring has grown in importance as failure and collapse of infrastructure result in harmful effects on a nation's economy and development. Recent advances as seen the use of sensors to obtain the dynamic characteristics of structures. This has led to the potential of collecting dynamic data at more frequent intervals thus leading to continuous monitoring. In this report, we use exploratory data analysis techniques to see if such dynamic data collected from structures can be used to infer the condition of a structure. We have also tried different machine learning algorithms to predict the condition of a structure based on given acceleration signals. For this study, the ASCE benchmark problem experimental phase II structure was used.

2. Introduction

2.1 Motivation behind the research experiment

Identification of damage from the analysis of vibration signals has received significant attention in the civil, mechanical, and aerospace fields. Structural health monitoring allows the engineer to use sensing of the structural responses in conjunction with appropriate analysis and modeling techniques, to monitor the condition of a structure. The problem most commonly considered is that where data is recorded at two different times and it is of interest to determine if the structure suffered damage in the time interval between the two observations. The behavior of the system during the observation periods is typically assumed linear and the damage is identified as changes in system parameters. A solution can be obtained in principle by using the measured data to optimize a model of the structure in the two states and inspecting the differences. The goal of the experiment was to develop an automated structural health monitoring system capable of providing early warnings against structural damage. In order to achieve this, damage was simulated by removing bracing within the structure in nine different ways as shown in the following sections.

2.2 Literature Review

Due to the importance of this research, there have been multiple papers regarding how to optimize the recollection of the data and quantity needed for training the machine learning algorithm. Examination of the literature reveals, however, that the assumptions used to establish the various approaches vary widely and it is unclear what is the true capability of the current state of the art in damage detection of civil engineering structures.

1-D CNNs for structural damage detection: Verification on a structural health monitoring benchmark data

Nowadays, a large number of measurement scenarios are needed to generate training data before placing the sensors in large civil structures. The paper "1-D CNNs for structural damage detection: Verification on a structural health monitoring benchmark data" provides an enhanced CNN-based algorithm which only requires two training sessions, ensuring accuracy and speed. The damage was estimated successfully by applying a CNN-based approach only requiring two measurement datasets. The overall structural health monitoring procedure for this paper took place on a very short amount of time, around 5000x faster than the conventional procedure. The authors indicated that this approach can be utilized on any structures, regardless of the size of the structure. Yet, the data processed was based exclusively on a monitored structure, and one may wonder how the fully

damaged scenario data could be obtained on a large civil structure that needs to be monitored. Also, other difficulties become more apparent if we take into consideration how large civil structures do not behave precisely as the replica in which accelerometers were placed. It is almost impossible to repeat the measurement procedure that took place for the Phase II benchmark study on any other structure. And finally, although anomaly detection is obtained using CNNs algorithms, the cause of such anomaly or the location is completely a mystery. It is important to know the state of a structure for public safety, but it is also very important to locate the damage in order to take action.

Sequential Multiple Structural Damage Detection and Localization: A Distributed Approach

The article focuses on a distributed approach to structural damage detection and localization that looks to address three main drawbacks in the structural damage detection process. The three drawbacks are damage being reported with short delay, damage locations have to identified simultaneously, and computational complexity is untraceable in large-scale wiles sensor networks. To address these problems the article attempts to introduce a new damage identification approach that focuses on time-series of damage sensitive features extracted from multiple sensors' measurements and the optimal change point detection theory.

Structural Health Monitoring of Cantilever Beam, A Case Study – Using Bayesian Neural Network and Deep Learning

This case study took a look at the use of machine learning models to predict damage to a cantilever beam. The beam in this study was modeled in a finite element analysis software and was subjected to dynamic loading in the software. The raw frequency response data from this analysis was the input for the machine learning algorithms. The study compared the accuracy of three algorithms: Bayesian Neural Network, Convolutional Neural Network, and Long Short Term Memory, to see which provides the best approximation for structural damage. The approach they took created the models based on the raw data, which is an advantage over traditional methods where data must be cleaned and prepped. This makes it more suitable for real-time monitoring, which is an important possible implementation.

2.3 Structure & experiment descriptions:

Benchmark Structure

The benchmark structure is a 2-bay by 2-bay, 4 story steel frame structure at the University of British Columbia.

Figure 2.1: Benchmark Structure

Cases with known and unknown input and damage scenarios including symmetrical and unsymmetrical loss of stiffness in the bracing system were considered. The experiment in question is regarding how damage can be simulated by removing bracing or loosening bolts within a four-story steel frame structure. Complete details of the damage cases, input excitation and other pertinent aspects of the study of phase I can be found below. To obtain the data, accelerometers were placed throughout the structure to provide measurements of the structural responses. In particular, three sensors per floor. One located at the center, one at the west side and one at the east side, as the MATLAB files indicate. Then, different cases took place in which members were loosen or removed to analyze the output and correlate the difference in acceleration values with the difference in setup.

The different cases

- Case 1 - Fully braced configuration.
- Case 2 - All east side braces removed.
- Case 3 - Removed braces on all floors in one bay on southeast corner.
- Case 4 - Removed braces on 1st and 4th floors in one bay on southeast corner.
- Case 5 - Removed braces on 1st floor in one bay on southeast corner.
- Case 6 - Removed braces on all floors on east face, and 2nd floor braces on north faces.
- Case 7 - All braces removed on all faces.
- Case 8 - Configuration 7 + loosened bolts on all floors at both ends of beam on east face, north side.
- Case 9 - Configuration 7 + loosened bolts on floors 1 and 2 at both ends of beam on east face, north side.

