Accelerometer Test Time Reduction with Machine Learning

by

Nicholas Debeurre

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2019 by the
Graduate Supervisory Committee:

Sule Ozev, Co-Chair
Sarma Vrudhula, Co-Chair
Margaret Kniffin

ARIZONA STATE UNIVERSITY

December 2019

ABSTRACT

With the steady advancement of neural network research, new applications are continuously emerging. As a tool for test time reduction, neural networks provide a reliable method of identifying and applying correlations in datasets to speed data processing. By leveraging the power of a deep neural net, it is possible to record the motion of an accelerometer in response to an electrical stimulus and correlate the response with a trim code to reduce the total test time for such sensors. This reduction can be achieved by replacing traditional trimming methods such as physical shaking or mathematical models with a neural net that is able to process raw sensor data collected with the help of a microcontroller. With enough data, the neural net can process the raw responses in real time to predict the correct trim codes without requiring any additional information. Though not yet a complete replacement, the method shows promise given more extensive datasets and industry-level testing and has the potential to disrupt the current state of testing.

ACKNOWLEDGMENTS

I would like to acknowledge and thank several amazing individuals who helped me along the way:

- Professors Ozev and Vrudhula and Peggy Kniffin from NXP, my committee for this thesis
- Ray, Peter, Brandon, and Peggy, my manager and coworkers from NXP, for their support, guidance, and encouragement
- Audrey, for her continuing love and support through every late night and early morning
- My family for encouraging me to take on the challenge of a master’s degree
- Patty Pun, my manager at Qualcomm, for cheering me on from San Diego and for providing me time and resources to complete my thesis remotely
- Arzuhan, my ASU advisor, for helping me organize all aspects of my degree

This degree is an amazing accomplishment for me and would not have been possible without help from everyone along the way.

TABLE OF CONTENTS

Page

LIST OF FIGURES...v

CHAPTER

1 INTRODUCTION 1

2 PROBLEM BACKGROUND 3

3 THE CURRENT APPROACH 6

4 THE PROPOSED APPROACH 9

5 EXPERIMENT 12

Section 1 – Hardware..12

Section 2 – Data Collection and Correlation 14

Section 3 – Neural Network Composition 18

6 RESULTS/DISCUSSION 21

REFERENCES 23

LIST OF FIGURES

Figure Page

1. Model Flowcharts 11

2. High-Level Hardware Setup 14

3. 10 Data Captures 17

4. 100 Data Captures 17

5. 2000 Data Captures 17

6. Mahalanobis Distance of Bitstream Data 18

7. Trim Code Distribution 24

CHAPTER 1

INTRODUCTION

In the world of semiconductors and electronics, testing individual products prior to delivery is an integral part of the design and manufacturing process. It is particularly important for safety-critical applications where final test is a crucial part of insuring that any product brought to market meets the highest standards of quality. While the importance of testing is evident to many, it is often overlooked or underestimated when preparing project specifications, which can lead to delays in product development or release.

The automotive industry specifically, along with most other transportation industries, is one of the most safety-critical industries due to the consumer nature of the products being delivered. It requires extensive testing, above and beyond the normal testing procedures, to ensure components can be as accurate and reliable as possible. One needs only to look as far back as the Boeing 737 MAX 8 incidents in October 2018 and March 2019 to highlight the importance of safety and predictability in the consumer market [1, 2, 3]. A rushed product lead to a worldwide negative spotlight on Boeing that hurt their business in the short term and will likely have a lasting impact in terms of consumer confidence in the long run. While the issue was eventually attributed to software as opposed to failing sensors, the fallout as a result of the incident is extraordinarily similar to that of any potential issue and highlights the importance of proper components in safety-critical systems. To this end, it is imperative that companies

dedicate a significant portion of their time to ensuring that any product brought to market achieves the highest standards of satisfaction.

Though component testing is a valuable and necessary step, it adds a significant amount of overhead to the design process. In order to reduce this costly overhead, companies are always looking for new ways to minimize the overall test time, referred to as test-time reduction (TTR), while maintaining the high standards for the components required by customers. This thesis seeks to provide a method that could significantly reduce test-time during the manufacturing and design process by both removing the dependence on expensive, specialized machinery while simultaneously reducing the required test time for any given product. This thesis will focus on TTR for an accelerometer, but the concepts presented can be expanded to other sensors and products.

CHAPTER 2

PROBLEM BACKGROUND

One of the late-stage tests during the manufacturing process used to ensure correctness of a component before final testing and release is the final trim code calculation. The trim code is a calibration metric used to ensure that the sensitivity of the device being produced is accurate to the designed target values. While there are a number of trim codes associated with any sensor, the final trim code represents the sensor system as a whole and is generally used to characterize how well any given sensor performs and correct the system to the original specifications. Trimming is a necessary step to ensure that the components behave as designed in the real world. The values that are trimmed throughout the device can refer to many different factors depending on the device, but for this project they will refer to sensitivity of the accelerometers used to collect the data. Though a device is designed to match certain theoretical standards, the manufacturing process introduces several variations due to differences between the theoretical and physical worlds. These differences are taken into account at every step in the design process and are represented overall in the final trim of the part. Like astrophysics, the differences between the system on paper and in the real world are generally abundant. Though the theoretical description may be a highly detailed representation of the real world, it cannot be and will never be perfect. Examples of these variations include silicon composition differences, microscopic cracks in the silicon, thickness of the silicon, and the curvature in the silicon among many others. These differences often lead to minute variations in conductivity or sensitivity, for example, which tend to affect the system on

an extraordinarily small scale. It is not uncommon for the differences in measurements to be in the micro, pico, or even nano ranges. Though small, these differences can have a major impact on the system's performance as a whole and must be taken into account before the product can be deemed satisfactory.

