A Study of Deep Neural Networks Transfer Learning For Fault Diagnosis Applications

Michael Franco-Garcia[1], Nithya Nalluri[2], Alex Benasutti[*], Larry Pearlstein[3] and Mohammed Alabsi[4]

[1,2,3,4] *The College of New Jersey, 2000 Pennington Rd, Ewing Township, NJ 08618, USA*

garcim20@tcnj.edu
nallurn1@tcnj.edu
pearlstl@tcnj.edu
alabsim@tcnj.edu
**benasua1@tcnj.edu*

ABSTRACT

Intelligent fault diagnosis utilizing deep learning algorithms has been widely investigated recently. Although previous results demonstrated excellent performance, features learned by Deep Convolutional Neural Networks (DCNN) are part of a large black box. Consequently, lack of understanding of underlying physical meanings embedded within the features can lead to poor performance when applied to different but related datasets i.e. transfer learning applications. This study will investigate the transfer learning performance of a Deep Convolutional Neural Network (DCNN) considering 4 different operating conditions. Utilizing the Lou & Loparo (2004) Case Western Reserve University (CWRU) bearing dataset, the DCNN will be trained to classify 12 classes. Each class represents a unique different fault scenario with varying severity i.e. inner race fault of 0.007", 0.014" diameter. Initially, zero load data will be utilized for model training and the model will be tuned until testing accuracy above 99% is obtained. The model performance will be evaluated by feeding vibration data collected when the load is varied to 1, 2 and 3 HP. Initial results indicated that the classification accuracy will degrade substantially. To improve the network generalization capabilities, this paper proposes the addition of white Gaussian noise to the raw vibration data. Results indicate that a very high level of additive noise can improve the transfer learning accuracy. The discussion will then focus on the influence of changing loads on fault characteristics, network classification mechanism, and activation strength in addition to the visualization of convolution kernels in time and frequency domains.

1. INTRODUCTION

Since the time of the first industrial revolution, special attention has been placed on the safety and reliable operation of rotating equipment. About 40% of machinery failures are due to bearings degradation and damages Frosini, Harlişca, & Szabó (2014). Hence, timely and efficient diagnosis of bearings faults is required to ensure continuous and sustainable industrial operation. Traditionally, fault diagnosis falls under two categories: physics-based and data-driven. Physics- based approaches predict the machine health based mathematical models. Generally, those models are not able to update the parameters to adapt the real-time stream of measured data Weiss, Khoshgoftaar & Wang (2016). On the contrary, data-driven approaches are designed to extract meaningful patterns from machinery data. Conventional data-driven methods depend on the design of handcrafted features which are fed into shallow machine learning models for classification i.e. logistic regression Yan & Lee (2005) and support vector machine Widodo & Yang (2007). The introduction of the fourth industrial revolution has revolutionized data-driven machine health monitoring techniques. In specific, deep learning methods showed high potential in intelligent fault diagnosis and machinery big data analytics. Traditional handcrafted features extraction is effectively replaced by end-to-end feature extraction performed by DCNN. However, one of the main challenges that faces DCNN model development is the high variability of fault characteristics and operating conditions in industrial environments. For this, DCNN models need to have generalization ability to account for varying operating conditions and fault progressions scenarios. This problem is usually addressed in the context of transfer learning Weiss, Khoshgoftaar & Wang (2016). Transfer learning for fault diagnosis applications can be categorized under three types: domain adaptation, parameter transfer and feature transfer.

Domain adaptation aims to leverage a limited amount of unlabeled data under different operating conditions and

improve the DCNN model generalization capability. This method aims to transfer results achieved at source domain with labeled data under given operating conditions, to a target domain with different operating conditions and unlabeled data Wang, Michau, & Fink (2019, May). Inspired by computer vision and natural language processing, several fault diagnosis papers applied domain adaptation methods to improve DCNN performance on new operating conditions Zhang, B., Li, W., Hao, Li, X. L., & Zhang, M. (2018) and Zhang, W., Peng, Li, Chen, & Zhang, Z. (2017). Evaluation of related literature indicates that domain adaptation research for fault diagnosis applications requires careful choice of network structure, data preprocessing, and training strategy Wang, et. al. (2019).

Parameter transfer learning methods adjust the DNN model to adapt to the changing operating conditions. This is accomplished by training the base model with source domain data, and updating some parameters to accommodate the changes in target domain Li, Hu, Li, M., & Zheng (2020). This method was applied by modifying the last NN softmax layer and keeping the previous layers unchanged Zhang, Tao, Wu, & Guan (2017). Another approach utilized pre-trained CNN instead of random initialized networks. The CNN extracts relevant features from the source domain and the last fully-connected layers are modified to accommodate the changes in the target domain Cao, Zhang, & Tang (2018) and Shao, McAleer, Yan, & Baldi (2018).

Feature based transfer builds a transfer function that links the source and target domains. The transfer function is designed to extract common representations from both domains which are not sensitive to changing operating conditions. Wang, Xie, Zhang L., & Duan (2016, August) applied the feature based transfer by projecting the raw data into low-dimensional space in order to minimize cross domain differences. Wen, Gao, & Li (2017) developed a three-layer sparse auto-encoder network to extract key features from raw data. The maximum mean discrepancy was used to penalize changes during joint feature learning.