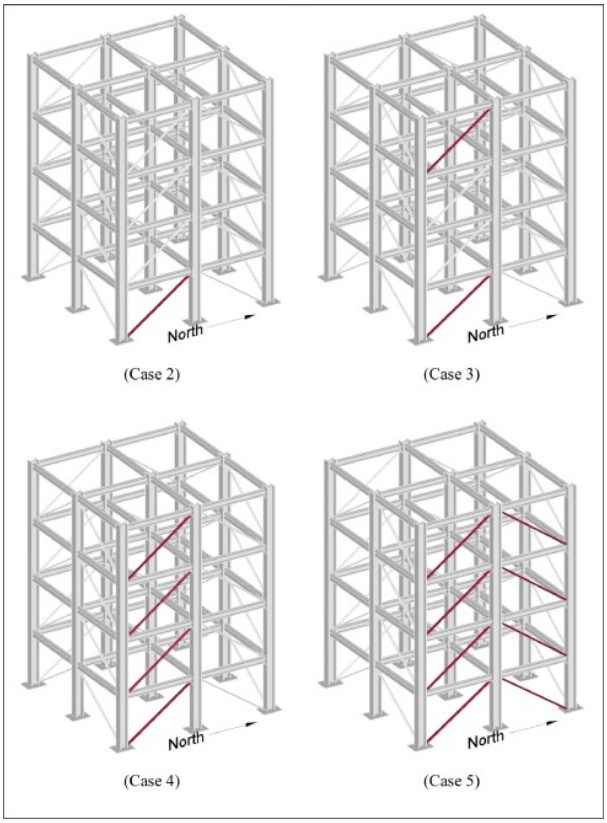

Figure 2.2: Cases 2-5

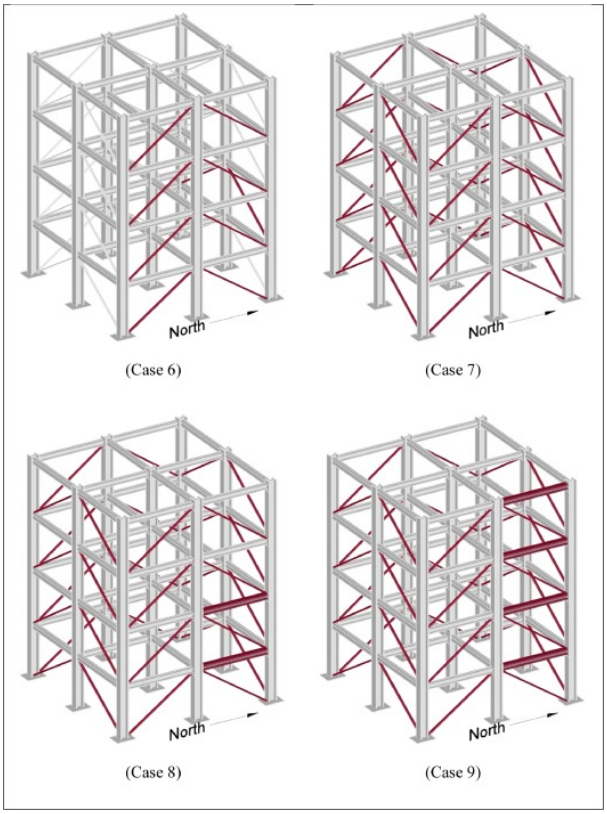

Figure 2.3: Cases 6-8

Force Input

Ambient vibration was inputted into the structure by two types of forced excitations. The forced excitation cases consider both impact hammer tests, and broadband excitations provided by an

electrodynamic shaker. The choice of these two methods is to simulate the structure's response during an earthquake.

2.4 Motivation behind the project on structural damage detection

Numerous structural health monitoring algorithms have been developed and been implemented on experimental and full-scale structure. Because the techniques are applied to different structures under various conditions, the relative merits of each algorithm are not obvious. Thus, the community would benefit from a comparison of several algorithms when applied to the same problems.

3. Exploratory Data Analysis

The goal of this EDA is to identify characteristics between damage and undamaged conditions in order to perform data preparation and develop a model in the future steps.

3.1 Preparing the data

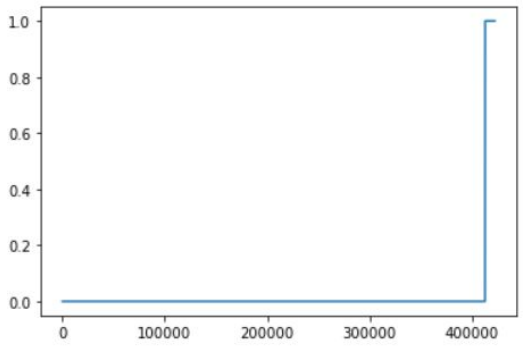

Figure 3.1: Checking if the data is sorted logically as undamaged becoming damaged.

As the dataset shows, we have damage within our data. Therefore, Case 1 is not represented in train data.

	DA04	DA05	DA06	DA07	DA08	DA09	DA10	E
0	0.00038590	0.00266320	-0.00099870	0.00217520	-0.00070116	0.000743260	0.001037760	-
1	0.00053300	0.00187336	-0.00125688	0.00254740	0.00086228	0.002021600	0.000112764	-
2	-0.00048172	-0.00416560	-0.00042344	0.00225200	-0.00179370	0.001765920	0.001906620	(
3	-0.00186968	-0.00340100	-0.00029984	0.00076952	-0.00456220	0.002955600	0.001176620	(
4	-0.00215380	-0.00296580	-0.00049546	-0.00027664	-0.00324420	0.002446200	-0.000391080	(
...	...	...	...	...	...	...	...	.
422459	-0.00162616	-0.00024338	0.00035416	0.00326620	-0.00095048	0.001888000	0.001341120	-
422460	-0.00050766	-0.00250700	-0.00112414	0.00511620	-0.00063280	0.000403600	0.003886600	-
422461	-0.00139544	-0.00384240	-0.00110612	0.00442520	-0.00130022	-0.000135348	0.003711400	-
422462	-0.00169482	0.00111986	-0.00077134	0.00325280	-0.00248400	0.001090560	0.001893800	(
422463	-0.00189654	-0.00267840	0.00079514	0.00278060	-0.00368760	0.002684200	0.001179360	-

Figure 3.2: Analyzing the training data provided.

The reason that there are accelerations values at the first row is because researchers started measuring the structure's response once it reached steady state. More details regarding the dynamic behavior of this structure will be discussed in the following sections.

Separating train data into damage and undamage dataset

```
In [8]: train_undamaged.isna().sum()

Out[8]:
1st_story_01    151971
1st_story_02    151971
1st_story_03    151971
2nd_story_01    151971
2nd_story_02    151971
2nd_story_03    151971
3rd_story_01    151971
3rd_story_02    151971
3rd_story_03    151971
4th_story_01    151971
4th_story_02    151971
4th_story_03    151971
Condition            0
dtype: int64
```

Figure 3.3: Missing data in the undamaged section.

As shown above, the given data set contained a large quantity of NA data values which are all located in the undamaged portion.

Between populating or removing the missing data, the missing results will be dropped. The reason for this decision is because populating the missing values with the mean of the training data will mostly develop a new dataset that does not have all the original parameters.

Aesthetic modification for easier comprehension of the training data The provided training dataframe had columns names such as DA04, which corresponds to the sensor located in the first floor west side. Instead of keeping these columns names, the datasets were named as shown below.

```
print(train_undamaged.columns)
print(train_damaged.columns)

Index(['Time', '1st_story_01', '1st_story_02', '1st_story_03', '2nd_story_01',
       '2nd_story_02', '2nd_story_03', '3rd_story_01', '3rd_story_02',
       '3rd_story_03', '4th_story_01', '4th_story_02', '4th_story_03'],
      dtype='object')
Index(['Time', '1st_story_01', '1st_story_02', '1st_story_03', '2nd_story_01',
       '2nd_story_02', '2nd_story_03', '3rd_story_01', '3rd_story_02',
       '3rd_story_03', '4th_story_01', '4th_story_02', '4th_story_03'],
      dtype='object')
```

Figure 3.4: Column Names.

Units

Based on *Experimental Phase II of the Structural Health Monitoring Benchmark Problem* , accelerometers were placed throughout the structure to provide measurements of the structural response. For this Exploratory Data Analysis, three sensors from each floor of the 4-story structure were taken into consideration. Specifically,

Sensor 01 = Sensor located at the west side of the structure.

Sensor 02 = Sensor located at the center of the structure.

Sensor 03 = Sensor located at the east side of the structure.

$Units = \frac{m}{s^2}$

- In the original data from the researchers, time is described as, $Time(seconds) = \frac{1{:}Length_{DA04}}{fs_{days}}$

where fs_days = 200 (Hz).

Therefore, the final step on the tidying portion of the exploratory data analysis was to transform the "Time" column into seconds by divided by 200.

3.2 Statistical Properties

We can learn certain details of the response of the structure by observing the data points that have a very drastic change in amplitude. In other words, there are common points in time among all sensors where the acceleration measured does a 180 degree change. This phenomenon occurs as the dynamics response of structure is harmonic and it develops nodes. A simplification of this idea is to understand how the sensors in the 4th story will move back and forward while the nodes underneath are ahead or behind that displacement.