The traditional and current industry-standard method to trim a consumer-rated part involves at least one physical movement, though that number usually rises to two, three, or more in most cases. The number of physical motions that are required to ensure that the part is trimmed correctly correlates entirely to the requirements of the agencies, whether public, private, or federal, who govern the systems in which the components will be placed. With regards to airplane components, for example, the parts that are used in planes that fly in the United States must meet the minimum requirements as described by the FAA. The physical movement used to test a component could consist of a shake, flip, or spin among others and depends entirely on the type of sensor being tested. As stated, this thesis is testing and focusing solely on accelerometers, so physical shakes are the only method of physical motion being taken into account. In order to perform a proper shake, an expensive device that is extensively calibrated to shake parts in a very specific manner must be used to ensure accuracy and consistency. While there are several issues introduced by these shakers, one of the overarching issues is the cost of each of these test platforms. Not only are the machines themselves expensive, but the optional add-ons to configure the machines, recurring service, both routine an emergency, and a lifetime of calibrations raises the cost of each of these devices substantially to well over one million dollars. Like most production costs, these high equipment expenses hurt the gross

margins of the companies that are using them. These costs are then propagated forward to end customers as a way of recuperating these additional expenditures.

CHAPTER 3

THE CURRENT APPROACH

While working to reduce and ultimately eliminate these high costs, engineers at NXP Semiconductors working on TTR for various sensor projects developed a method to test devices without requiring an external shake from a tester. While the idea originated in 2007, a patent for the idea was granted to NXP in December 2015. Described in US Patents No. 9,834,438 and No. 9,221,627 titled "Compensation and Calibration of MEMS Devices," the method makes use of the responsiveness of the microelectromechanical system (MEMS) in a device to electromagnetic stimulation in order to calibrate the parts free from the necessity of physical motion [5, 6]. This method, upon full implementation and certification, would effectively eliminate the reliance on expensive shakers. Rather than have a device mechanically shake the part, a known stimulus can be used to produce the same effect. This stimulus does two things to replace the tester. First, it recreates the motion produced by the physical shake, thus eliminating the need for a shaker. Second, it provides a constant pattern with which to move the MEMS device. This response from that movement can then be correlated to the input pattern which allows for consistent testing as well as a way to correlate trim codes to responses. While the tester had previously been responsible for running the tests on the devices being tested, the machine that costs upwards of one million dollars can be replaced by a cheap microcontroller (MCU) which costs somewhere in the range of thirty to sixty dollars depending on the components [8, 9, 10]. Even if multiple MCUs are used in place of the tester, there is still an astronomical reduction in test cost without even

taking into account the reduced costs associated with TTR. In addition to reducing the initial cost, multiple MCUs could be used in tandem to parallelize the algorithm, further reducing the overall cost.

As described in the patent, the current test method consists of three parts: data collection, measurement collection, and the algorithm to produce the final trim code. The data is collected using a pseudorandom frequency to stimulate the part. The stimulus is referred to as pseudorandom because it is meant to replicate a random signal passing through the part while at the same time being known in order to produce the necessary correlations. The pattern is known outside of the system which contains the device being tested and was specifically created in order to cover all corners of the test space. The pattern must be known outside of the test system for three reasons. First, a correlation of the data would not be possible for any algorithm without knowing the pattern. Second, the stimulus for the test system must be identical each time; otherwise, there would be no way to verify that it is indeed the same without manually creating the signal. Lastly, without a defined pattern it would be impossible to ensure that all the corners of the test space are covered during testing. As the part is stimulated by the pseudorandom signal, the motion of the part as reported by the accelerometer is collected. The data can continue to be collected for as long as necessary, though there exists a trade-off between accuracy and test time that must be taken into account. By balancing the amount of data collected from the part as it reacts to the pseudorandom sequence, the result can be more or less precise to match customer specifications. Should the tolerance interval be wide enough, for example, in the case of a cheap sensor used for educational purposes, the amount of

data collected can be reduced which would then reduce both the test time and final cost. Once data collection is complete, several different measurements of the part are taken to ensure it is operating properly. Both these measurements and the previously collected raw data are fed into the algorithm and the resulting value is the expected trim code for the part. Both this new algorithm and the physical shake yield the same results, which demonstrates that the current method can be replaced with the method requiring no physical stimulus. Though current government and industry standards require at least a single shake for consumer-rated, safety-critical devices in tandem with this method, a single verification is better than multiple verifications throughout the production process and could even be removed entirely should this method succeed and pass certifications.

Though the idea for this method came about in 2007 and was later patented in 2015, development of the process is still ongoing. From May 2018 to August 2019, I was employed at NXP Semiconductors as part of the team working to develop and implement this algorithm. Prior to my time at NXP, extensive work had been done to tune the algorithm and prepare the hardware for this groundbreaking technique. My contribution to the project consisted of the software research and development required to make the algorithm run on an MCU platform. Though this current method works well, this paper seeks to further improve the speed and accuracy as well as remove the need to develop an explicit model of the signal chain for each new component.