2. PROPOSED APPROACH

Our approach for classification of vibration data uses a deep convolutional neural network for processing time-domain data sampled at 12 KHz. We propose a network architecture based on the work of Zhang, et. al. (2017), but with 12 outputs from the final softmax layer, rather than 10. Most notably we explore the use of additive noise as a form of regularization and examine its effect on the network's ability to generalize its learning to correctly classify faults under operating conditions that differ significantly from those related to training. This strategy of naive transfer via aggressive regularization may be well suited to real-world engineering problems where one may encounter a continuum of operating conditions and other variations on an ongoing basis.

We assigned the 12 fault classes following the work of Alabsi, Liao, & Nabulsi (2021). We trained our network on data captured under no-load conditions (0hp), and investigated transfer of learning via generalization, to cases where the machine was loaded at 1 HP, 2 HP and 3HP. Our work used the CWRU vibration data, described by Lou et. al. (2004).

Our models were trained and evaluated using the TensorFlow framework. We trained the neural networks by minimizing the cross-entropy loss of the softmax output values. The optimizer used, TensorFlow's 'NAdam' procedure, combines Nestorov momentum and an Adam-like approach for learning-rate adaptation.

In order to understand how our networks were able to perform classification, we explored the results using Fast Fourier Transforms (FFTs) of vibration data and kernel maps as well as t-Distributed Stochastic Neighbor Embeddings (t-SNEs) of the first layer filter output powers. Vibration data under no-load conditions were fed to the model for training. For comparison we also trained the model architecture from scratch under all load conditions.

3. PROCEDURE

3.1 Dataset Description

CWRU data obtained via 12 KHz sampling at the drive end (DE) were used. Samples at no-load were used, as well as those with loads of 1, 2, and 3 HP. Twelve cases were studied for classification, which are described in Table 1.

Table 1: Definition of 12 fault cases studied

Fault Type - Inner Race (IR) Outer Race (OR) Ball (B)	Fault Size & Position	Class Number
no fault	0	1
B	0.007	2
B	0.014	3
B	0.021	4
IR	0.007	5
IR	0.021	6
IR	0.028	7
OR	0.007@3	8
OR	0.007@6	9
OR	0.014@6	10
OR	0.021@3	11
OR	0.021@6	12

3.2. Data Preparation

The CWRU dataset includes sampled vibration data acquired at 12 KHz at the drive end, under 16 different fault conditions. The CWRU dataset consists of MATLAB data files, where each data file represents one or more arrays, with each file corresponding to a different combination of fault class and load power. We used only the drive-end data array from each file, and we focused on the 12 fault conditions listed in Table 1, following the work of Alabsi, Liao, & Nabulsi (2021). The dataset included different component failures and failures of the same class but with varying severity levels.

Each data array contained a different number of samples, so we trimmed all of the arrays used for experimentation to match the shortest array, which contained $M = 120{,}801$ samples. We handled the data in a manner similar to Zhang, et. al. (2017) by following the steps illustrated in Fig. 1, and outlined below.

1. After trimming to M samples, each array was partitioned into a training/validation section and a testing section.
2. Training/validation data was obtained by sampling overlapping segments of data, with each segment consisting of 2048 samples. Consecutive segments overlapped each other with a ratio of 95%. A total of 660 segments of training/validation data were thus obtained.
3. Data for testing was obtained by partitioning the testing section into 25 non-overlapping segments of 2048 samples each.
4. Where Additive White Gaussian Noise (AWGN) was used for data augmentation and regularization, each of the 660 segments was included in the training/validation set twice -- once without added noise, and once with AWGN applied.

We looked into applying min/max normalization, but this was found to dramatically hamper classification performance, so this step was abandoned.

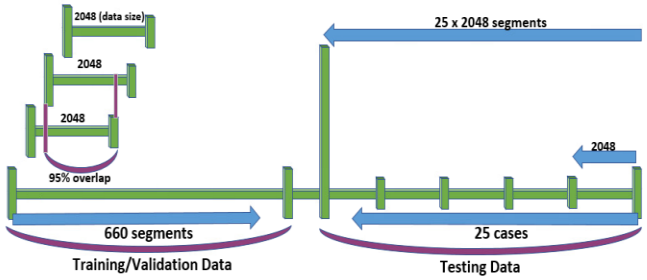

Fig. 1: Illustration of dataset partitioning and segment creation

3.3. Network Architecture and Optimizer

Our experiments used a deep convolutional neural network architecture adapted from that presented in Zhang, et. al. (2017), but with 12 softmax outputs, corresponding to the 12 classes defined in Table 1. The network architecture is shown in Fig. 2.

Following the work of Zhang, et. al. (2017) we used batch normalization (indicated by the blocks labeled *BN* in Fig. 2) during training to improve training performance. During inferencing, normalization was done based on a first order recursive lowpass filter of statistics gathered during training (with Keras filter parameter *momentum*=0.99).