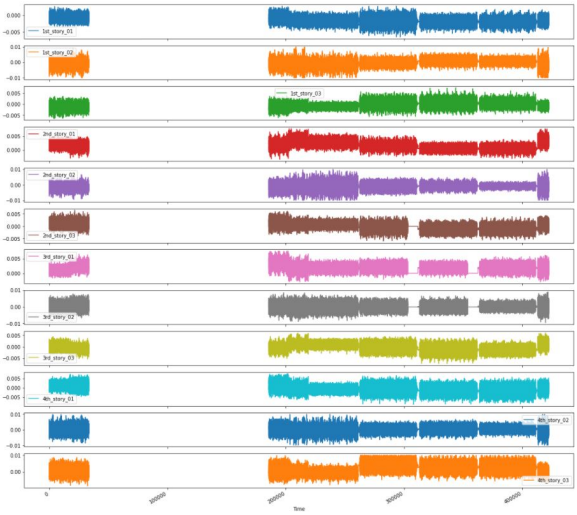

Figure 3.5: Visualization of the overall data.

The mean for all the sensors is very close to 0, which may indicate normalized normal distribution. Also, the standard deviation is not equal to 1 for any of the sensors, but a close value to 0 too. These characteristics are present for normal distributions of narrow dispersion.

In the case of undamaged dataset, the standard deviation values of the fourth floor are the largest among all floors. This indicates a flexibility in the structure as the dynamic response took place. Another very interesting fact that we can learn from the previous tables is how the maximum value of 0.01 takes place at two sensors in the 4 story, and the sensor located at the center for the 1st and 2nd floor. The absence of this value at the 3rd floor may indicate an anomaly. This value stays constant after the structure is considered damaged for the sensors located at the center of the second and fourth floor.

Checking head, tail of data

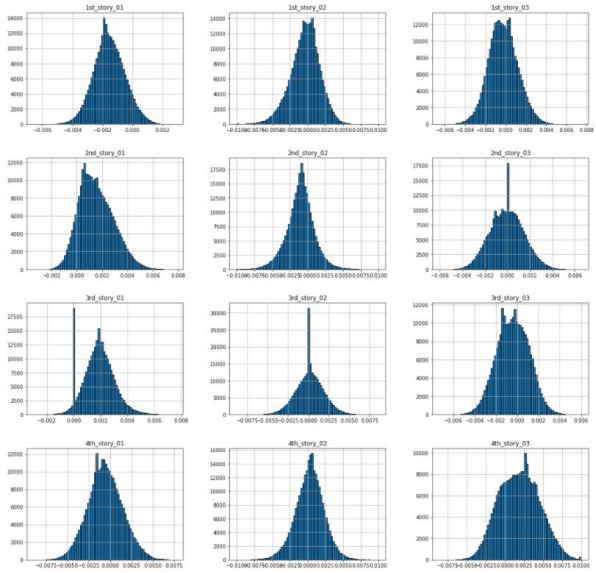

Figure 3.6: Histogram of undamaged acceleration values at each sensor

In the case of the undamaged dataset, all the graphs show normal distribution. However, they are not centered with an exact mean of value 0. Instead,

- Sensors located at the west side of the structure are skewed to the right in the first and fourth floor while the second and third floor are skewed to the left.
- Sensors located at the center of the structure behave symmetrically. The first and third floor have bell-shaped distribution with a mean of 0. The second and fourth floor are lightly skewed in opposite directions.
- Sensors located at the east side of the structure are all skewed except the one located at the second floor. The sensor at the 4th floor captured the most out of plane behavior as the '4th_story_03' sensor is significantly skewed to the left.

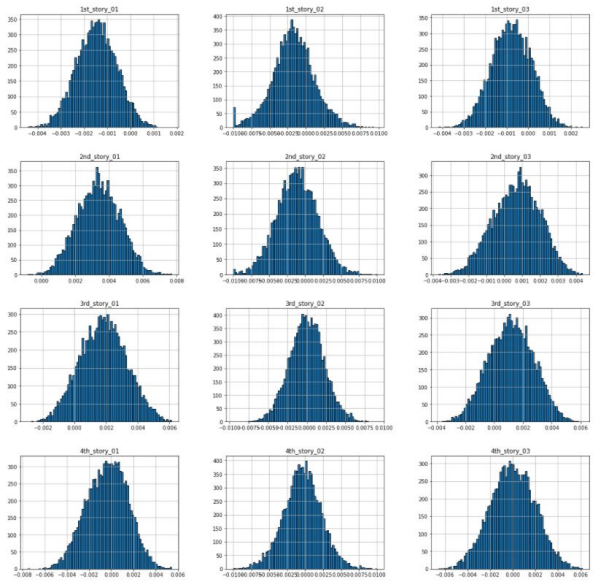

Figure 3.7: Histogram of damaged acceleration values at each sensor.

For the case of damage condition, the normal distribution is not as smooth as shown for the undamaged condition. This behavior matches with the physical phenomenon that took place as the acceleration of the sensors will tend to be more extreme if the structure is damaged.

- The first floor endured the most extreme values as the base is not static anymore during excitation. The three sensors are skewed to the right.
- The second and third floor has similar behavior since the center sensor is still normally distributed with mean very close to 0 and sensors on the west and east side are skewed to the left.
- The fourth floor now shows the most centered behavior. However, it is important to recall the statistical characteristics such as standard deviation. Now the 4th-floor values are significantly wider.

3.3 Exploring the dataframe that contains the undamaged condition

Previously, we explored some characteristics of the undamaged dataset. Then, the dataset has been arranged and tidied to finally observe how there was a large among of non-available data points, which can be observed in the last row (index number > time_sec)

Is the change in acceleration always the same? For the following inspection, recall that data acquisition was started several seconds after the excitation was turned on to ensure that the system had reached a steady state condition during the shaker testing.

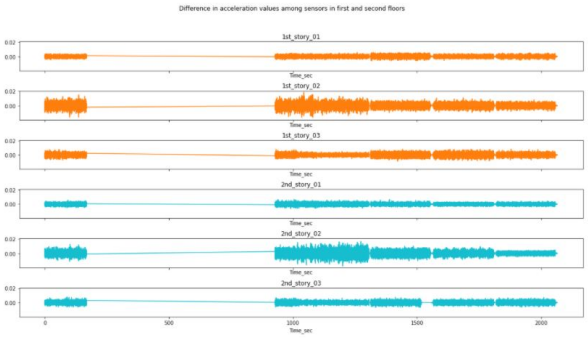

Figure 3.8: Changes in acceleration during the undamaged condition for the first and second floor.

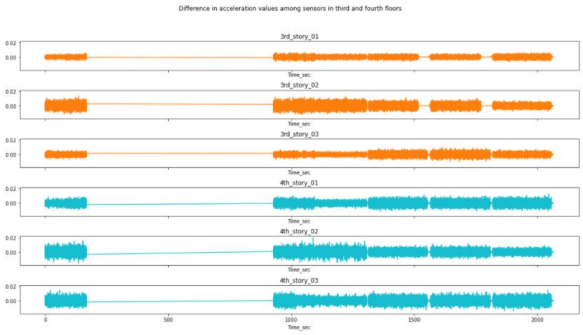

Figure 3.9: Changes in acceleration during the undamaged condition for the third and fourth.

Interestingly, the analysis has shown how the location of the sensor affects directly to the change in acceleration of the sensors. The y-axis has been kept constant throughout all the plots to ease comparison. Therefore, we can observe how though the distribution among sensors in different floor is different, the difference in acceleration values is very correlated to location.

Also note how the missing data produced zero values in the left portion of the data. Those values are not representing a constant acceleration.

