Unsupervised Deep Learning for Gear Health Monitoring

Tyler Cody[1], Stephen Adams[2], and Peter A. Beling[3]

[1,2,3] *University of Virginia, Charlottesville, VA, 22904, USA*
tmc4dk@virginia.edu
sca2c@virginia.edu
beling@virginia.edu

ABSTRACT

Deep learning has revolutionized many fields in recent years by replacing expert-designed, handcrafted features with learned representations. Gear health monitoring is a field where expert-designed features are heavily used for predictive modeling. This paper investigates how unsupervised deep learning can be applied to gear health monitoring to make predictions on low frequency scales using high frequency data given small, sparsely labeled data sets. Deep convolutional autoencoders are trained and used to generate learned features. The learned features are compared with relevant handcrafted features via their performance in training machine learning models to predict discrete gear fatigue states. The learned features performed poorly against the handcrafted features, however models trained on feature sets tended to outperform those exclusively trained on handcrafted features. The top performing model was a multi-layer perceptron trained on both feature sets that leveraged the ability of the condition indicators to represent healthy and failure states and the ability of the learned features to represent the intermediate worn state. This work demonstrates that unsupervised deep learning techniques can be used to bolster the performance of handcrafted features in small, sparsely labeled data sets in gear health monitoring.

1. INTRODUCTION

Gear health in rotorcraft is critical to maintaining flight, and is therefore a major concern in the field of rotorcraft health monitoring. Rotorcraft and gear health testing are data intensive processes, and the generated data are complex. There are many relationships, both known and unknown, between the features of the data set.

Gear health monitoring is an area where statistical and signal processing techniques have been the prevailing approach to tracking damage (Vevcevr, Kreidl, & Smid, 2005). These techniques are used to tackle a dilemma in gear health monitoring: the primary data measuring gears contains very high frequency information, however, gear damage typically progresses on low frequency scales. Traditionally, to cope with this issue, a class of signal processing techniques known as condition indicators are used to transform high frequency information to lower frequencies. Subsequently, models can be built on top of this new, transformed, low frequency information. While condition indicators are effective tools, they may underrepresent the inherent complexity of rotorcraft and gear health data. Trends across many fields suggest that deep learning may offer a more effective tool for representing raw, high frequency data for the purposes of model building.

Deep learning has revolutionized many fields by replacing domain-specific, expert-driven features with learned representations of data. Deep learning has been successfully applied to image analysis (Krizhevsky, Sutskever, & Hinton, 2012), natural language processing (Collobert & Weston, 2008), and machine translation (Cho et al., 2014) to perform similar tasks. However, their success has been bolstered by the availability of large, labeled data sets. Gear health data, on the other hand, is much scarcer, and when it is available, it is often sparsely labeled. Data constraints pose serious concerns to any application of deep learning. This issue is further exacerbated in a research setting where the material and structural properties of the gears often changes between experiments.

Unsupervised learning offers a method for learning a representation of the data without reliance on class labels. Successful application of unsupervised deep learning would allow for unlabeled data to contribute to the training of representation learning algorithms, and thus allow for the algorithms to harness much more of the available data than a traditional supervised approach which is limited to data that has been manually inspected.

This paper compares the ability of machine learning algorithms to predict multi-class damage states when trained on learned features, condition indicators, and a combination of

the two. The paper is structured as follows. In Section 2 a brief background on condition monitoring of gears, condition indicators, representation learning, and deep learning is given. In Section 3 the methodology is outlined, and specific motivations for using deep convolutional autoencoders are described. Section 4 gives a description of the data set and Section 5 descibes the testing process and the results. Section 6 draws conclusions and Section 7 suggests future related work.

2. BACKGROUND

Condition monitoring of gears primarily involves observing oil debris, acoustic, or vibration data. Oil debris and acoustic data are important and can be effective for monitoring rotorcraft systems. For our work, however, we are concerned with how best to interpret vibration data to predict the damage state of a gear. Condition indicators are values computed from a vibration signal that seek to help identify gearbox failures. To calculate condition indicators, the time domain signal is synchronously averaged to increase the signal to noise ratio, and then condition indicators are computed on this new averaged signal (Vevcevr et al., 2005). The specific condition indicators in our data set are specified in the data set description. In the research setting, these condition indicators are then fed into a classifier which determines whether or not to send a stop signal to the test rig to stop the gears before cracking progresses. Researchers can then analyze where the crack initially formed, and designers can use their analysis to improve performance.

We consider a different case than binary detection. We are concerned with identifying progressive states of wear from the vibration signal. Machine learning offers a powerful tool for approaching this task.