A cross-entropy loss function was used as the target metric for optimization, and the entire DCNN model was represented using the TensorFlow 2 (TF2) framework with Keras. The TF 'Nadam' optimizer was used to minimize the loss function, which combines Nestorov momentum (which typically speeds adaptation) with Adam (which adaptively, but monotonically, dampens the adaptation rate over time). This differs from some of the studies cited above, as most use a simple 'Adam' optimizer. Although NAdam might be expected to converge faster than Adam, in practice we did not find conclusive evidence for this assumption.

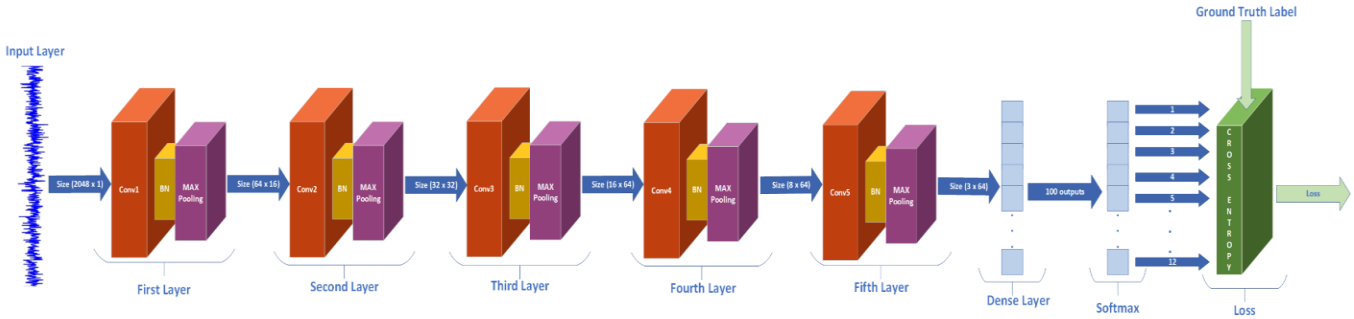

Fig. 2: Graphical representation of the deep convolutional neural network architecture studied

3.4. Test Accuracy and Confusion Matrices

The twelve softmax outputs from our model were collapsed into a single hard decision from the set $\{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{12}\}$ by the arg-max function. The model was tested under 25 non-overlapping segments held back for testing, extracted from signals in each of the twelve classes. An overall accuracy number was assigned based on the fraction of the resulting $\mathbf{25 \times 12 = 300}$ test segments that were decided correctly by the model. In some cases we attained test accuracies of 100%. In tests where there were errors we used confusion matrices to help understand the nature of the errors.

Confusion matrices are used to summarize the performance of machine learning classifiers. Predicted values fall under four categories: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN). In this paper, each confusion matrix has the ground-truth labels arrayed at the bottom of the matrix, and the labels predicted by the model arrayed down the left side. Numbers in the matrix represent how many of the test segments in each ground-truth class were assigned to each hypothetical class label. Classification accuracy for each class is calculated using the equation below:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

The overall network classification accuracy is calculated by averaging the accuracy for individual classes.

3.5. Convolution Layer #1 Visualization

A convolutional neural network contains a combination of convolutional layers and fully connected (dense) layers. It is noteworthy that a digital convolution is a kind of implementation of a linear time-invariant operator -- a processing element that can be understood as a frequency selective filter. As such, it can be thought of as imparting a frequency-dependent gain, and a frequency-dependent phase shift. In the time domain a discrete-time convolution can be represented as:

$$y(n) = \sum_{k=0}^{N-1} h(k)\, x(n-k) \tag{2}$$

and in the frequency domain as:

$$\left|Y(e^{j\omega})\right| = \left|H(e^{j\omega})\right|\left|X(e^{j\omega})\right| \tag{3}$$

where: $x(\cdot), y(\cdot), h(\cdot)$are the input stream, output stream and N-point filter kernel, respectively, and $X(e^{j\omega}), Y(e^{j\omega}), H(e^{j\omega})$are the respective discrete-time Fourier transforms. Thus $|H(e^{j\omega})|$is the gain response, and (2) describes how a convolution accomplishes selective weighting of different frequencies.

Note that a cascade arrangement of convolutions is mathematically equivalent to a single convolution. Referring back to Fig. 2, the block labeled 'Conv1' represents a set of 16 different filters, each with an order, $N = 64$. Although the first five layers of the DCNN shown in Fig. 2 are all convolutional, the non-linear activations, and non-linear pooling operators, prevent mathematically collapsing cascades from input to output into simple filters. The outputs of convolutional layers and activations can be thought of as *feature maps*, a term used in the context of DCNNs for computer vision. Thus Conv2 applies filters to the feature maps produced by Conv 1, and so forth. While the filtering operations performed by Conv2 through Conv5 are hard to interpret, we can, and do, investigate the filters created by the Conv1 layer. For the Conv1 layer it can be more instructive to analyze the input signals and trained kernels in the frequency domain than in the time domain.