Correlation values

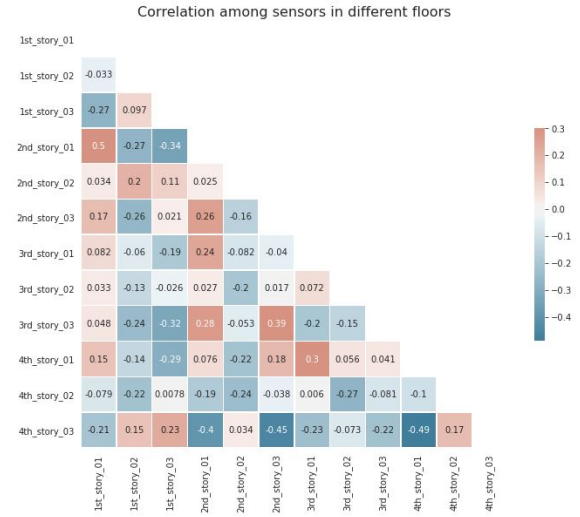

Figure 3.10: Correlation values among all sensors in the undamaged condition.

The table above is with the purpose of locating the directly correlated and inversely correlated sensors. Just as the difference in acceleration graphs showed, there is significant correlation between those sensors that are located in the same side of the structure.

However, there is an inverse correlation in those sensors located at the fourth floor. The reason behind this behavior is that we are analyzing an elastic structure that is being excited by an harmonic input from the ground. Therefore, the top floor is swinging, which creates a driving behavior in one of the corners at a time.

3.4 Exploring the dataframe that contains the undamaged condition

Is the change in acceleration always the same?

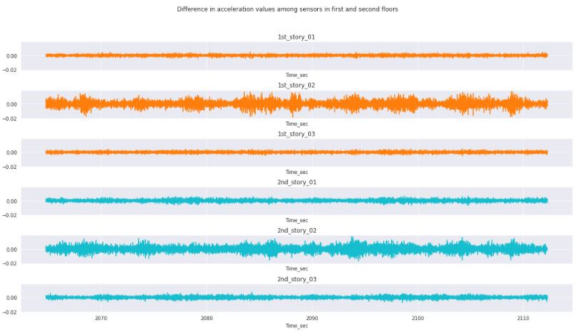

Figure 3.11: Change in acceleration between sensors in the damaged condition for the 1st and 2nd floor.

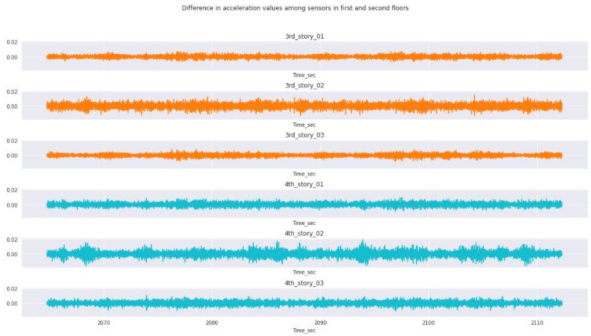

Figure 3.12: Change in acceleration between sensors in the damaged condition for the 3rd and 4th floor.

There are certain conditions that we can observe by comparing the undamaged and damaged conditions. First of all, there is still a correlation in the change in acceleration with the location of the sensors. Also, the delta value has been significantly decreased over the length of the response. The largest change in acceleration is located at the center sensors for this particular condition, which might mean that the structure is not displacing as much once it reaches a damaged condition.

Correlation values

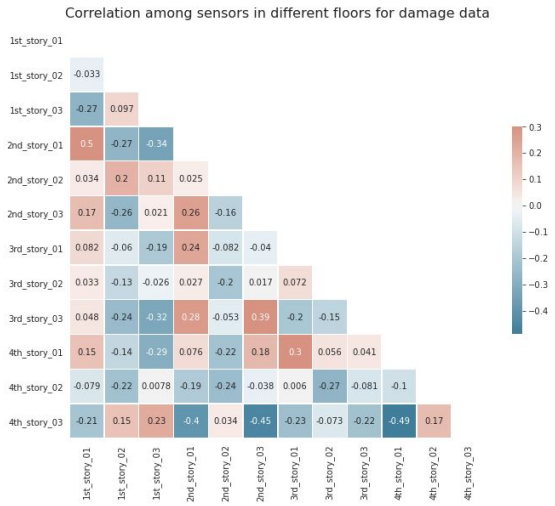

Figure 3.13: Correlation value among all sensors for the damaged condition.

In this scenario, the correlation has changed greatly. Now, the 4th-story sensors display similarities with other sensors placed in the same side of the structure. However, the 3rd-story sensors are those that are inversely correlated.

3.5 Conclusion

The training data obtained has been statistically explored, cleaned, and analyzed for the purpose of identifying parameters for modeling later on.

- Missing data has been removed.
- Datatype and Data Info has been discussed and visually listed.
- The distribution for the undamaged and damaged conditions proved to be normally distributed with skewness at different direction based on the sensors. This analysis portion showed how the skewness was correlated with the behavior that was taking place during excitation as the dynamic response of a steel frame structure is not rigid. Also, once the damaged condition was achieved, the distribution showed a larger standard deviation.
- The individual analysis of each condition proved to be succesful in correlating the behavior of each sensor among stories.
  - The change in acceleration depended directly with the location on the sensor within the steel frame structure.
  - There was a high correlation between sensors located at different floors that were place on the same side.
- Due to the high symmetry among all the results and comparisons, it is probable that the training data belonged to Case 6, Case 7 or Case 8. However, the final information indicated how the 3rd

story behaved differently, which potentially indicated that the training data has a higher probably of being Case 8.

4. Machine Learning Methods

Overview

Following a comprehensive exploratory data analysis as outlined in the previous section, which was valuable in looking at different patterns and features of the data obtained from various sensors. The next task then, was to classify this data as either damaged or undamaged. Classification problems are very common in the data mining and machine learning applications. As such, several machine learning algorithms have been proposed and developed over the years for solving such problems. For our project, we have evaluated and tested five of those algorithms namely, Logistic regression, polynomial regression, artificial neural networks, recurrent neural networks and random forests.

4.1 Explanation of the Problem:

Our Kaggle Project was focused on Binary classification. Binary Classification is the task of classifying the elements of a set into two groups on the basis of a classification rule. Our problem revolved around classifying the structural damage detected in a structure. We had to choose different machine learning algorithms to help classify the data.

4.2 Steps For each algorithm:

First we had to create a model, softmax or sigmoid used in output layer, so that we could train the model and tune hyperparameters, such as accuracy, precision etc. Finally, we had to test the performance of model against the sample data. This allowed us to compare results from different models and see what model was the most accurate at classifying the data.

4.3 Data Preparation:

First we had to split the train dataset into 2, one set for training and the other for validation. Then we had to shuffle the train dataset before using it. We then used regularizations to apply penalties on layer parameters or layer activity during optimization. Dropouts were used to incorporate non-linearity. The data was normalized by subtracting the mean and dividing by standard deviation.