The performance of machine learning algorithms depends heavily on the representation of the data they are given. In some cases, expert-designed features, such as condition indicators, serve as sufficient representation for a given learning task, however, in many cases, it is difficult to know what features should be extracted, let alone how to accurately extract them. A solution to this problem is to not only use machine learning to discover a mapping from representation to output, but also the representation itself. This is known as representation learning (Bengio, Courville, & Vincent, 2013). Representation learning is a major area of interest in artificial intelligence systems as it enables them to discover a good set of features for complex tasks on the order of minutes and hours, whereas handcrafted features can take decades of human time and effort (Goodfellow, Bengio, & Courville, 2016).

In real-world applications, there are many factors that influence the raw data. Good features are those that can disentangle these factors and discard unnecessary information. In complex data sets, factors are difficult to identify, let alone separate. Doing so requires sophisticated, nearly expert-level understanding of the data set. Deep learning helps overcome this problem by expressing representations in terms of other, simpler representations, thereby enabling complex concepts to be built from simpler ones (Goodfellow et al., 2016). This hierarchical approach introduces the concept of depth to models, and thus the name deep learning.

Convolutional neural networks are a class of deep learning models that have been successfully applied to signal processing in the supervised setting, and have been shown to outperform expert-designed features (Janssens et al., 2016). In the supervised setting, we have a large, robust, labeled data set that we use to learn the parameters of the model. Convolutional neural networks consist of two primary parts: convolutional layers that learn a set of filters for feature extraction, and fully connected layers that make a prediction using the set of extracted features (LeCun & Bengio, 1995). Prediction error is propagated through the entire network, and as such, the representation learned by the filters is rooted in the prediction task.

The success of supervised methods is heavily dependent on the size and quality of the labeled data set. In many cases, producing an accurately labeled data set is not necessarily an inherent step in the data collection process. In gear health monitoring, gears are inspected periodically. The timing of these inspections is not necessarily ideal for interpolating the labels in between them, and the periods between inspections can vary. Furthermore, the inspections are judgmental, and thus inherently imperfect.

To overcome these limitations, we would like our representation learning algorithm to have as little reliance on the data labels as possible. Unsupervised learning methods involve no data labels. In this paper we explore the use of deep convolutional autoencoders, an unsupervised representation learning algorithm, in gear health monitoring.

3. DATA SET DESCRIPTION

The data used in this research spanned multiple tests on different sizes of spiral bevel gears. Tests were performed in the Spiral Bevel Gear Fatigue Test Rig at NASA Glenn Research Center. A detailed description of this test facility is provided in (Handschuch, 1995) and (Handschuch, 2001). Two sets of gears are driven against each other by their respective pinions, which act as a speed reducer for the left gear and a speed increaser for the right gear. The result is increased fatigue in the left gear. This process is measured with optical tachometers that are mounted on the left pinion shaft and the left gear shaft to produce a once-per-revolution tachometer pulse for the pinions and gears. Additional details on the vibration data collected during these tests can be found in (Dempsey, 2014).

The failure mode to be investigated was surface or contact

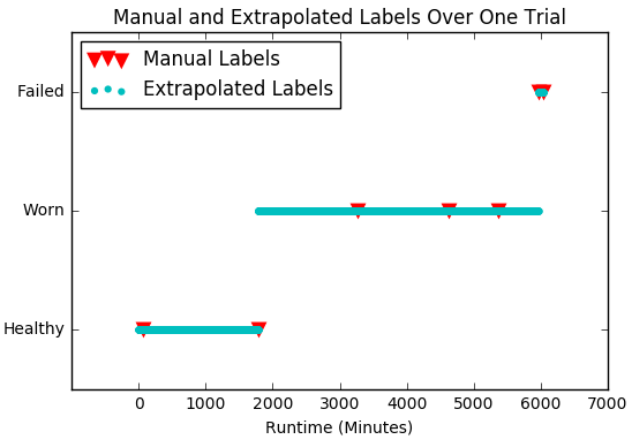

Figure 1. Plot of manual labels relative to interpolated labels.