CHAPTER 4

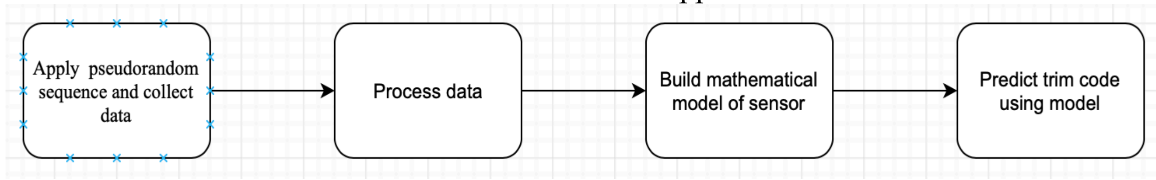
THE PROPOSED APPROACH

The goal of this thesis is to replace the current, model-based testing procedure for accelerometers with a mathematical function derived from a neural net. For the remainder of this paper, the model-based procedure will be referred to as the model method and the new method proposed by this thesis will be referred to as the neural net method. While the model method works well and is already an improvement over the physical testing, passing large amounts of data through any model takes a significant amount of time and processing power compared to what could be achieved by using a neural net. While the model training may take longer than passes through the current mathematical model, the trained model will only need to do forward propagation which will result in a shorter test time. The amortized cost of the neural net model compared to the mathematical model would be lower. Thus, to maximize the throughput of the test suite, the process can be reduced to two steps: one, collect data based on the response to a pseudorandom stimulus and, two, feed this response data into a neural net which will produce the expected trim code. This approach, as well as the current method, can be seen below in Figure 1.

In order for a neural net to be able to generate a trim code and replace the model method, a correlation must be identified in the data that is collected from all the parts. This correlation, should it exist, would need to occur between the motion of the part and the stimulus being applied, whether that stimulus be electromagnetic or physical. Given that the parts are being trimmed based on the response following a physical motion or electromagnetic stimulus, it stands to reason that there must indeed exist a correlation in

the parts. While the current method uses a mathematical model to correlate new devices, the benefit of a neural net would be that it eliminates the need for a defined correlation. It is sufficient to show that a correlation exists and can be recorded so that the neural net can be tuned to recognize the response. Once the correlation is proven, the response data can be used to train the neural net according to the trim codes produced by different responses. The main time-saving feature with this approach compared to the model-based approach is that the data does not need to pass through a mathematical model to calculate the trim code based on the correlation. Because the model takes up 80% - 90% of the computation time, eliminating that portion would reduce the test time by over 90% - 95% when compared to the current industry standard. Once the response data is correlated to the trim code that is produced, the neural net can quickly and easily identify the proper trim code for any given input data and continue to tune itself when needed. Finally, this application can be extended to multiple platforms and work for a variety of different devices. If successfully implemented, this test could save millions of dollars and reduce test time by 50%-75% or more, in turn increasing the gross margin and reducing the final cost for the manufacturing companies and their consumers.

Mathematical Model Approach



Neural Net Approach

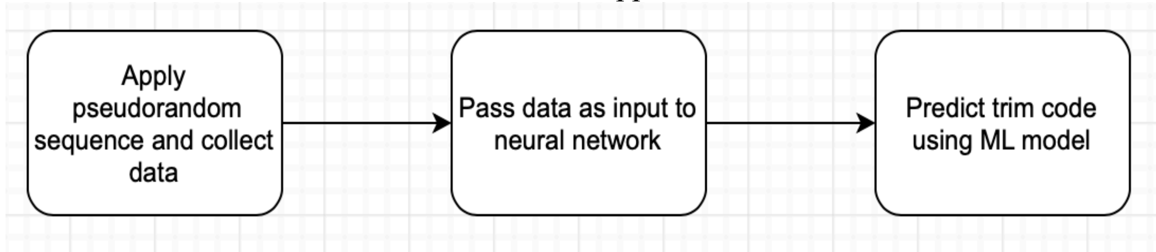


Fig. 1

CHAPTER 5

EXPERIMENT

Section 1 – Hardware

While the focus of this thesis is on the creation of a powerful neural net for signal processing, this experiment used several important pieces of hardware in order to properly collect data to be used to train and test the neural net. While the hardware setup described in this paper will be the one used to collect the data, some proprietary details will be omitted. A similar hardware setup can be used to collect the data in the same manner. The three key pieces of hardware used were the MCU development platform to drive the data collection, the socket for the accelerometer, and the accelerometer used.

The MCU development platform used for this project was the NXP FRDM-K66F [8, 9]. This particular MCU is the most powerful development platform in this package currently offered by NXP out of the box. It features a 180 MHz high-performance Arm Cortex-M4F (MK66FN2M0VMD18) with 2 MB of flash memory and 256 KB of SRAM [10]. This platform was selected for its high-speed processor, high flash and SRAM memory capacity, excellent I/O capabilities, and wide range of peripherals. While this MCU development platform is excellent in terms of the hardware offerings, the lack of extensive documentation and examples make it a tricky platform with which to work. An excellent plug-and-play alternative is the Teensy 3.6, which features an identical MCU save for 1 MB of flash memory instead of the 2 MB offered on the FRDM-K66F [11]. For future projects, the Teensy 3.6 would be the best option as it is easier to program and use due to its compatibility with the Arduino libraries.