4.4 Method 1: Logistic Regression:

Logistic regression is a technique commonly typically used for predicting binary classes but can also be used for multi-class problems and is adopted from the field of statistics. It describes and estimates the relationship between one dependent variable and the independent variables. Logistic regression is a special case of linear regression that produces a constant output which is categorical in nature. Thus it is based on the linear regression equation:

$$y = \beta + w_1 x_1 + w_2 x_2 + w_3 x_3 ...$$

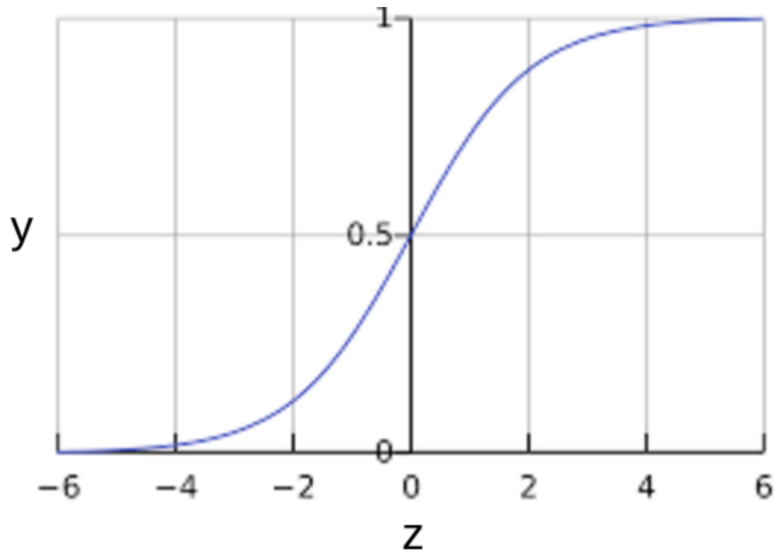

Figure 4.1: Logistic Regression (Sigmoid Activation Function)

4.4.1 Justification for Logistic Regression:

The (sometimes surprising) observation is that this is still a linear model. Logistic regression is one of the most simple machine learning algorithms that can be used for binary classification problems. It is easy to implement and can be used as a baseline for any binary classification problem. As such, we decided to use it as a baseline to which all the other models will be compared with. It can also be built upon for developing more complex machine learning algorithms for deep learning. It computes a probability output and in order to map this output to a binary category we needed to define a classification threshold (also known as the decision threshold). However, a linear regression does not necessarily yield an output between 0 and 1. Therefore, to ensure the output probability is always between 0 and 1 we used a sigmoid activation function in the output layer. Different combinations of hyperparameters were tested for this algorithm, however to save computation time an epoch value of 20 was used with a learning rate of 0.001. A classification threshold of 0.2 was also found to work best for the given dataset. Although logistic regression is sensitive to outliers and multicorrelation between the features, our exploratory data analysis showed we do not have this problem for our given dataset. Therefore, we expect to achieve reasonable results using this method.

```
# Add the feature layer (the list of features and how they are represented)
# to the model.
model.add(feature_layer)

# Funnel the regression value through a sigmoid function.
model.add(tf.keras.layers.Dense(units=1, input_shape=(1,),
                                activation=tf.sigmoid),)

# Call the compile method to construct the layers into a model that
# TensorFlow can execute.  Notice that we're using a different loss
# function for classification than for regression.
model.compile(optimizer=tf.keras.optimizers.RMSprop(lr=my_learning_rate),
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=my_metrics)
```

Figure 4.2: Overview of Logistic Regression Model

4.5 Method 2: Polynomial Regression:

One common pattern within machine learning is to use linear models trained on nonlinear functions of the data. This approach maintains the generally fast performance of linear methods, while allowing them to fit a much wider range of data.

For example, a simple linear regression can be extended by constructing polynomial features from the coefficients. In the standard linear regression case, the model is based on the expression previously shown above. If we want to fit a paraboloid to the data instead of a plane, we can combine the features in second-order polynomials, so that the model looks like this:

$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 {x_1}^2 + w_5 {x_2}^2$ del: to see this, imagine creating a new variable

$z = [x_1,\ x_2,\ x_1 x_2,\ {x_1}^2,\ {x_2}^2]$

With this re-labeling of the data, our problem can be written as:

$y = w_0 + w_1 z_1 + w_2 z_2 + w_3 z_3 + w_4 z_4 + w_5 z_5$

4.5.1 Justification for Polynomial Regression:

Sometimes the relationship between the dependent and independent variables maybe non-linear. As in this case, we do not expect a linear relationship between the features and the target variable, it made sense to try a non-linear algorithm. The simplest and most common non-linear method to use is the polynomial regression as described earlier. Intuitively, we would expect a regression model with higher orders to perform better than a simple one. However, we didn't find significant improvements between 2nd degree and higher order polynomials. Therefore, for this project we chose to use 2nd degree polynomial regression models to help save computation time.

```
from sklearn.preprocessing import PolynomialFeatures

poly_reg = PolynomialFeatures(degree=2)

X_train_2_d = poly_reg.fit_transform(X_train)
X_test_2_d = poly_reg.transform(X_test)

lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train_2_d,y_train)

test_pred = lin_reg.predict(X_test_2_d)
train_pred = lin_reg.predict(X_train_2_d)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('====================================')
print('Train set evaluation:\n_____________________________________
_')
print_evaluate(y_train, train_pred)
```

Figure 4.3: Overview of Polynomial Regression Model

4.6 Method 3: Artificial Neural Networks:

The second method utilized was Artificial Neural Networks. Artificial Neural Networks are more complex than logistic regression. This method adds a bunch of hidden non-linear layers to the logistic regression model. Our group used this method to check if it offered an improvement on the previous model. The artificial neural networks yielded average results with great accuracy ~96%, Improved precision ~45%, Improved recall ~60%, although the metrics were not good overall. The model is shown in figure 4.4.

```
model.add(tf.keras.layers.Dropout(rate = dropout_rate))

model.add(tf.keras.layers.Dense(units = 8,
                                kernel_regularizer=tf.keras.regularizers.l2(l=reg)))

model.add(tf.keras.layers.Dense(units = 4, activation='relu',
                                kernel_regularizer=tf.keras.regularizers.l2(l=reg)))

model.add(tf.keras.layers.Dense(output_values, activation="softmax"))

model.compile(
    optimizer = tf.keras.optimizers.Adam(learning_rate=lr1),
    loss = tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics = [tf.keras.metrics.SparseCategoricalAccuracy()])
```

Figure 4.4: Visualization Artificial Neural Networks

4.7 Method 4: Recurrent Neural Networks:

Recurrent neural networks (RNNs) are a very powerful tool, however they suffer from the vanishing gradient problem. In order to avoid this issue, we have used a better variation of RNNs called Long Short Term Networks (LSTM) for this project. This basically consists of cells that are responsible for "remembering" values over a time interval. This differs from the traditional neural network in that not only does it learn from the features, but it also takes care of sequence values over time. That is, for traditional neural networks it is assumed that all inputs and outputs are independent of each other. However, for RNNs the output depends on the previous computation. As a result of this, RNNs are very popular for sequential data such as time series because they perform much deeper understanding of sequence when compared with other algorithms. They are also applicable to any data that can be rearranged to resemble sequential data. In terms of visualization, one can think of RNN models as shown below:

