IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 February 2023, accepted 13 March 2023, date of publication 16 March 2023, date of current version 22 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3258399

METHODS

Design and Analysis of Convolutional Neural Layers: A Signal Processing Perspective

MOHAMMED M. FARAG, (Member, IEEE)
Electrical Engineering Department, College of Engineering, King Faisal University, Al-Ahsa 31982, Saudi Arabia
Electrical Engineering Department, Faculty of Engineering, Alexandria University, Alexandria 21544, Egypt

e-mail: mfarag@kfu.edu.sa

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia, under Project GRANT2677.

ABSTRACT Convolutional layers (CLs) are ubiquitous in contemporary deep neural network (DNN) models, commonly used for automatic feature extraction. A CL performs cross-correlation between the input to the layer and a set of learnable kernels to produce the layer output. Typically, kernel weights are randomly initialized and automatically learned during model training using the backpropagation and gradient descent algorithms to minimize a specific loss function. Modern DNN models comprise deep hierarchical stacks of CLs and pooling layers. Despite their prevalence, CLs are perceived as a magical tool for feature extraction without solid interpretations of their underlying working principles. In this work, we advance a method for designing and analyzing CLs by providing novel signal processing interpretations of the CL by exploiting the correlation and equivalent convolution functions of the layer. The proposed interpretations enable the employment of CLs to develop finite impulse response (FIR) filters, matched filters (MFs), short-time Fourier transform (STFT), discrete-time Fourier transform (DTFT), and continuous wavelet transform (CWT) algorithms. The main idea is to pre-assign the CL kernel weights to implement a specific convolution- or correlation-based DSP algorithm. Such an approach enables building self-contained DNN models in which CLs are utilized for various preprocessing and feature extractions tasks, enhancing the model portability, and cutting down the preprocessing computational cost. The proposed DSP interpretations provide an effective means to analyze and explain the operation of automatically trained CLs in the time and frequency domains by reversing the design procedures. The presented interpretations are mathematically established and experimentally validated with a comprehensive machinery fault diagnosis application example illustrating the potential of the proposed methodology.

INDEX TERMS Machine learning, signal processing, convolutional layer, interpretable neural networks, machinery fault diagnosis.

I. INTRODUCTION

Convolutional layers (CLs) are the backbone of modern deep neural network (DNN) models. Convolutional neuron foundations date back to the 1950s [1], and the modern framework of convolutional neural networks (CNNs) has been established since 1989 in LeNet-5 [2]. CLs have two prominent features compared to the fully connected dense layers: first, the small kernel size imposed by the limited receptive field of a CL, which significantly reduces the layer computational complexity; and second, the automatic feature extraction ability arising from the correlation operation performed by the CL. These features have promoted using CNNs in plentiful DNN architectures and applications [3].

Typically, CLs are instantiated and hierarchically stacked in a DNN with a set of hyper-parameters, including the number of filters, kernel size, stride length, and dilation rate, which are progressively tuned during model training to minimize the loss function assigned for a specific task. Kernel weights of CLs are automatically learned during model

The associate editor coordinating the review of this manuscript and approving it for publication was Kaustubh Raosaheb Patil.

training using the gradient descent and backpropagation algorithms, where each kernel learns a specific feature from the training data. Such an approach proved its effectiveness for feature extraction, minimizing the need for expert knowledge, and optimizing the model performance in numerous applications [1]. However, using CLs does not entirely limit the need for preprocessing and feature extraction procedures.

Feature engineering is the stage in which important features of the dataset are extracted and prepared for the machine learning (ML) task. A feature is any information collected from data that can be used to accomplish the ML task. Despite the automatic feature extraction capabilities of CNNs, some tasks and applications still require data preprocessing to extract representative features of the dataset rather than feeding raw data to the CNN. Data preprocessing in ML refers to the techniques of transforming or encoding raw data into a suitable format that is more effectively processed by an ML model It encompasses several techniques including data cleaning, integration, transformation, and reduction [4]. Data cleaning is concerned with filling in missing values and labels, noise reduction, and outlier removal. Data integration merges data present in multiple sources into a single source. Data transformation includes steps for transforming data into suitable forms for ML such as normalization, domain transformation, data segmentation and aggregation, data reshaping, and dimensionality reduction. Typically, preprocessing tasks are implemented as a pipeline preceding the ML model and applied to both training and testing data. Many data preprocessing tasks and transformations are implemented using signal processing techniques such as noise reduction, domain transformation, and dimensionality reduction, to name a few.