In conjunction with the MCU development platform was the socket used to connect with the accelerometer. This particular socket is a proprietary design that belongs to NXP. The socket enables the accelerometer to connect to the MCU via a SPI connection and allows access to all the functions of the accelerometer. Additionally, it provides a level base in which to place the part that allowed the data to be collected without any noise due to the tilt of the sensor. While not explicitly necessary to communicate with the sensor, the socket used allows for the contacts on the accelerometer to be routed to test pins for easy connections. Since the socket is unavailable externally, it is sufficient to solder the sensor contacts to test pins using a perfboard.

Arguably the most important component in the hardware setup, an accelerometer was chosen for this project due to the simplicity of the design which isolates the motion to a single axis. The system within the accelerometer contains a mass connected to two capacitors by springs on either side. As the accelerometer experiences accelerations via motion, the capacitance in each plate changes depending on the distance from the mass and can be measured. The result recorded from this motion creates the response. The accelerometer used for the data collection in this thesis was NXP's MMA68xx Dual-Axis SPI Inertial Sensor. According to the data sheet, the MMA68xx is a "SPI-based, 2-axis, medium-g, over-damped lateral accelerometer for use in automotive airbag systems" [7]. This accelerometer was chosen as it was readily available for testing when I was employed by NXP. Additionally, while it provides access to any pair of the X, Y, or Z axes, each axis can be stimulated independently to keep the data processing simple. This

particular accelerometer can be replaced with a similar accelerometer provided that it is able to provide raw response data to a microcontroller via a SPI connection. The high-level diagram of the setup can be seen below in Figure 2.

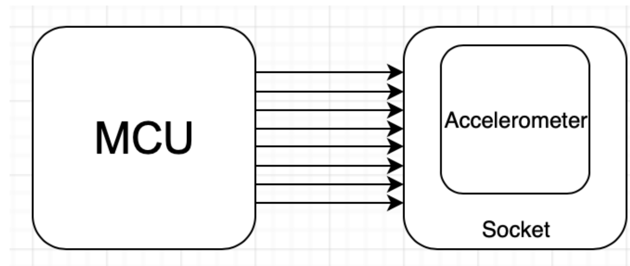


Fig. 2

Section 2 – Data Collection and Correlation

The most important step in this entire process is the data collection and correlation. Without proper data, the correlation cannot be defined, and the neural net will not run properly. In order to collect data properly, it is important to know how the data is prepared by the accelerometer and read by the MCU. The MCU is connected to the accelerometer with eight wires: a 5V power line, a ground line, a SPI slave-in line (SIN), a SPI slave-out line (SOUT), a SPI chip-select line (CS), a SPI clock line (SCLK) and two digital interrupts. The accelerometer contains two internal, 16-bit data buffers. These buffers are used to collect the raw data from the motion of the accelerometer. Each of the two buffers is connected to one of the interrupt pins. As the stimulus is applied to the accelerometer, the raw data stream fills each of the buffers bit by bit, alternating between the two as they become full. Once the first buffer is full, the accelerometer triggers the interrupt associated with it, which tells the MCU to collect the data, and begins to fill the second buffer. The entire buffer is read as a single 16-bit integer instead of bit by bit in

order to ensure that the read is fast enough. As the data is read by the MCU, the other buffer is filled with data and triggers the interrupt when complete. This pattern repeats and must be synchronized in order to ensure proper data collection. Any deviation will cause the data to be jumbled and unreadable. For each part, 2,000 of these 16-bit raw data points were captured and the total runtime per accelerometer was about 64 milliseconds. The stimulus used to collect the data for this thesis is an NXP-proprietary stimulus signal.

Once the data captures are complete, it is imperative to identify the correlation between the captured data samples. This correlation is what will link a particular pattern with a trim code. Because each part is being stimulated in the same manner, it stands to reason that each part should react in a similar manner. In a perfect world, the response for each part would be identical as would the resulting trim codes, but miniscule changes in the structure of each part can cause changes to its behavior. For multiple captures, the correlation is visually identifiable when plotted. Though not exact, each capture shows a similar behavior in response to the stimulus. As the number of captures graphed together increases, the correlation becomes easier to discern. The correlation is shown by the charts in Figures 3, 4, and 5. The correlation in these figures is a result of raw data as opposed to the bitstream. Please note that these charts represent a subset of 50 data points from the indices 700 to 750 out of the original 2,000 data points for readability and that they show 10, 100, and 2,000 data points respectively. This range was chosen at random to demonstrate the response of the part in the middle of the stimulus pattern.