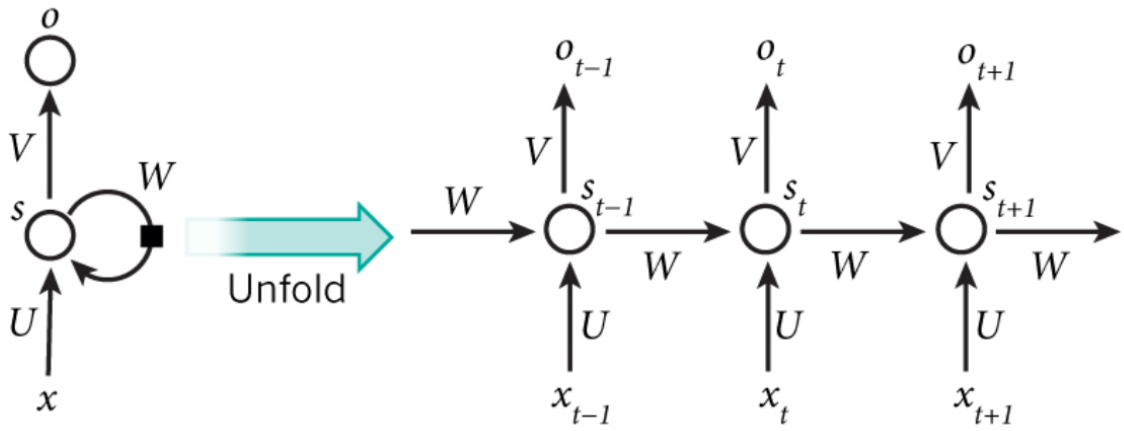

Figure 4.5: Visualization of Recurrent Neural Network Models

4.7.1 Justification for recurrent neural network:

Since, the index for our dataset actually represents time, then perhaps we can consider it as a time series data. As such, we decided to see how RNNs will perform in comparison with the other models. As was mentioned earlier, we used LSTM for this project. This type of model requires a 3-dimensional input, therefore we had to transform the shape of the dataset we had to make use of this model. In order to save computation time, a LSTM of 40 units was found to give the best performance and therefore used for testing the model. Additionally, a sigmoid activation function was applied in the outer layer. The hyperparameters were the same as with other models to ensure a fair comparison between the different algorithms used.

```
model3 = tf.keras.Sequential()



model3.add(tf.keras.layers.LSTM(40))

model3.add(tf.keras.layers.Dense(units=1,activation='sigmoid'))
model3.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='binary_crossentropy',
    metrics= [
      tf.keras.metrics.BinaryAccuracy(name='accuracy',
                                      threshold=classification_thres
hold),
      tf.keras.metrics.Precision(thresholds=classification_threshol
d,
                                 name='precision'
                                 )
```

Figure 4.6: Overview of Recurrent Neural Network Model

4.8 Method 5: Random Forest Regression:

The third method utilized was Random Forest Regression. Random Forest Regression performs both regression and classification tasks with the use of multiple decision trees and bagging. It is easy to use and often returns good results even without hyperparameter tuning. Our group used this method to check if it offered an improvement on the previous model. The Random Forest Regression provided the worst results with extremely inaccurate accuracy rate, while also not working well for the type of data we had. The model is shown below in figure 4.5.

```
from sklearn.ensemble import RandomForestClassifier

rf_reg = RandomForestClassifier(criterion="gini",
                          n_estimators=750,
                           min_samples_split=10,
                           min_samples_leaf=1,
                           max_features="auto",
                           oob_score=True,
                           random_state=0,
                           n_jobs=-1)
rf_reg.fit(X_train, y_train)
```

Figure 4.7: Visualization of Random Forest Regression Models

4.9 Problem with Model:

We noticed that the problem was the precision and recall were very low even though accuracy is high. This can be shown in figure 4.8 below. Low Recall occurs because most positive values are not predicted. This was the case for our dataset as the damaged data was too sparse. The data was too sparse because only 3.5% of the data represented the damaged condition. Because of this the model was not able to catch the postive class resulting in mostly negative data, thus the low recall rate occurs even though there is a high accuracy rate. The accuracy rate is high because the model is still accuratly predicting according to the data provided.

```
Number of damaged data points:  9512
Number of undamaged data points:  260981
Percentage of data that has the damaged condition:  3.52 %
```

Figure 4.8: Problem with Model

The solution was too add copies of the damaged data into the training set. We did this because we found that sparse data leads to models not learning properly. This allowed the data in the model to be more balanced leading to a higher recall and more efficent results. This can be shown in figure 4.9 below.

```
Number of damaged data points:  85608
Number of undamaged data points:  260981
Percentage of data that has the damaged condition:  24.7 %
```

Figure 4.9: Solution to Issue

5. Results

5.1 Metrics

Metrics are an important aspect of machine learning because they provide an insight to the performance of the model. Therefore, it is important to think about the metrics that need to be used in order to properly judge a model's behavior and also allow the model to better improve its predictions. The metrics that were used in this project include root mean squared error, R^2 value, accuracy, precision, and recall.

5.1.1 Root Mean Square Error

Root mean square error is a metric commonly used in statistics for measuring the difference between predicted and actual values for a set of data. It is calculated by taking the square root of the sum of the squares of the difference between the predicted and actual values. Because of the way it is calculated, the RMSE can never be negative. A lower RMSE is better, meaning that the predictions are closer to the actual values.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Figure 5.1: Formula for RMSE [1]

5.1.2 R^2 Value

The R^2 value, also called the coefficient of determination, is a measure of how well a model is fit to the data. It gives an indication of how likely the model is to be able to predict a future value well based on the current data and outputs a number between 0 and 1, where higher numbers are desired. The R^2 calculation is based on the correlations between the independent variables and the correlations between the independent variables and the dependent variable. The equation can be found below in Figure 5.2. There, c is the vector containing the correlations between the independent and the dependent variables, and R is the correlation matrix of the dependent variables.

$$\boldsymbol{R^2 = \mathbf{c}^\top R_{xx}^{-1}\, \mathbf{c},}$$

Figure 5.2: Formula for R^2 [2]

5.1.3 Accuracy

Accuracy, as it is used in machine learning, is a measure of the fraction of datapoints correctly identified by the model over the total amount of datapoints. It is calculated by the formula seen in Figure 5.3. It is one of the most common metrics for determining the results of a statistical problem.

$$\text{Accuracy} = \frac{\text{True Positives + True Negatives}}{\text{All Samples}}$$

Figure 5.3: Formula for Accuracy [3]

5.1.4 Precision

Precision is a measure of the fraction of datapoints identified correctly over the total amount of datapoints identified for a given problem. This shows how often datapoints are incorrectly identified by the model. It is calculated as given in Figure 5.4.

5.1.5 Recall

Recall is the ratio of how many identified datapoints make up the total set of that category of data. Recall is the indicator of how well the model is finding all of the data being sought. It is calculated as given in Figure 5.4.

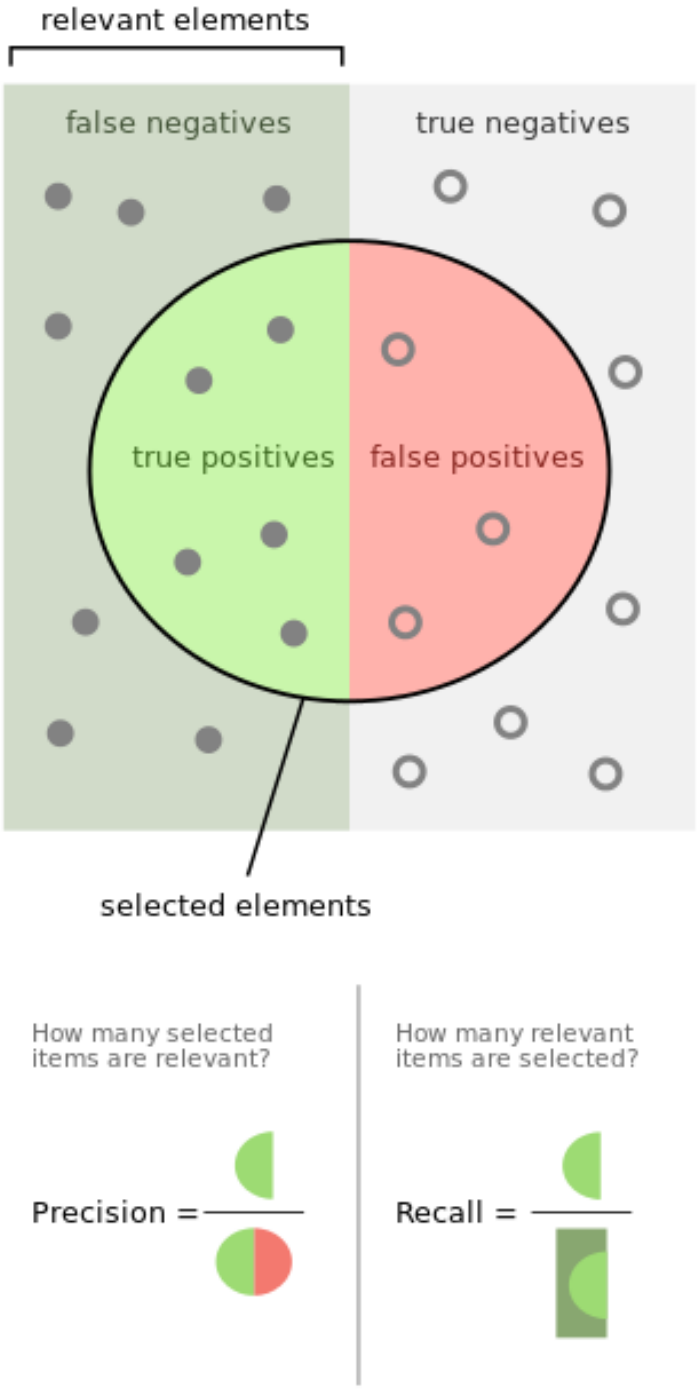

Figure 5.4: Formulas for Precision and Recall [4]