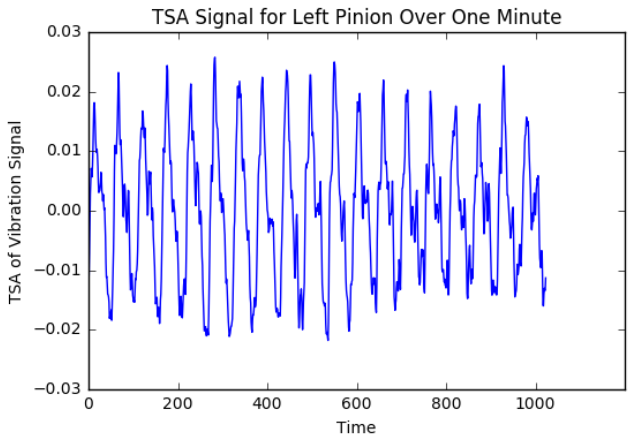

Figure 2. Plot of TSA signal.

fatigue that occurs when small pieces of material break off from the gear surfaces due to surface and subsurface stressors (Townsend, 1991). The failure mode for these tests are defined by American Gear Manufacturers Association (AGMA) standards. They are AMGA class (contact fatigue), general mode (macro pitting), and degree (progressive) in which pits are observed in different shapes and sizes greater than 0.04 in diameter (*Appearance of Gear Teeth - Terminology of Wear and Failure*, 2014). Gear sets were tested until progressive macropitting was observed on a significant area of two or more gear or pinion tooth surfaces.

The testing process is as follows. The test rig is loaded with gear sets and set to run. At intermittent periods researchers shut down the test rig and visually inspect the damage state. The decision to stop the process and inspect the gears is based on the researcher's judgement, and the interval between inspections varies. Gear sets are inspected multiple times over the course of a test, and when there is sufficient indication of failure, the experiment is ended.

Upon inspection the gears were labeled by the researcher as having experienced (1) no damage, (2) micropitting or edgewear, (3) 1 tooth macropitting, or (4) 2 or more teeth macropitting. These measurements were interpreted as discrete states, and to provide labels for periods outside of direct observation, labels were interpolated. Time between successive inspections varied. To minimize the effect of imprecise labeling, the labels were mapped to three damage states: healthy (1), worn (2 and 3), and failed (4). A plot of the manual and interpolated labels is shown in Figure 1.

The data set consists of vibration data drawn from four tests. Each test observed gears of different sizes being driven by a common gear size and lasted 1291, 2833, 6037, and 9578 minutes. These tests were parsed into 20 minute samples, and labeled with the damage state 10 minutes into the future. The collection of these samples defined the data set. The label was chosen to be 10 minutes into the future in part to account for the processing time between the event occurring, data acquisition, and generating a prediction. A more domain-specific reason for this choice was that rotorcraft operators need sufficient time to react to damage information. A plot of one-minute sample of the TSA signal is shown in Figure 2.

Vibration data were collected at sample rates that provided sufficient vibration data for calculating time-synchronous-averaged (TSA) data. TSA refers to techniques for averaging vibration signals over several revolutions of the shaft, in the time domain, to improve the signal-to-noise ratio (Martin, 1989). From the TSA data, several gear condition indicators were calculated for this analysis: figure of merit 4 (FM4), root mean square (RMS), sideband index (SI) and M8A (Martin, 1989). FM4, RMS and M8A are common time-domain, statistically based, vibration algorithms used in commercial HUMS (Stewart, 1977). The TSA and condition indicator calculations were made for both the left gear and left pinion.

There were two frequencies at which this data set was represented: the high frequency TSA data set at 1024 samples/minute with 2 features and the low frequency condition indicator data set at 1 sample/minute with 10 features. The high frequency data was used for extracting learned features at 1 sample/minute with 32 features, and the learned features and condition indicators were used to build models and conduct comparative analysis. Models were tested on a subset of the data, and trained on the rest. The testing subset contained the manually inspected points as well as additional points near inspection times to better balance the test set.

4. METHODOLOGY

In this section we present the methodology and motivations for deep convolutional autoencoders, and describe their application to gear health monitoring. Deep convolutional autoencoders act as powerful unsupervised learning tools for

mapping vibrations signals at high frequencies to low frequencies. The benefit of using convolutional layers instead of a sequence based model when working in the time domain is that the convolutional layers learn to extract features based on the occurence of a pattern within a sequence, with a low dependence on the patterns exact location therein.

4.1. Deep Convolutional Autoencoders

The autoencoder is a prime example of a representation learning algorithm. Autoencoders are comprised of two components: an encoder function that converts input data into a different representation, and a decoder function that converts the new representation back to its original form. Autoencoders are trained to preserve as much information as possible and to make the new representation have ideal properties (Vincent, Larochelle, Bengio, & Manzagol, 2008). Once the algorithm is trained, the encoder can be used to generate learned features from new data.