Though the raw data is loosely correlated, it may not be a strong enough correlation to result in a proper trim code output. The issue lies in the method that the raw

data is collected. Because the data is arbitrarily “chopped” when it is placed in the buffers, each data point does not necessarily or properly represent the data as a whole. While the goal is to limit the data pre-processing as much as possible, it may be necessary in order to accurately reflect the data captured in the response. In order to correctly represent the data set should the raw data not suffice, each data point can be converted back into the 16-bit representation and connected to each of the others in a single stream to recreate the original bitstream. In order to ensure that the data is correlated with no outliers when it is converted back into a bitstream, the mahalanobis distance can be calculated between each bitstream in the dataset. The mahalanobis distance is a unitless metric that provides a way to represent the data as a cluster in order to easily identify outliers [12]. If the data is correlated, the data should cluster when graphed. The mahalanobis distance for the neural net input data yielded the graph shown in Figure 6 below. Please note that this graph represents a random sample of 500 of the bitstreams compared against each other. Sampling was used to reduce the processing time of the result and the result represents the dataset as a whole. Based on the correlation of the raw input data as well as the processed bitstreams, it is clear that the data does indeed show a correlation that can be used by the neural net.

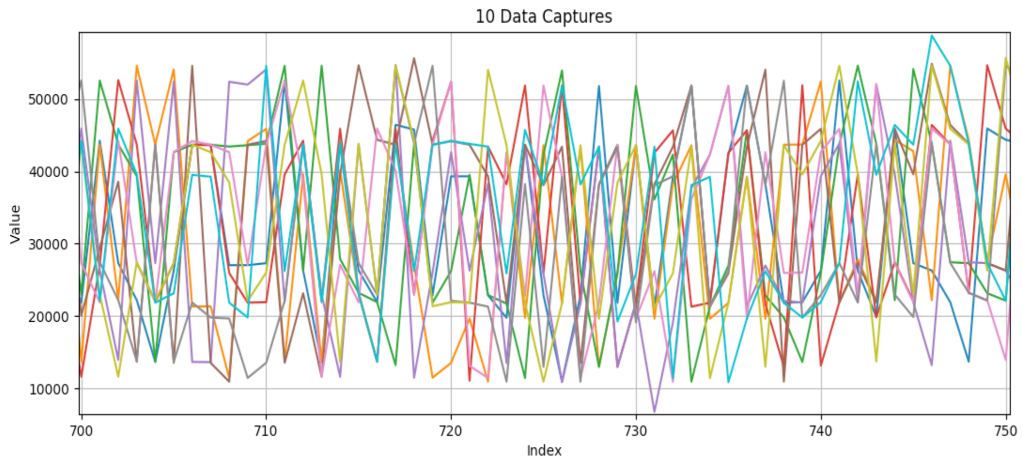


Fig. 3

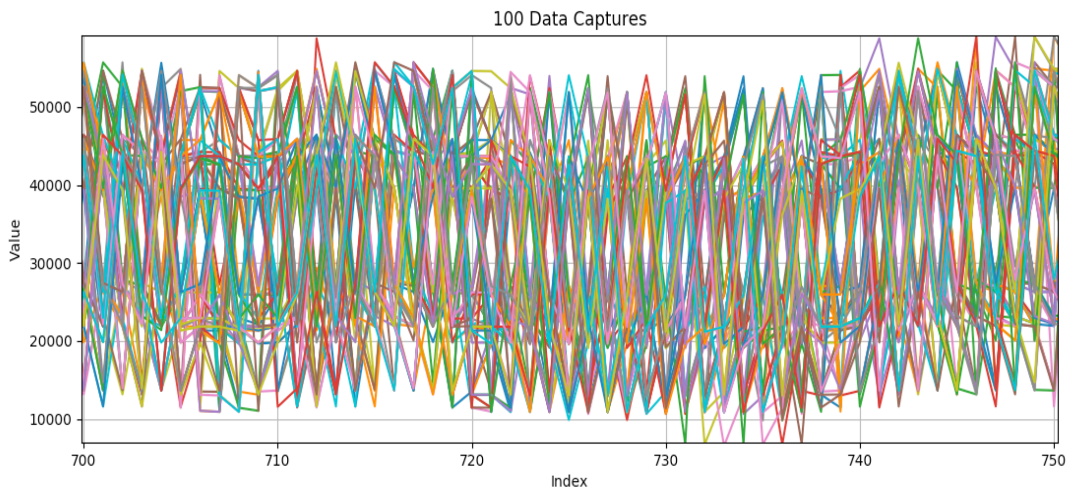


Fig. 4

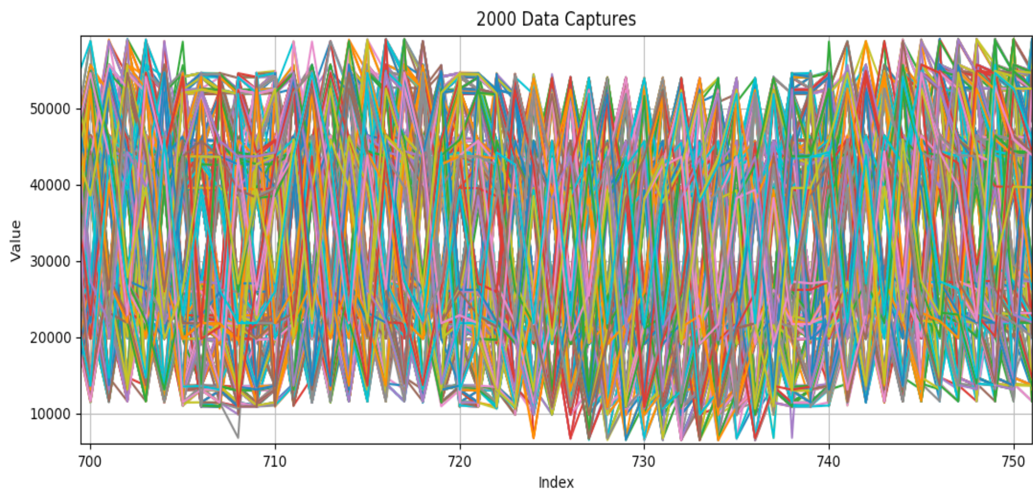


Fig. 5

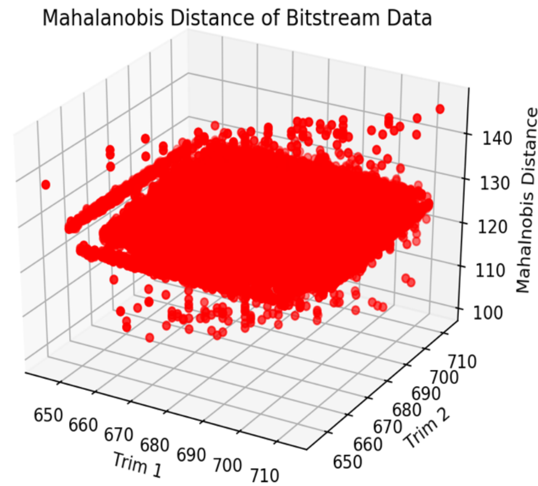


Fig. 6

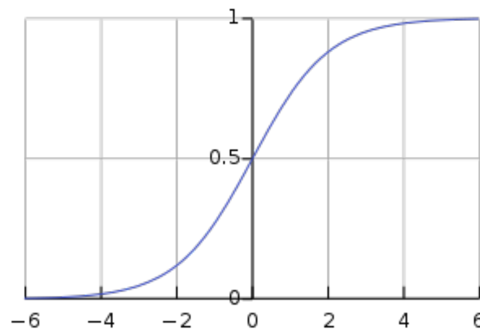
Section 3 – Neural Network Composition