5.2 Logistic Regression

Logistic regression proved to perform relatively well in this experiment, especially once the dataset was augmented with additional copies of the sparse data. First looking at the results before the dataset was changed, the model showed some unexpected results, as seen in Figure 5.5. The results of the validation set matched closely to the training set, however this did not mean that the model was working properly. The accuracy was very high, around 96%, while the precision and recall ended up very low, both around 20%. This was a symptom of the sparsity of the dataset, as mentioned previously in the report. In addition, the recall started out very high and plummeted to a low value, which is a very unexpected behavior for the model. The model was clearly unable to learn the difference between an undamaged vs damaged data point. This is evidence of the fact that precision and recall are much more important metrics than accuracy for our experiment with the given dataset.

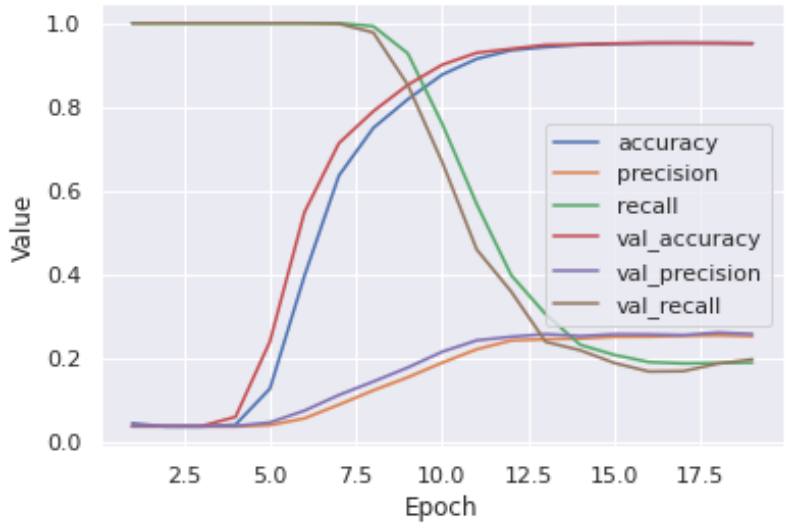

Figure 5.5: Results of Logistic Regression Model Before Dataset Augmentation

Once the dataset was augmented, the model produced a much improved result. As seen in Figure 5.6, the recall and precision are much improved from before. The recall was now 99%, and the precision went up to ~60%. The accuracy did suffer, now ~83%, however the model is much improved overall. The results showed that this model was correctly identifying almost all damaged datapoints, given the nearly perfect recall, and was incorrectly identifying a good amount of undamaged datapoints as damaged. This is not an ideal result but it is much better than missing damaged datapoints.

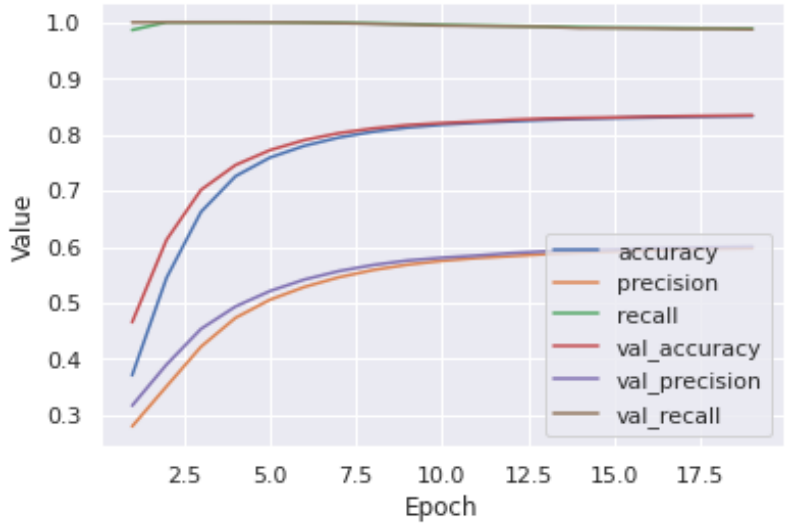

Figure 5.6: Results of Logistic Regression Model After Dataset Augmentation

5.3 Polynomial Regression

Polynomial regression did not perform very well in this experiment. As seen in Figure 5.7, the low R^2 value of ~0.175 in the validation set showed that the predictions of the model were not very well correlated with the actual values. Once the dataset was augmented and rerun through the model, the R^2 value greatly increased to around 0.537. This result is shown in Figure 5.8. Although this result is still quite low, the improvement is very noticeable and shows that a more refined model using polynomial regression may be useful. The RMSE of each run can also be compared. The first run had a lower score of 0.167 in comparison to the 0.292 of the second run. This is not what was expected, since a better model should have both a lower RMSE and a higher R^2 value. This result may be explained by the predictions of the model in the second run being more well correlated but having a

larger variance from the actual values. This could likely have been due to the fact that a much larger fraction of the data after augmentation was the damaged condition, which increased the variance of the dataset.

```
Test set evaluation:
----------------------------------
RMSE: 0.16681445337668216
R2 Square 0.17549483728169013
==================================
Train set evaluation:
----------------------------------
RMSE: 0.16728229185764404
R2 Square 0.1770909968453942
```

Figure 5.7: Results of Polynomial Regression Model Before Dataset Augmentation

```
Test set evaluation:
----------------------------------
RMSE: 0.29228008649886505
R2 Square 0.5374867907568724
==================================
Train set evaluation:
----------------------------------
RMSE: 0.29307223237197116
R2 Square 0.5395556169239921
```

Figure 5.8: Results of Polynomial Regression Model After Dataset Augmentation

5.4 Artificial Neural Network

The artificial neural network performed the best out of all the methods attempted. Using the original data, this model had a similar result to the logistic regression model, although with improved precision and recall of ~45% and ~60%, respectively. These results can be seen in Figure 5.9.

Figure 5.9: Results of Artificial Neural Network Model Before Dataset Augmentation

After the dataset was augmented and the model re-ran it produced a fantastic result that was far better than the rest of the models that were used. The model now had an accuracy of about 98%, a precision of ~91%, and a nearly perfect recall of 99.8%. This can be seen in Figure 5.10. These results show that the issue of sparse data was the only issue preventing this model from performing very well. It should be noted that the validation set precision did deviate away from the training set precision, which was likely a limitation of how well the model was able to learn the particularities of the dataset, which resulted in some undamaged datapoints being falsely labeled as damaged. Nevertheless, the extremely high recall shows that almost all of the damaged datapoints were found correctly, which was far more important.