Traditional autoencoders use fully connected networks similar to multi-layer perceptrons as the encoder and decoder functions. Convolutional autoencoders instead use convolutional layers, an architecture found in convolutional neural networks, as the encoder and decoder functions (Geng et al., 2015).

Whereas multi-layer perceptrons use matrix multiplication for all of their layers, convolutional neural networks use the convolution operation for at least some of their layers (Goodfellow et al., 2016). The convolution operation provides a way to obtain a smoothed estimate at a point by using a weighted average that gives weight to different elements in a sequence based their location in the sequence.

In the continuous case, the convolution operation takes the form

$$s(t) = \int x(a)w(t-a)da,$$

and in the discrete case it takes the form

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a),$$

where $s(t)$ is the smoothed estimate, x is the function being smoothed, and w is the weighting function. In deep learning our goal is to learn the optimal weighting function.

In convolutional neural networks, the length of the sequence that is convolved over is typically set to be less than the length of the greater sequence. This convolution operation thus acts as a filter that is applied multiple times across the sequence. That is, the same filter is used across the entire sequence. Our goal is to optimize the effectiveness of this filter at distilling useful information by optimizing the weighting function w. Often times many filters are learned, and the what results is that at each convolutional layer the input sequence is mapped

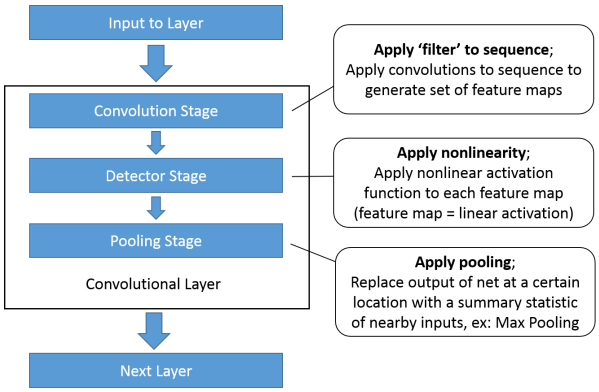

Figure 3. Diagram of convolutional layer.

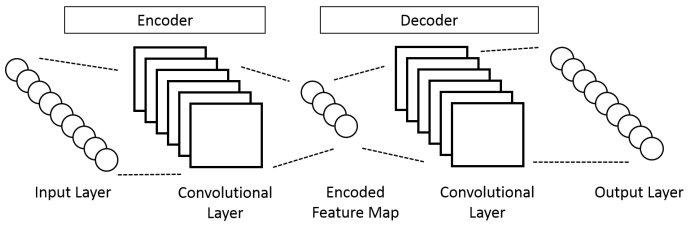

Figure 4. Diagram of convolutional autoencoder.

to multiple sequences that are the result of applying multiple smoothing functions to the original sequence. This process can be generalized to higher dimensions.

Convolutional layers consist of a convolution stage, detector stage, and pooling stage. The convolution stage performs several convolution operations on input data in parallel to produce a set of linear feature maps. The detector stage subsequently applies a nonlinear activation function to each feature map. Finally, the pooling layer reduces the size of the feature maps by replacing the output at a certain location with a summary statistic of nearby outputs (Goodfellow et al., 2016). Figure 3 is a diagram of a convolutional layer. For the decoder function, the pooling layer is replaced with an unsampling function.

Deep convolutional autoencoders define the encoder and decoder functions to consist of multiple convolutional layers (Geng et al., 2015). This gives the autoencoders depth and enables the aforementioned hierarchical approach to learning. Figure 4 depicts a single layer convolutional autoencoder. A deep convolutional layer would consist of more than one convolutional layer in between the input layer and the encoded feature map.

4.2. Motivations

The use of the convolution operator and pooling give convolutional neural networks and convolutional autoencoders a desireable structure for pattern recognition in time series. Con-

volution leverages a number of key ideas to improve machine learning systems: sparse interactions, parameter sharing, and equivariant representations (Goodfellow et al., 2016). Pooling makes the representation approximately invariant to small translations of the input (LeCun & Bengio, 1995).

Whereas traditional neural network layers use matrix multiplication and involve interaction between every input and output unit, convolutional networks typically have sparse interactions by making the size of the convolution operator smaller than the input. This means fewer parameters need to be stored, which improves statistical efficiency.

Because the kernel of the convolution is applied to every position of the input, convolutional networks can be said to have an element of parameter sharing. Sharing weights leads to decreased model complexity, because instead of learning a separate set of parameters for every location, the algorithm only learns one set. Parameter sharing leads to more efficient memory requirements and statistical efficiency than traditional neural networks.