There are many different types of neural nets, from a single node with an activation function to the incredibly complex deep-learning neural nets with millions of inputs, layers, nodes, and outputs. Though all these different compositions exist, machine learning is not an exact science and there is certainly no single solution to solve a given problem. It takes an incredible amount of patience, endurance, knowledge, and even luck to find the proper composition.

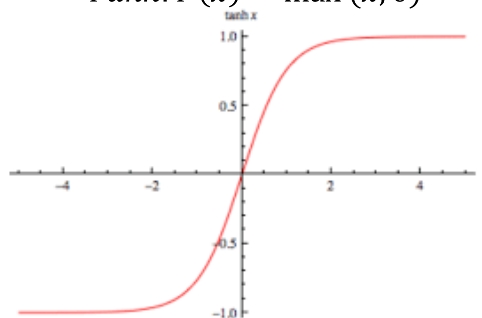
For this thesis, a fully connected, deep neural net with a regression model was used to correlate the sensor data to the trim data. A regression model, as opposed to its counterpart the classification model, is a technique used in machine learning and, more broadly, data science to correlate an input to a specific value. A classification model, on the other hand, categorizes inputs into groups. The composition of this neural net consists of fifteen inner layers, a single input layer with 2,000 or 32,000 inputs depending on the type of input, and a single output value. Added to the inner layers were dropout layers to

ensure that the training remained randomized and did not "learn" the input data and overtrain. For each layer, the rectified linear unit (ReLU) function was used as the activation function and the Adam Optimizer Algorithm was used with mean-squared error to train the neural net during the backpropagation step. The ReLU function was chosen as the activation function because it is able to solve the vanishing gradient problem [14, 16]. The vanishing gradient, an issue that exists in the sigmoid and tanh activation functions, occurs because the derivative of the functions for large input becomes very small due to the curve in the graph of both functions. The sigmoid, tanh, and ReLU functions are defined by the following equations and graphs:

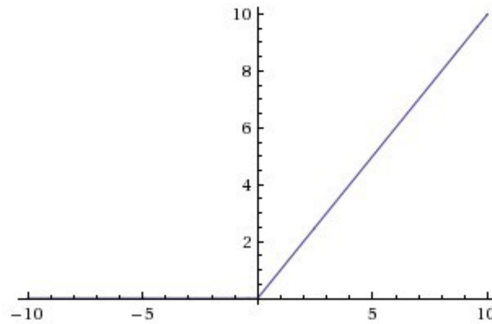
$$Sigmoid: F(x) = \frac{1}{1 + \mathrm{e}^{-\mathrm{x}}}$$



$$Tanh: F(x) = \max\,(x, 0)$$



$$ReLU: F(x) = \max\,(x, 0)$$



Additionally, the ReLU function is extremely efficient for computation because it is nothing more than a simple comparison between the input and zero. Similarly, the Adam Optimizer algorithm was selected for this neural net because it has been shown to be the most efficient algorithm for neural net training [15].