Considering how much information regarding the machine's status seems to be contained in the frequencies and modes of vibration, we hypothesized that the vector of output powers of the sixteen trained filters of Conv1 might be sufficient for effectively clustering and isolating the desired classes. We investigated this by implementing the processing graph illustrated in Fig. 3, which uses t-SNE dimensionality reduction to enable visualization of clusters.

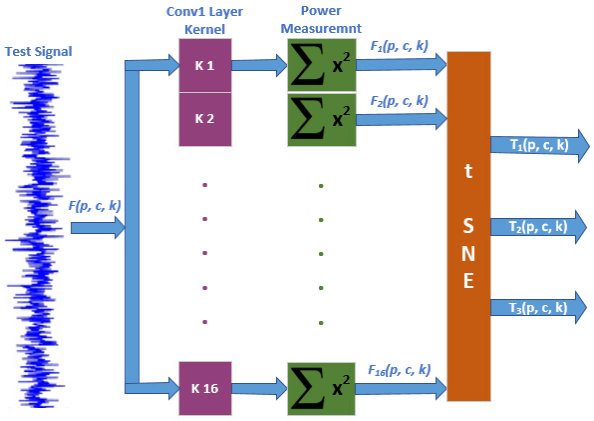

Fig. 3: Block diagram of the process used for t-SNE analysis of the effects of Convolution Layer #1 kernels. Here $f(p, c, k)$ is a length-2048 vector of vibration samples, where p represents the load power ($p = 0, 1, 2, 3\ HP$), c is the ground truth class ($1 \le c \le 12$)k is the test segment index ($1 \le k \le 25$). The t-SNE algorithm was used to generate a reduction in dimensionality to either two or three (as shown as $T_{\{1,2,3\}}(p, c, k)$) values.

While t-SNE is highly effective for performing dimensionality reduction for human interpretation, the approach is not repeatable, and it does not preserve scale factors. As a result, we also used the PCA method for reducing dimensionality in cases where it was valuable to retain the ability to directly compare different experiments.

When one trains a DCNN the thousands of model weights are typically initialized based on pseudo-random numbers. In general, different *seed* values for the pseudo-random number generator result in very different DCNNs. Even two DCNNs

with the exact same input-to-output behavior may have internal differences due to permutations of kernels and weights within hidden layers. We attempt to compare Conv1 from different DCNNs trained under different conditions. While it is impossible to define an unambiguous correspondence between kernels in DCNNs trained under different conditions (or even under the same condition, but with different seeds), we define a *mean frequency* for each kernel, which allows sorting kernels for comparison. The mean frequency is represented as:

$$\bar{H}_r = \frac{f_s}{N}\sum_{m=0}^{N/2} m\,\tilde{H}_r(m) \quad (4)$$

where $\tilde{H}_r(m)$is the normalized magnitude response of the DFT of kernel r:

$$\tilde{H}_r(m) = \frac{|H_r(m)|}{\sum_{j=0}^{N/2}|H_r(j)|}, \quad (5)$$

$|H_r(m)|$is the m^{th}coefficient of the discrete fourier transform (DFT) of the weights of convolution kernel r, and f_s is the sampling rate, 12 KHz.

4. RESULTS

4.1 Classification Task Accuracy

Our experiments primarily focused on the task of training models under generic (no-load) conditions, and exploring the ability of these models to transfer that learning to the task of classification under different load conditions of 1, 2, and 3 HP. When training we used either segments from the Case Western Reserve University dataset directly, or those segments augmented with zero-mean AWGN with standard deviations in the set, $\boldsymbol{S}$ ={0.01, 0.03, 0.10, 0.30, 1.0, 3.0, 10.0}. We were interested in each model's accuracy at a load of 0 HP (the load at which it was trained), and its average accuracy including load 0 HP, and transfer to loads of 1, 2, and 3 HP. Since each fault class exhibited a different amount of signal power, the signal to noise ratio of the noisy-half of the training data, depends on the class, as shown in Table 2.

We trained the same network architecture 10 times, each with a different seed for the pseudo-random generated used to initialize the weights. The results are illustrated by scattering the data in Fig. 4, and as a summary in Fig. 5. Examining the figures we observe a significant spread of average accuracies for each amount of additive noise, but also some significant trends. It appears that the addition of small amounts of noise have relatively little effect, and that noise in the range of about 0 dB,$i.e.\sigma = 0.3$, can have a deleterious effect. But the peak and average transfer learning improves as the noise level increases substantially, and appears to be maximized at about $\sigma = 3.0$.

Table 2: Signal-to-noise ratios (SNRs) for each fault case's data, as a function of AWGN standard deviation. Units are dB.