Despite its significant importance, data preprocessing reduces the portability of ML models and incurs extra computational overhead for ML tasks, limiting their applicability to resource-constrained platforms [5], [6]. This work presents a methodology to implement several signal processing-enabled preprocessing tasks inside DNN models using CLs rather than separate preceding pipelines. Such an approach would reduce the preprocessing computational cost and enable the building of self-contained DNNs that take full advantage of the model computational resources and accelerators during model execution at inference time. Frequency-selective and noise-removal filters are examples of the preprocessing stages that can be implemented using the proposed method. Domain transformations such as time-to-frequency transformations, which are fundamental signal processing transformations applied to extract spectral features, are another example of potential applications of the proposed method.

The fundamental operation performed by a CL is computing the cross-correlation between the layer input signal and the kernel weights assigned to the layer to produce the layer output. Mathematically, the cross-correlation between two signals is equivalent to the convolution between a signal and the time reversal of the other. Linear correlation and convolution are fundamental operations in signal processing that constitute the foundation of several concepts and applications. Both operations are key enablers for designing interpretable DNN models and analyzing their operation from the signal processing perspective.

In this work, we advance correlation and convolution interpretations of the CL and exploit them to implement a set of digital signal processing (DSP) algorithms inside the DNN model, which can alleviate the need for the corresponding preprocessing pipelines. The main idea is to design a convolution- or correlation-based DSP filter that performs a specific task and pre-assign the filter taps to the CL kernel weights instead of relying on the backpropagation and gradient descent algorithms to learn layer weights. Specifically, we exploit the correlation operation performed by CLs to implement matched filters (MFs) and continuous wavelet transform (CWT) and exploit the equivalent convolution operation to implement finite impulse response (FIR) filters, short-time Fourier transform (STFT), and discrete-time Fourier transform (DTFT). Linear inactivated CLs are employed to build the corresponding linear DSP algorithms. Generally, the proposed approach can be applied to implement other correlation- or convolution-based DSP algorithms. This study is confined to 1D convolutional (Conv1D) layers and time series signals; nevertheless, the presented approach is extensible to higher dimensional CLs. 1D CNNs have significantly lower computational complexity than their 2D counterparts, enabling the design of CLs with relatively larger kernel sizes, which serves the purpose of this study.

Such an approach enables building self-contained DNN models which have several advantages compared to using preprocessing pipelines or internal lambda layers (the lambda layer is used to build custom functions inside a DNN model) to implement DSP-based preprocessing steps, including:

- This approach alleviates the need for separate preprocessing and feature extraction stages which enhances the ML model portability and reduces the pre-processing computational cost.
- Self-contained DNN models built using standard CLs are fully compatible with existing model optimization and quantization techniques applied for model deployment and execution on edge devices, special-purpose accelerators, or specific instruction set architectures. This is in contrast to models that have separate preprocessing pipelines or internal lambda layers. Current deep learning development frameworks such as Tensorflow and Pytorch can only optimize a limited set of standard layers and activation functions and still have limitations for optimizing models with lambda layers [7], [8]
- Self-contained DNN models take full advantage of the model computational resources and accelerators during model execution at inference time unlike models with separate preprocessing pipelines.
- Initializing a CL with weights designed to perform a specific DSP task while enabling layer training can

simultaneously guide the model learning experience and enable fine-tuning of the algorithm parameters to optimize the model performance on the training data.
- Pre-assigning layer weights reduces the number of model learnable parameters and, consequently, shortens the model training time.

On the other hand, the presented interpretations of the CL offer a means for analyzing automatically trained DNN models in both the time and frequency domains. At the level of time domain analysis, the convolutional kernels can be interpreted as MFs, with kernel weights serving as the MF template. Such an interpretation provides an informed knowledge of the temporal features extracted by the CL kernels. At the level of frequency domain analysis, the convolutional kernels can be interpreted as FIR filters, where the filter impulse response is equivalent to the time-reversed kernel. The frequency response of a CL can be computed from the corresponding FIR impulse response and leveraged to provide a comprehensive vision of the spectral features extracted by a CL.