The form of parameter sharing afforded by convolutions causes the layers to have a property called equivariance to translation. An equivariant function is one where changes to the input result in similar changes in to the output. In time-series data, this means that the convolution creates a timeline that identifies when different features appear in the input. If the event is moved later in time in the input, the same representation of it will appear in the output, it will simply occur later.

Pooling helps lead to invariance to translation, which means that if the input is translated by a small amount, the values of most pooled outputs will not change. Invariance to local translation is useful when the occurrence of a feature is more important than exactly where it occurs.

4.3. Application to Gear Health Data

The goal of applying these ideas to gear health monitoring is to find sufficient representations from high frequency vibration data to perform model building at a lower frequency. Using a deep convolutional autoencoder aligns with these goals, and affords many ideal properties for the task at hand.

The high frequency data set contains 20 minute sequences with 1024 samples/minute for both the gear and pinion. The goal is to find a representation at 1 sample/minute. The depth of the model allows for features to be learned with a dependence on information at different frequencies. The convolutions operate along the time axis, and in combination with pooling and the depth of the model, allow for features to be built not only from local information at higher frequencies, but also from sequence-level information at lower frequencies. That is, because of successive down-sampling in the pooling stages, the learning algorithm observes the data at multiple, successively lower frequencies. By convolving exclusively across the time axis, the gear and pinion are treated separately.

The properties of equivariance and invariance to translation lower the dependence of the learned representation on the sequence parsing method. The sequences are arbitrarily parsed out of the data set with respect to the information represented by a sequence, similar to how samples might be drawn in real-world systems. Because of equivariance and invariance to translation this is not a significant burden. The model will be more focused on the existence of features or groupings of features rather than the location within the sequence that they occur.

Using an autoencoder approach to train the weights, as opposed to training the weights using a convolutional neural network, makes deep learning a viable tool given the data constraints. The main difference between these two approaches is that the convolutional neural network learns a representation rooted in the prediction task, whereas the encoder learns a representation more akin to an identity mapping. While the representation steeped in the prediction task would theoretically be more ideal, it requires more accurately labeled data to train and tune.

It is important to note the need for more accurately labeled data rather than just more data to make convolutional neural networks viable. A major problem faced in the experimentation done for this paper was that the data was not originally collected for the purposes of machine learning. As a result, manually observed damage states were measured aperiodically. The interpolated labels between these observations have inherent inaccuracies, and there are likely occasional sequences with similar features but contradictory labels. This is not an uncommon issue when applying machine learning to gear health monitoring, and health monitoring in general, due to the original purpose of data collection and the cost of labeling. These occasional contradictions from mislabelings are often irreconcilable in convolutional neural networks trained on small data sets. The network can suffer from divergence or the inability to learn. Autoencoders, however, are unsupervised, which means that they are trained without knowledge of input labels. This makes autoencoders particularly useful when data and labeling constraints exist.

The final model architecture used is as follows. The encoder condensed the input representation from 20480 samples x 2 features (gear and pinion) to 1 sample x n features. This condenses 20 minutes of data into one sample. The number of learned features n was tuned for the prediction task, with the goal of minimizing redundant features. The final encoder used condensed the representation to 32 features. The encoder consisted of 3 convolutional layers and a fully connected layer, where the pooling function used was max pool-

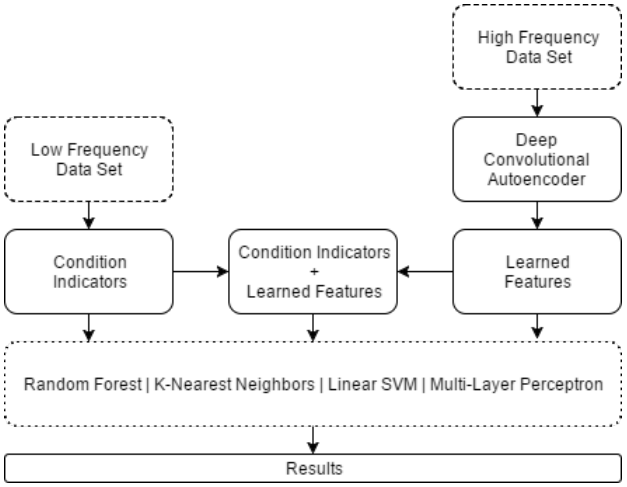

Figure 5. Diagram of testing process.

Table 1. Accuracy score for each feature set.

Accuracy Score				
Data	RF	KNN	LSVM	MLP
Autoencoder	.618	.324	.556	.294
Condition Indicator	.706	.618	.794	.765
Both	.765	.618	.794	**.853**

ing, and the input shrank by similar factors in each convolutional layer.