Std Dev	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
0.01	17	23	24	23	29	34	38	38	37	20	30	35
0.03	8	13	14	13	20	25	29	28	27	11	20	26
0.1	-3	3	4	3	9	14	18	18	17	0	10	15
0.3	-12	-7	-6	-7	0	5	9	8	7	-9	0	6
1.0	-23	-17	-16	-17	-11	-6	-2	-2	-3	-20	-10	-5
3.0	-32	-27	-26	-27	-20	-15	-11	-12	-13	-29	-20	-14
10	-43	-37	-36	-37	-31	-26	-22	-22	-23	-40	-30	-25

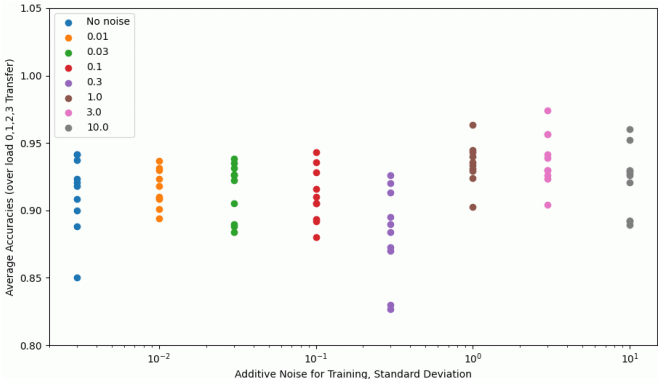

Fig. 4: Average Accuracy vs. AWGN Std. Dev. as scatter plot

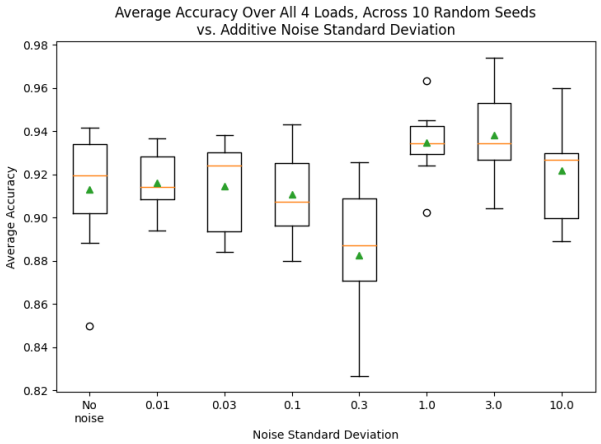

Fig. 5: Average Accuracy vs. AWGN Std. Dev. - data from ten training runs per noise level are summarized using box-and-whiskers

Based on the results obtained, we investigated four cases in detail:

Case 1: Train based on 0 HP (no load) data only, with no additive noise. We selected the result with the lowest 'Test Loss'.

Case 2: Train based on 0 HP (no load) data only, with additive noise standard deviation, $\sigma = 0.3$. We selected the result with the maximum average accuracy for transfer learning.

Case 3: Train based on data at loads 0,1,2,3 HP, with no additive noise. We selected the result with maximum accuracy.

Case 4: Train based on data at loads 0,1,2,3 HP, with additive noise standard deviation, $\sigma = 0.3$. We selected the result with the maximum accuracy.

All cases were trained considering the classes described in Table 1. The classification accuracies of the four cases described above are presented in Table 3. Here it is clear that the basic network architecture produces relatively poor transfer learning - with an average accuracy of 85%. Case 2, based on a high level of AWGN, is of particular interest, as it represents excellent transfer learning. In fact, although a direct comparison is not possible, the transfer learning accuracies that we obtained through the use of high-powered additive noise appear to be superior to those presented in Zhang, et. al. (2017). We sought to understand the mechanism that produced this desirable characteristic.

Table 3: Test accuracy vs. load and noise standard deviation used for data augmentation.

Train on ...	Case# / AWGN σ	Load (HP)	Accuracy
0 HP load data only	1 no noise	0	100.00%
		1	84.67%
		2	81.67%
		3	73.67%
		Average Loads 0-3	**85.00%**
	2 noise$\sigma = 3.0$	0	96.67%
		1	97.34%
		2	98.00%
		3	97.67%
		Average Loads 0-3	**97.42%**
All loads: 0, 1, 2, 3 HP	3 no noise	0	99.67%
		1	100.00%
		2	100.00%
		3	100.00%
		Average Loads 0-3	**99.92%**
	4 noise$\sigma = 3.0$	0	100.00%
		1	99.67%
		2	100.00%
		3	100.00%
		Average Loads 0-3	**99.92%**

We focused our attention on understanding the operation of the first convolutional layer -- this layer operates on the raw data samples, and it consists of 16 filters, where each filter is composed of 64 coefficients. The fairly long filters allow this layer to attain high-frequency selectivity. For illustration, the kernels obtained for Case 1 are shown in Fig. 6. It is difficult to discern the mechanism of operation from direct examination of the kernels, but we gained insight by visualizing the FFTs of the kernels. In order to effectively compare the filters obtained from the 4 cases of interest, we computed the mean frequency of each filter, and sorted them according to increasing kernel mean frequency. The mean frequencies for each of the 4 cases are shown in Fig. 7. Looking at Fig. 8 we note that many of the kernels are bandpass filters, or narrow two-band filters. This suggests that the most important information for classification is contained in the frequency band between 2000 Hz and 3300 Hz. This is somewhat surprising, as the fundamental forcing functions produced by the various faults produce frequency components below 1000 Hz.