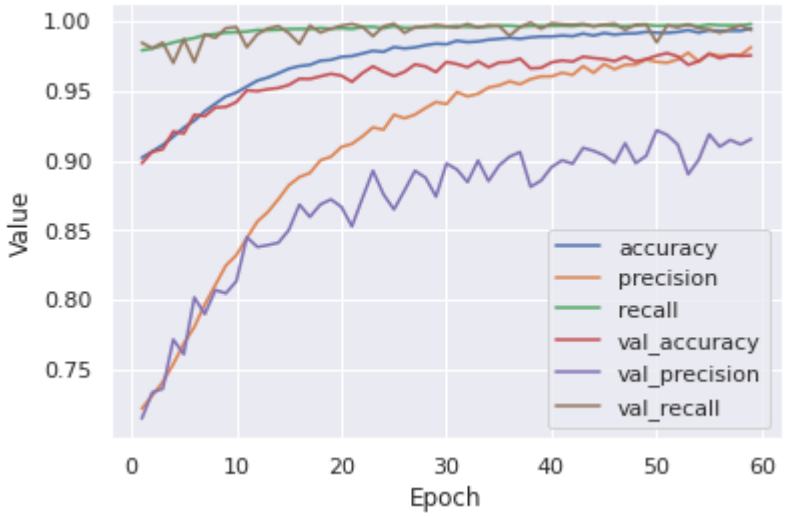

Figure 5.10: Results of Artificial Neural Network Model After Dataset Augmentation

5.5 Recurrent Neural Network

The recurrent neural network model performed in a similar manner to the logistic regression and artificial neural network models. Before data augmentation, the model resulted in high accuracy but lower precision and recall, 20% and 34%, respectively. This is in line with the results of the other two mentioned models. The results are seen in Figure 5.11. After data augmentation, the accuracy suffered, going down to ~86%, but the precision increased to 62% and the recall to 98%. These results are on par with the logistic regression model, but now are far below what the artificial neural network was able to accomplish. Figure 5.12 shows these results.

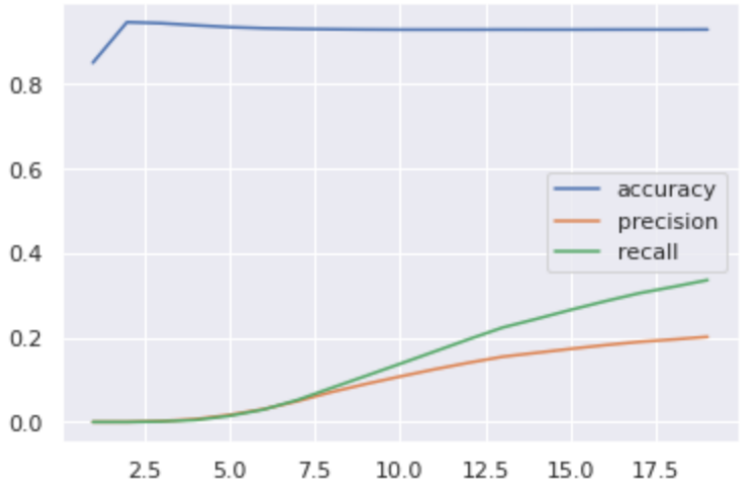

Figure 5.11: Results of Recurrent Neural Network Model Before Dataset Augmentation

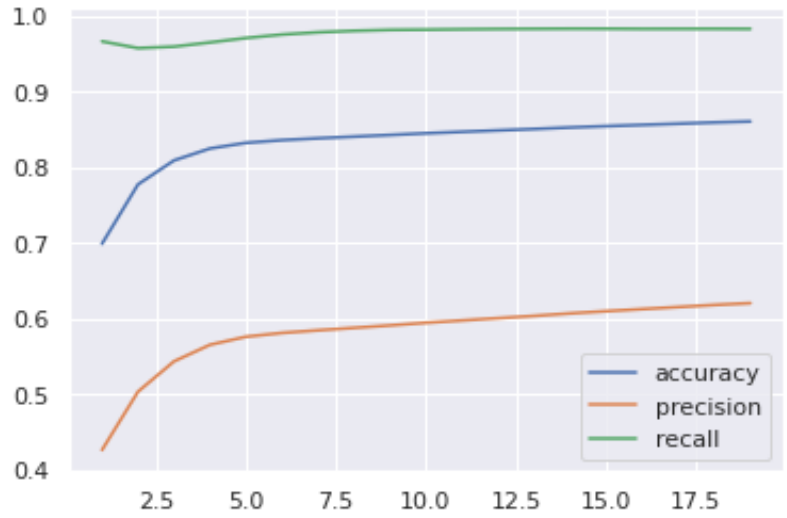

Figure 5.12: Results of Recurrent Neural Network Model After Dataset Augmentation

5.6 Random Forest Regression

The random forest regression model can be most closely compared to the polynomial regression model. Before the dataset was augmented, this modal fared worse than the polynomial regression with an R^2 value of 0.092. The results are shown in Figure 5.13. It is seen that this model is a very poor fit for the type of dataset used in this project, since it did the worst out of the 4 models tried. After dataset augmentation, this model also vastly improved, up to an R^2 of 0.408, but was still not satisfactory. These results can be found in Figure 5.14. This model also shared the peculiar result of a higher RMSE after the augmented dataset was run and can be similarly explained.

```
Test set evaluation:
-----------------------------------
RMSE: 0.17506760173939853
R2 Square 0.09189181374302258
Train set evaluation:
-----------------------------------
RMSE: 0.17555483713913608
R2 Square 0.0936885337737473
```

Figure 5.13: Results of Random Forest Regression Model Before Dataset Augmentation

```
Test set evaluation:
-----------------------------------
RMSE: 0.3307290084792896
R2 Square 0.4077974625510703
Train set evaluation:
-----------------------------------
RMSE: 0.3318000408441319
R2 Square 0.4098251410867356
```

Figure 5.14: Results of Random Forest Regression Model After Dataset Augmentation

6. Conclusion

6.1 Key Takeaways

The first takeaway from the results of this project is that artificial neural networks are the best option for binary classification problems like this project. This is not surprising, given the popularity of ANNs. They are one of the most widely applied and useful machine learning algorithms and are able to take on most, if not all, machine learning problems and give a great result.

The second takeaway is that datasets such as ours that have very sparse data may result in poor learning of the model. In our case, the dataset had barely any damaged datapoints in comparison to undamaged, so the model was not able to learn how to identify between the two. Our solution to this was to add copies of the damaged data that we did have in order to allow the model to see them more. This is dangerous, however, because this does run the possibility of overfitting to those specific datapoints. Therefore, there is a limit to how many copies should be included in the dataset to avoid these issues.

6.2 Applications

The results of this project show that machine learning has a place in structural health monitoring. A team with more machine learning experience and more data can create a model that can determine structural damage in real time and thus may be able to prevent or predict a structural collapse and save lives. The idea can be taken even further and can be adapted and expanded to allow for detection of damage in specific regions or even specific structural members in the structure, as well as predicting useful structural life. This can save large amounts of money that are spent on structural inspections, repairs, and rebuilding after catastrophic failures.

6.3 Concluding Statement

This project has shown that with relatively simple data important structural behaviors can be recognized. With this project as a proof of concept, future models can be made to simplify and vastly improve the efficiency of structural health monitoring.

References

1. **What does RMSE really mean?**
   James Moody
   Medium (2019-09-06) https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e

2. https://wikimedia.org/api/rest_v1/media/math/render/svg/2543e0a46d792a8bb93c4789e5aa10a7a3693cbc

3. https://lawtomated.com/wp-content/uploads/2019/10/Accuracy_2.png

4. https://upload.wikimedia.org/wikipedia/commons/thumb/2/26/Precisionrecall.svg/350px-Precisionrecall.svg.png