The deep learning model was built and tested using the Keras library (Chollet et al., 2015).

5. RESULTS

The general workflow of the testing process is shown in Figure 5. The condition indicators are drawn from the low frequency data set. The learned features are generated by passing the high frequency data set through the deep convolutional autoencoder. The TSA time signal is the high frequency data. This input iss 20 minutes worth of TSA data, which equates to 20 concatenated sequences of 1024 TSA time signals. These sets of features are combined, and the condition indicators, the learned features, and the combined set are used to train random forest, k-nearest neighbors, linear support vector machine, and multi-layer perceptron machine learning models. These models are tested and their results are output for analysis.

The results from testing on autoencoder generated data, condition indicator data, and a combined data set is shown in Table 1 and Table 2. RF, KNN, LSVM, and MLP correspond to the random forest, k-nearest-neighbors, linear support vector machine, and multi-layer perceptron respectively. The AUC-ROC is the area under the receiver operating characterisic and acts as a heuristic measure of informativeness. The top performing model is the multi-layer perceptron trained on the

Table 2. AUC-ROC for each feature set.

AUC-ROC				
Data	RF	KNN	LSVM	MLP
Autoencoder	.730	.619	.637	.722
Condition Indicator	.915	.8545	.868	.859
Both	.889	.854	.868	**.987**

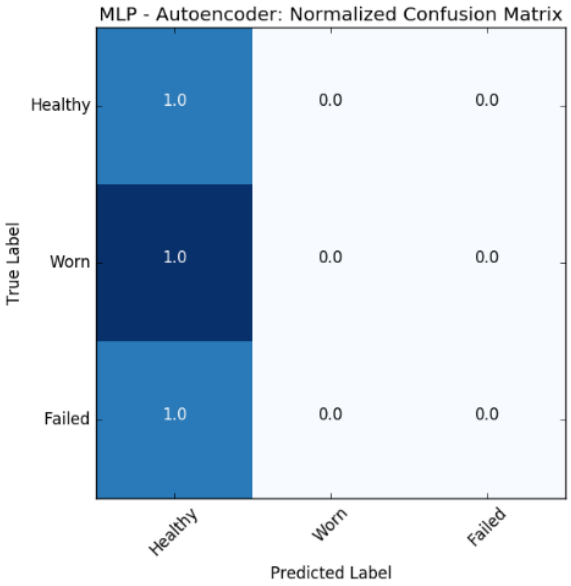

Figure 6. The confusion matrix for the MLP on autoencoder features.

combined data set with a score of .853 and an AUC-ROC of .987 which are bolded in their respective tables. The AUC-ROC was calculated using a one-against-all approach.

The confusion matrices of the multi-layer perceptron on the autoencoder, condition indicator, and combined data are shown in Figure 6, Figure 7, and Figure 8 respectively. The multi-layer perceptron failed to learn given only autoencoder data as evidenced by the confusion matrix. However, the extra information supplied by the autoencoder data improved the accuracy over the model trained exclusively on the condition indicators.

5.1. Discussion of Results

The autoencoder data was unable to train models to outperform condition indicator data. Autoencoders learn a function for encoding the data such that it can be decoded. As a result, the encoder is trained to be an approximate identity mapping. Identity mappings can be poor features for predictive models. The encoder learns to discard information unnecessary for compression taskhowever, the discarded data may be relevant for the prediction task. This is likely compounded by the fact that the autoencoder data condensed the input data from 20480 samples to 32 features. This was done in an ef-

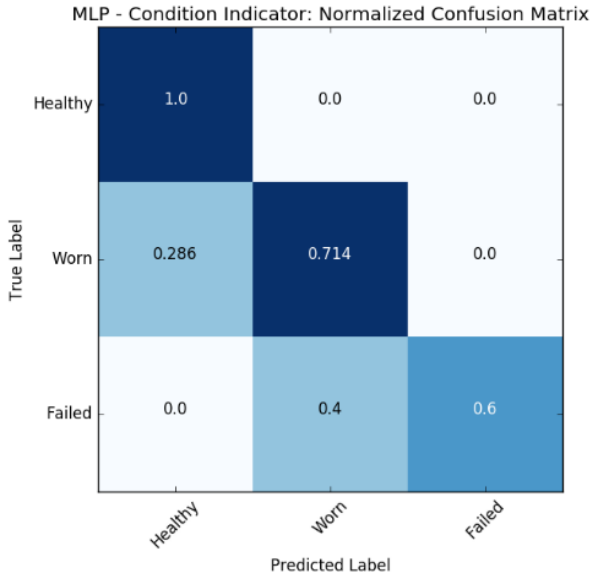

Figure 7. The confusion matrix for the MLP on condition indicators.