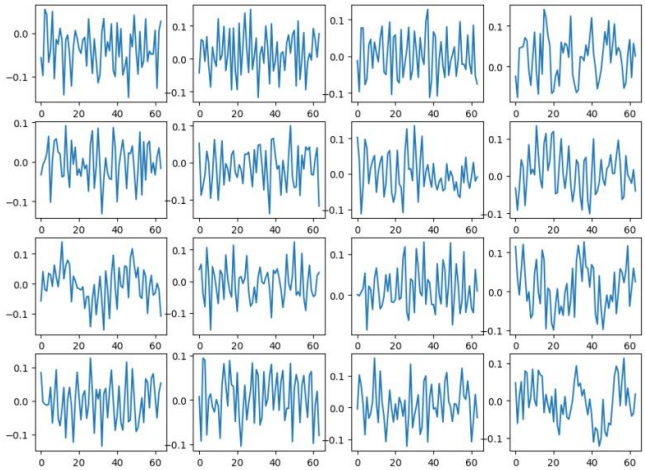

Fig. 6: Time domain plots of all 64 values in the first convolutional layer (from Case 1).

4.2. Visualization Results

The frequency-domain analysis of the sampled vibration signals is shown in Fig. 9. Examining the figure we note that the no-fault case produces a significant component at 1100 Hz, and, evidently, a harmonic at 2200 Hz. The cases with ball faults produced low vibration amplitudes, and only one narrow component at 1400 Hz, and a band of power distributed between 2500-3500 Hz. The inner-race and outer-race faults produced significant narrowband components distributed throughout the bands up to 3500 Hz. Considering that each fault class produces a distinct signature, we used t-SNE to investigate how well Conv1 is able to cluster incoming data according to fault class, and how those clusters relate to clusters of data corresponding to different loading conditions.

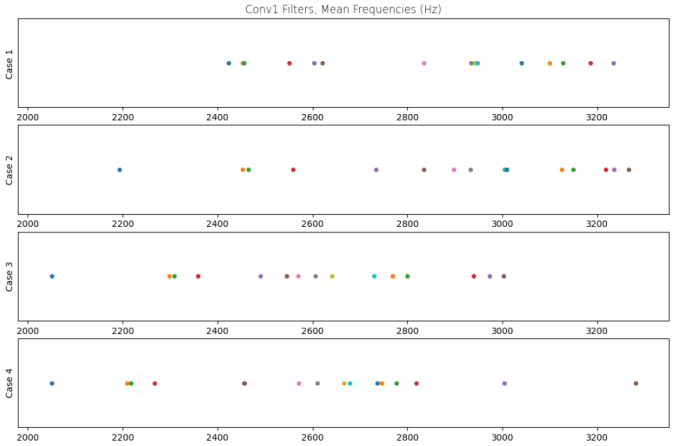

Fig. 7: Comparing all four experiments by scaling each of their first convolutional layer kernel weights with frequency, and taking the mean of the weights at each kernel. There are a total of 16 kernels for the first convolutional layer.

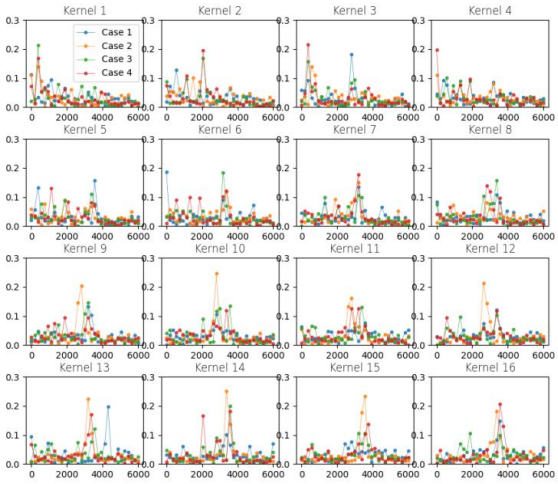

Fig. 8: Kernel FFTs, only showing the first half of the signal (32 values out of 64), sorted by mean frequency from lowest to highest.

Fig. 10 represents a reduction to two dimensions of the 16-point vector of filter signal powers produced at the output of Conv1, using t-SNE. In this plot, each hue/marker style represents a unique fault class. Two cases were run for each fault class -- one at zero load, and one at the maximum, 3 HP, load. The DCNN was trained based on the no-load data, which is represented by the darker shade of the same hue. The data at 3 HP is used to study the most extreme case of transfer learning, and is represented by the lighter shade. We see that most of the classes are nicely clustered by the Conv1 filter output power. The only classes where the clustering is inconclusive is between Classes 2, 3 and 4, and Classes 9 and 12. Of course, our DCNN has a great deal of additional resources available to further cluster the data -- with 4 convolutional layers that produce increasing numbers of feature maps.