Two different neural net input methods will be examined within the same neural net to identify which method leads to a stronger output. The first method is to correlate trim codes using the raw data without first converting the stream back into a bitstream. The drawback to this method is that the arbitrary chunking previously discussed may weaken or destroy any existing correlation and lead to binning. Binning is when large chunks of inputs, or bins, lead to the same result. Should the initial method fail, the second method is to interpolate the raw data back into the bitstream format and correlate the data to the trim codes using the bitstream. The trade-off between the two methods is the processing speed versus the accuracy of the correlation. By not converting the raw values into a bitstream, the first method would result in a greater TTR than the second by removing some extra pre-processing. The downside is that the preprocessing may be necessary in order for the neural net to produce an adequate result. For both of these methods, the output will be the same: the trim code for the sensor being tested.

CHAPTER 6

RESULTS/DISCUSSION

The main reason behind training the neural net using the raw signals from the accelerometer was to increase the TTR by reducing the data processing time. By training a neural net to identify the trim code for an accelerometer based on the raw 16-bit chunks, approximately 1 second is saved for every 500 to 1000 parts that are trimmed. While that value is not much on an individual basis, this results in an aggregate TTR of between 17 and 34 seconds per million parts that are trimmed. This TTR does not take into account the time saved when switching from the model-based method to the machine learning method. For a production-level device which will see many millions of parts tested and trimmed, this is a significant savings.

After extensive testing, a major flaw presented itself in this hypothesis that prevented the raw data from properly correlating with the trim codes. Several different neural network compositions were tested in addition to the 15-layer model described earlier, but each of them behaved in the same manner. As suspected and previously discussed, the arbitrary chunking of the response data caused the correlation to become too weak for the neural net. As the buffers fill and are passed to the MCU, the arbitrary chunking, while correlated, is not able to properly preserve the signal that is outputted by the accelerometer. In essence, the chunking is blurring the data. Compounding this is the fact that there is only a finite distribution of trims codes as shown below in Figure 7. Since the maximum error between each end of the trim code spectrum from the center is about 5%, the neural net is training to the exact center of the range of trim codes. In the

worst case, the trim for the part is going to have a 5% error associated with it. While this does improve over the original maximum 10% error from the mathematical model, it cannot be a viable replacement because the trim codes are arbitrary. Since every part comes out with the exact same trim code, there is no way to tell if the part is trimmed correctly or not. It must then be assumed that they are all trimmed incorrectly, which will require a shake test to confirm in the best case or a re-trim in the worst.

Because the raw data did not hold the correlation as hoped, the second method using the bitstreams for each data capture was tested. While this method did show some promise, it did not demonstrate enough separation in the output to be considered a success. As discussed, the neural net with raw data centered the output to the middle of the trim code distribution during training. Similarly, the bitstream data did begin to show this pattern as well, although it did not train to the exact same center point. Instead, the bitstream inputs caused the output trim value to shift slightly for every part instead of producing the same output for every input. The shift was always on a scale of 10^{-3} or 10^{-4}. After discovering the shift in values, both neural networks were retrained, and the spread of the output values was captured in addition to the output values. While the model with the raw data as input continued to display the same behavior by outputting values that had a spread of 0, the net with the bitstream data as input displayed a different behavior. Instead of returning a spread of 0 for the test data, it showed a spread of 0.001. After several tests, this spread was consistently in the range $0.001 \leq x \leq 0.005$.

As a follow-up exploration, the number of nodes in each layer was steadily increased. Beginning with an initial layer size of 2,000 nodes, the inner layers were

modified by increasing the node count by 500 for several tests up to 10,000 nodes. As more nodes were added, the resulting spread increased from 0.001 up to a maximum of 4. While not remarkably useful, this does indicate that an increase in nodes should result in a more robust model that can accurately report the trim code for a particular part. The downside to this is that the training time becomes exponentially more expensive as more nodes are added. By 10,000 nodes per layer, the total number of trainable nodes in the model becomes roughly 1.8 billion, increasing the training time to approximately eight hours. Further tests were not possible due to limitations in computing resources that caused the training to crash.

The behavior exhibited by the neural provided some useful insights. Given that this was a black box approach to trimming sensors where a correlation does indeed exist, a large and complex enough network would theoretically eventually solve this problem. By inputting a more targeted input with additional parameters that better describe the accelerometer, the neural net would likely be much smaller than the final model produced for this thesis.

If this experiment were to be continued, future tests should attempt to further separate the neural network output by increasing the number of nodes per layer. A good starting point would be 32,000 nodes per layer, which would raise the total number of weighted connections between layers to over 1 billion connections. Additionally, it would be useful to reduce the number of bitstream data points to pass into the net as well as identify other parameters that may aid in the training. Giving the model a starting value may aid in the training by providing an estimate that the model can use to decide the

initial weights instead of blindly guessing and checking. Lastly, an alternative solution could be to train a separate classification model to recognize different trim codes. Since the number of possible trim codes is finite, the accelerometer trim codes could be determined by passing each response through a classification network in parallel. Each model can provide a percentage estimate of the correlation to the particular trim code to which the model is tuned. Though training may take some time and computing resources may come at a cost as well, the TTR should be similar to what could be achieved by trimming the parts directly with a regression model. In conclusion, though not a total success, this method of trimming accelerometers shows promise as a new avenue to explore for production TTR as well as providing yet another application to the growing list of existing machine learning applications.