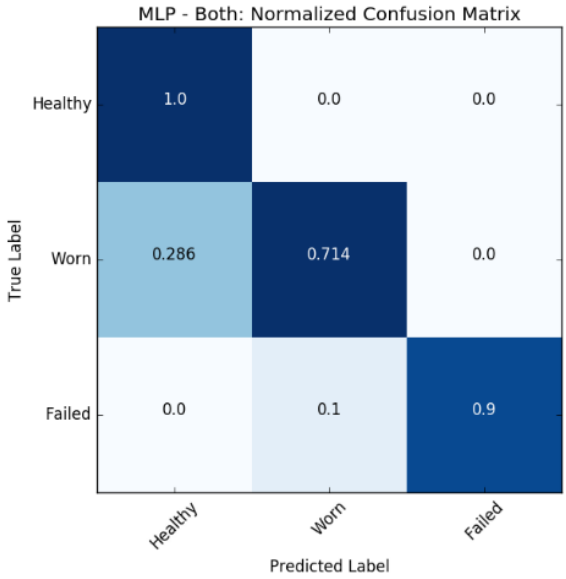

Figure 8. The confusion matrix for the MLP on both sets of features.

fort to align with the condition indicator data set for testing purposes.

The autoencoder data, however, did show the ability to represent the input data. In particular, some of the models trained exclusively on autoencoder data could better differentiate the intermediate worn damage state than models trained exclusively on condition indicator data. However, the autoencoder models struggled to differentiate between healthy and failed states. The models trained on both autoencoder and condition indicator data seemed to leverage these trends. In particular, the top performing model, the multi-layer perceptron, exemplifies this. The MLP trained on both data sets outperformed the MLP trained on exclusively condition indicators in identifying worn test samples. The other machine learning algorithms were unable to harness this benefit of the autoencoder data.

The MLP may have outperformed the other models because it is considered a deep learning algorithm. That is, the MLP is a representation learning algorithm, unlike the others it was tested against. The MLP may be able to better make use of the autoencoder features because the training process continues to learn features, while the other algorithms construct a model and do not learn new features.

Taking a closer look at the confusion matrices for the MLP, one can see that the worn-failed classification improved for the combined set while the healthy-worn classification remained unchanged. This result is dependent on both the of training the autoencoder and the mapping function it learns as well as the fact that the data was manually labeled, among other things. That is, there is bias introduced in labeling and in the mapping function and speculating the root cause of this health-worn versus worn-failed improvement is difficult. It may be that the healthy-worn misclassification comes from data that is so similar that the additional features do not increase performance. This similarity is possible because of the process used to label the data.

6. CONCLUSION

Deep convolutional autoencoders demonstrated that deep learning can successfully be used to make predictions on low frequency scales using high frequency data in gear health monitoring. Although the learned representation was outperformed by the condition indicators, the learned features bolstered the performance of the condition indicator data. The top performing model was a deep neural network that learned from both autoencoder and condition indicator feature sets. This model using the combined feature set outperformed all other model and feature set pairs. It had an accuracy of 0.853, offering a 0.059 lift over the next best pair, and it had a near perfect AUC-ROC of 0.987, offering a 0.072 lift over the next best pair. This shows promise for the method, and for the use of deep learning in gear health monitoring. The autoencoder feature set seemed to represent information not expressed by the condition indicators. However, the autoencoder feature set was useless without the condition indicators, whereas the condition indicators were able to train models with respectable performances without the autoencoder features.

The size of the data set and the labels were severe constraints on the performance of the deep learning models. The autoencoder was able to leverage deep learning to benefit the modeling of gear fatigue, however, the representation learned

by the autoencoder was not routed in the prediction task, and ultimately needed to be combined with condition indicator data to be used for classification. The driving reason behind using an autoencoder was that learning an identity mapping did not require accurate labels. Other methods like convolutional neural networks learned very little and converged very quickly, or they diverged. While this could be due to the training set size, the inability to lower training error suggests that contradictory labels played a role as well. This could have also been exacerbated by inconsistent gear size across the training sets.

7. FUTURE WORK

Deep learning is a viable approach in gear health monitoring that can be leveraged to bolster existing techniques. There are two main paths for further investigation: the data and the algorithms.