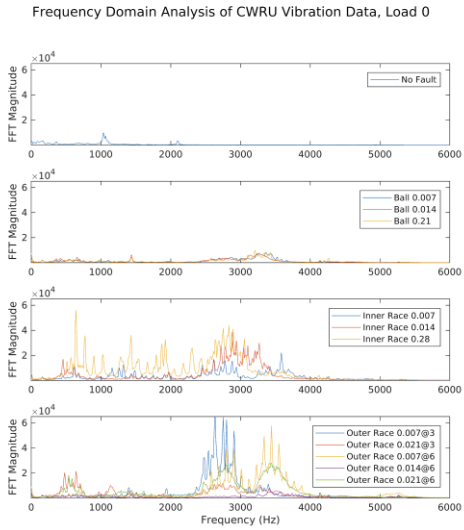

Fig. 9: Frequency domain analysis of sampled vibration data signals at no load (0 HP)

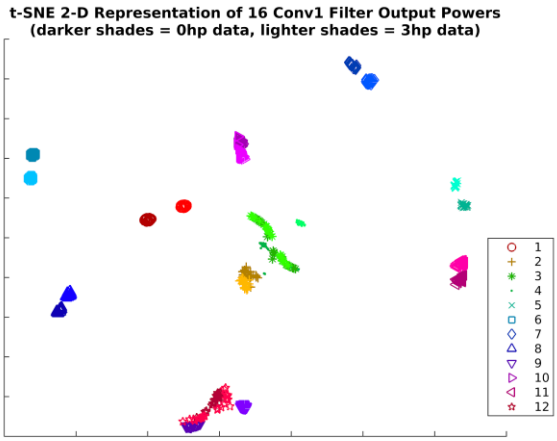

Fig. 10: 2-D representation of the Conv1 kernels' output power using t-SNE.

A detailed look into the structure of the filter output data for Classes 2, 3, and 4, is shown in Fig. 11. Here, PCA is used to reduce the data to two dimensions. Again, we see fairly easy separability of the classes, with only a small number of input data vectors appearing to present a challenge.

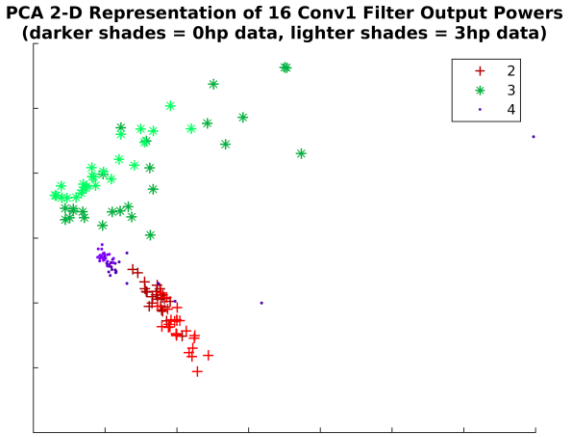

Fig. 11: Detailed view of 2-D representation of Conv1 kernels' output power using PCA.

4.3. Transfer Classification Results

Details on the types of misclassifications by the DCNN from Case 2 can be found in the confusion matrices shown in Fig. 12. The confusion matrix has the ground-truth labels arrayed at the bottom of the matrix, and the labels predicted by the model arrayed down the left side. Numbers in the matrix represent how many of the test segments in each ground-truth class were assigned to each hypothetical class label. As expected we primarily find errors relating to confusion between Classes 2, 3, and 4 (all ball faults, of different size). We also find confusion between Classes 2 and 3, and Class 10 (outer race fault at 6 o'clock).

5. CONCLUSION

We demonstrated how very high levels of additive noise can be used advantageously to train DCNNs with significant improvements in transfer learning. We showed that reducing the output of each Conv1 filter kernel to a single power measurement produces a well-clustered statistic, suggesting the possibility that convolutional layers 2-5 may actually be of little value for the classification task. We found that DCNNs trained on the CWRU dataset tend to extract narrowband features, and are especially concentrated on high frequencies. Finally, we developed a technique to compute the mean-frequencies of convolution kernels, and used this metric as the basis for sorting the kernels from different training scenarios, thus allowing direct comparisons.