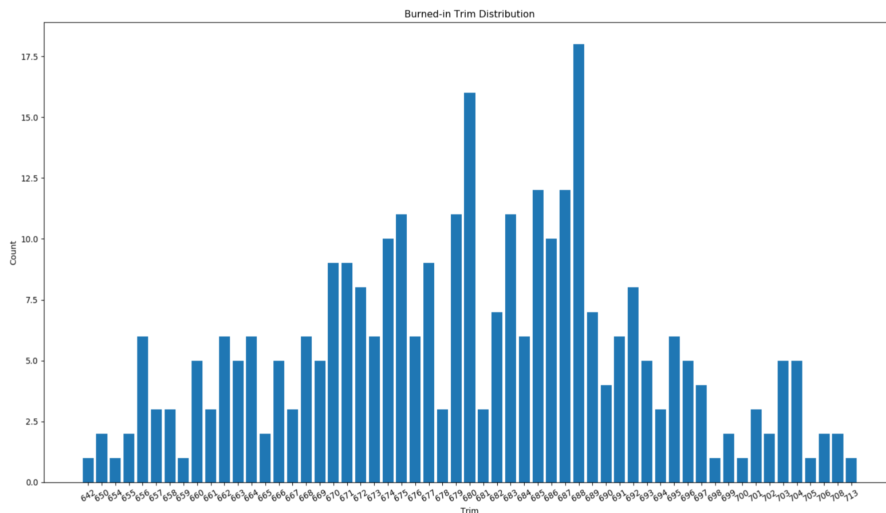


Fig. 7

REFERENCES

[1] Ellis, Ralph. “Experts Say There Were Similarities in the Ethiopian Airlines and the Lion Air Crashes. What Were They?” *CNN*, Cable News Network, 4 Apr. 2019, www.cnn.com/2019/03/18/world/boeing-737-crashes-similarities/index.html&xid=17259,15700002,15700022,15700186,15700190,15700256,15700259,15700262.

[2] Frankel, Todd C. “Sensor Cited as Potential Factor in Boeing Crashes Draws Scrutiny.” *The Washington Post*, WP Company, 17 Mar. 2019, www.washingtonpost.com/business/economy/sensor-cited-as-potential-factor-in-boeing-crashes-draws-scrutiny/2019/03/17/5ecf0b0e-4682-11e9-aaf8-4512a6fe3439_story.html?utm_term=.c5111020f2b6.

[3] Gates, Dominic. “How Much Was Pilot Error a Factor in the Boeing 737 MAX Crashes?” *The Seattle Times*, The Seattle Times Company, 15 May 2019, www.seattletimes.com/business/boeing-aerospace/how-much-was-pilot-error-a-factor-in-the-boeing-737-max-crashes/.

[4] Dar, Tehmoor M, et al. (2017). US Patent No. 9,834,438. https://patentimages.storage.googleapis.com/b9/0f/bc/0edb1063256c46/US9834438.pdf.

[5] Dar, Tehmoor M, et al. (2015). US Patent No. 9,221,679. https://patentimages.storage.googleapis.com/1f/fb/c8/c355647a02cf79/US9221679.pdf.

[6] Chatterjee, Soham. “Good Data and Machine Learning.” *Medium*, Towards Data Science, 14 Oct. 2018, https://towardsdatascience.com/data-correlation-can-make-or-break-your-machine-learning-project-82ee11039cc9.

[7] *MMA68xx, Dual-Axis SPI Inertial Sensor*. https://www.nxp.com/docs/en/data-sheet/MMA68xx.pdf.

[8] *FRDM-K66F: Freedom Development Platform for Kinetis® K66, K65, and K26 MCUs*. https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-k66-k65-and-k26-mcus:FRDM-K66F.

[9] *Kinetis K66: 180MHz Cortex-M4F MCU, 2MB Flash, 256KB SRAM, Dual USBs (FS HS), Ethernet, 144-MAPBGA*. https://www.nxp.com/part/MK66FN2M0VMD18

[10] *Kinetis K66 Sub-Family 180 MHz ARM® Cortex®-M4F Microcontroller.* https://www.nxp.com/docs/en/data-sheet/K66P144M180SF5V2.pdf

[11] *Teensy 3.6 without Headers*.
https://www.adafruit.com/product/3266?gclid=EAIaIQobChMI2KPNk7-N5QIVYh6tBh2Z4g3eEAQYASABEgJ8yPD_BwE

[12] "Mahalanobis Distance." *Wikipedia*, Wikimedia Foundation, 20 Aug. 2019, https://en.wikipedia.org/wiki/Mahalanobis_distance.

[13] Ma'amari, Mohammed. "Deep Neural Networks for Regression Problems." *Medium*, Towards Data Science, 25 Oct. 2018, https://towardsdatascience.com/deep-neural-networks-for-regression-problems-81321897ca33.

[14] Brownlee, Jason. "A Gentle Introduction to the Rectified Linear Unit (ReLU)." *Machine Learning Mastery*, 6 Aug. 2019, https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[15] Brownlee, Jason. "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning." *Machine Learning Mastery*, 6 Aug. 2019, https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[16] Arora, et al. "Understanding Deep Neural Networks with Rectified Linear Units." *ArXiv.org*, 28 Feb. 2018, https://arxiv.org/abs/1611.01491.