The TSA time signal was used to build the convolutional autoencoder because the same time signal was used to calculate the condition indicators. However, future work could include exploring different techniques for processing the raw signal data. We exclusively used vibration data; future studies could draw on additional sources of data such as oil debris sensors.

Convolutional layers have many properties ideal for pattern extraction from high frequency data. Unsupervised representation learning methods help to address the inherent difficulty in labeling gear data. Unsupervised deep learning methods are scarce, however, there are several ways that supervised methods can be tweaked to make training more feasible and deserve further investigation.

Autoencoders can be used to train the weights for the convolutional layers of a convolutional neural network (Masci, Meier, Cireşan, & Schmidhuber, 2011). Then, the convolutional layers can be either locked or minimally tuned when training the fully connected layers. The idea would be to train the parameters of the convolutional layers sufficiently enough to learn a useful representation of the data using an autoencoder approach. These weights would then be loaded into a convolutional neural network with a fully connected layer that has randomized weights. The subsequent training can control the evolution of the weights in the convolutional layers separate from those of the fully connected layers by fixing them, by tuning them in only a limited number of passes through the data set, or by decreasing the learning rate so that they do not update as quickly as the fully connected layers.

Similarly, randomized weights have shown significant performance ability in tasks related to computer vision, where model complexity can be controlled by tuning select layers of a deep network, while other layers are locked (Saxe et al., 2011). The general idea is that randomized or minimally trained weights in early layers of the network may be sufficient for basic feature extraction. If so, the overall complexity of the model can be reduced, thus offering a direct benefit for constrained training set sizes.

Transfer learning is used in deep learning to tune pre-trained networks for a specific task (Pan & Yang, 2010). General models could be trained on larger data sets or simulated data and then tuned for specific problems. In computer vision, well-established models can be downloaded, and their weights can be tuned for a new task related to image recognition. To apply this concept to signal processing, one could pre-train a convolutional neural network to make predictions on a well-established data set, and then tune the network for making predictions over a different data set. Another option might be to first train on simulated data, and then tune the network on real-world data. By testing performance on simulated data, one might be able to study the impact of different factors on the effectiveness, training time, and architecture of the model. For example, one could explore the noise tolerance of a particular network architecture by varying the amount of artificial noise in a signal. Naturally, the success of using simulated data would likely be bounded by the ability to ground the simulation in reality.

REFERENCES

Appearance of gear teeth - terminology of wear and failure (No. ANSI/AGMA 1010-E95). (2014). AGMA Nomenclature Committee, American Gear Manufacturers Association.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chollet, F., et al. (2015). *Keras.*

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167).

Dempsey, P. J. (2014). *Investigation of spiral bevel gear indicator validation via ac-29-2c using test rig damage progression test data* (Tech. Rep. No. TM-2014-218384). NASA.

Geng, J., Fan, J., Wang, H., Ma, X., Li, B., & Chen, F. (2015). High-resolution sar image classification via deep convolutional autoencoders. *IEEE Geoscience and Remote Sensing Letters*, *12*(11), 2351–2355.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Handschuch, R. F. (1995). *Thermal behavior of spiral bevel gears* (Tech. Rep. No. TM-106518). NASA.

Handschuch, R. F. (2001). *Testing of face-milled spiral bevel gears at high-speed and load* (Tech. Rep. No. TM-2001-210743). NASA.

Janssens, O., Slavkovikj, V., Vervisch, B., Stockman, K., Loccufier, M., Verstockt, S., ... Van Hoecke, S. (2016). Convolutional neural network based fault detection for rotating machinery. *Journal of Sound and Vibration*, *377*, 331–345.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10), 1995.

Martin, H. R. (1989). Statistical moment analysis as a means of surface damage detection. In *Proceedings of the 7th international modal analysis conference.*

Masci, J., Meier, U., Cireşan, D., & Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011*, 52–59.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Saxe, A., Koh, P. W., Chen, Z., Bhand, M., Suresh, B., & Ng, A. Y. (2011). On random weights and unsupervised feature learning. In *Proceedings of the 28th international conference on machine learning (icml-11)* (pp. 1089–1096).

Stewart, R. M. (1977). *Some useful data analysis techniques for gearbox diagnostics* (Tech. Rep. No. Report MHM/R/10/77). Machine Health Monitoring Group, Institute of Sound and Vibration Research, University of Southampton.

Townsend, D. P. (Ed.). (1991). *Dudley's gear handbook*. McGraw-Hill.

Vevcevr, P., Kreidl, M., & Smid, R. (2005). Condition indicators for gearbox condition monitoring systems. *Acta Polytechnica*, *45*(6).

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on machine learning* (pp. 1096–1103).