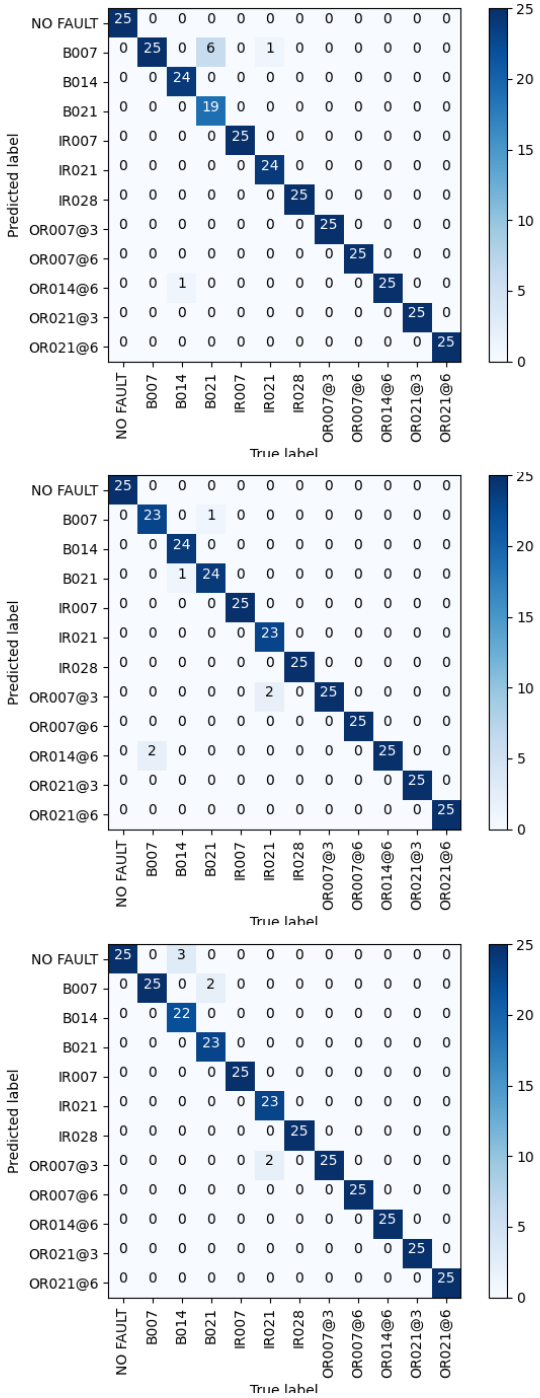

Fig. 12: Confusion matrices for Case 2. (top) Transfer learning to 1 HP at 97.34% test accuracy. (center) Transfer learning to 2 HP at 98.00% test accuracy. (bottom) Transfer learning to 3 HP at 97.67% test accuracy.

REFERENCES

Alabsi, M., Liao, Y., & Nabulsi, A. A. (2021). Bearing fault diagnosis using deep learning techniques coupled with handcrafted feature extraction: A comparative study. *Journal of Vibration and Control*, *27*(3-4), 404-414.

Cao, P., Zhang, S., & Tang, J. (2018). Preprocessing-free gear fault diagnosis using small datasets with deep convolutional neural network-based transfer learning. *Ieee Access*, *6*, 26241-26253. http://dx.doi.org/10.1109/ACCESS.2018.2837621.

Frosini, L., Harlişca, C., & Szabó, L. (2014). Induction machine bearing fault detection by means of statistical processing of the stray flux measurement. *IEEE Transactions on Industrial Electronics*, *62*(3), 1846-1854.

Li, X., Hu, Y., Li, M., & Zheng, J. (2020). Fault diagnostics between different type of components: A transfer learning approach. *Applied Soft Computing*, *86*, 105950.

Lou, X., & Loparo, K. A. (2004). Bearing fault diagnosis based on wavelet transform and fuzzy inference. *Mechanical systems and signal processing*, *18*(5), 1077-1095.

Shao, S., McAleer, S., Yan, R., & Baldi, P. (2018). Highly accurate machine fault diagnosis using deep transfer learning. *IEEE Transactions on Industrial Informatics*, *15*(4), 2446-2455. http://dx.doi.org/10.1109/TII.2018.2864759.

Wang, J., Li, S., Han, B., An, Z., Xin, Y., Qian, W., & Wu, Q. (2018). Construction of a batch-normalized autoencoder network and its application in mechanical intelligent fault diagnosis. *Measurement Science and Technology*, *30*(1), 015106.

Wang, J., Xie, J., Zhang, L., & Duan, L. (2016, August). A factor analysis based transfer learning method for gearbox diagnosis under various operating conditions. In *2016 International Symposium on Flexible Automation (ISFA)* (pp. 81-86). IEEE. http://dx.doi.org/10.1109/ISFA.2016.7790140

Wang, Q., Michau, G., & Fink, O. (2019, May). Domain adaptive transfer learning for fault diagnosis. In *2019 Prognostics and System Health Management Conference (PHM-Paris)* (pp. 279-285). IEEE.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, *3*(1), 1-40. https://doi.org/10.1186/s40537-016-0043-6

Wen, L., Gao, L., & Li, X. (2017). A new deep transfer learning based on sparse auto-encoder for fault diagnosis. *IEEE Transactions on systems, man, and cybernetics: systems*, *49*(1), 136-144. http://dx.doi.org/10.1109/TSMC.2017.2754287.

Widodo, A., & Yang, B. S. (2007). Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical systems and signal processing*, *21*(6), 2560-2574.

Yan, J., & Lee, J. (2005). Degradation assessment and fault modes classification using logistic regression. J. Manuf. Sci. Eng. 127 912–4

Zhang, B., Li, W., Hao, J., Li, X. L., & Zhang, M. (2018). Adversarial adaptive 1-D convolutional neural networks for bearing fault diagnosis under varying working condition. *arXiv preprint arXiv:1805.00778*.

Zhang, R., Tao, H., Wu, L., & Guan, Y. (2017). Transfer learning with neural networks for bearing fault diagnosis in changing working conditions. *IEEE Access*, *5*, 14347-14357. https://dx.doi.org/10.1109/ACCESS.2017.2720965.

Zhang, W., Peng, G., Li, C., Chen, Y., & Zhang, Z. (2017). A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals. *Sensors*, *17*(2), 425.