Contributions of this work include:

- Advancing DSP interpretations of CLs established using mathematical foundations and supported with implementations and visualizations,
- Proposing a method that enables the development of self-contained DNN models in which specific signal processing-based preprocessing tasks are implemented inside the model, eliminating the need for the corresponding preprocessing pipelines,
- Providing a means to analyze and explain the operation of automatically trained DNN models.
- Presenting a comprehensive application example of machinery fault diagnosis to demonstrate the design and analysis capabilities of the proposed method.

The remainder of this work is organized as follows: Section II provides a brief background about the significance and applications of DSP in ML and related work. Section III advances the mathematical foundations of the DSP interpretations of CLs and the design and analysis methodology of CLs according to the proposed interpretations. Section IV presents a comprehensive application example from the machinery fault diagnosis literature to illustrate the potential and usage mechanisms of the proposed methodology for designing self-contained DNNs and analyzing automatically trained CNNs. Conclusions and future work are portrayed in Section V. Appendixes provide the implementation details and visualization results of the DSP-based CLs using Python and TensorFlow.

II. RELATED WORK

DSP and ML share many common foundations and mutual links that can be exploited to push the development of highly efficient, pervasive tools of broad applicability in various domains [9]. Numerous DSP tools and algorithms, such as filters, Fourier Transform, and CWT, play an essential role in data reprocessing and feature extraction for a wide range of ML models in various applications [10], [11], [12], [13], [14]. Time-frequency representations of 1D signals are widely used in DNN models for electrocardiogram (ECG) classification [15], [16], machinery fault diagnosis [17], [18], [19], and audio feature extraction [10], [20], to name a few. Discrete wavelet transform (DWT) is pervasively utilized for noise removal and feature extraction in ML models [21], [22], [23].

On the other hand, various ML methods have been recently applied to many traditional and extended signal processing areas [24], [25], [26]. Applications include, but are not limited to, audio processing and speech recognition [27], [28], [29], image and video processing [26], [30], language modeling [31], natural language processing [32], [33], [34], biomedical signal processing [35], radar signal processing [36], and communication system design and modeling [37], [38]. Common to this research is the application of data-centric ML techniques to tackle classical signal processing problems, including classification, detection, forecasting, and regression.

Despite their immense success in ML, DNN models are often utilized as ''black box'' models since their underlying operation and decision mechanisms are poorly understood by developers. The inability to comprehend DNN models inhibits their use in mission-critical applications and restricts their full potential [39]. The close link between signal processing and ML can be exploited to provide solid interpretations of DNN models. In this study, we aim to advance DSP interpretations of the CL and use these interpretations to develop explainable DNN models with improved performance and reduced computational resources.

Srinivasamurthy [40] presented a frequency domain interpretation of 1D CNN filters, investigated the dropout regularization impact, and proposed guidelines for setting the model hyperparameters. Jia et al. [41] presented a frequency domain interpretation of 2D CNNs. The effect of controlling the proportion of different frequency filters in the first layer on network classification accuracy and model robustness is explored. Stankovic and Mandic [42] utilized the MF theory as a mathematical tool to explain the CNN operation, which revealed a direct link between CLs and MFs in finding data patterns or features. Such an approach reveals the data flow in CNN learning and elucidates a strategy for optimal parameter selection.

Some works provide interpretations and implementations of DSP algorithms using full DNN and CNN architectures [43], [44]. In our previous works, we presented MF- and STFT-based CNN classifiers for ECG classification and human activity recognition (HAR) [45], [46], [47]. The proposed models achieved remarkable classification and real-time performance results compared to state-of-the-art rivals at a much lower computational cost. This work advances CL interpretations and implementations of additional DSP algorithms and provides a comprehensive example of using the proposed method to design and analyze

DNNs in a machinery fault diagnostic application. To the best of our knowledge, this is the first work to address using a single CL to implement inclusive DSP algorithms inside DNN models. The proposed method alleviates the need for preprocessing and feature engineering steps, provides beneficial tools for designing and analyzing DNN modes, and opens new frontiers in ML architectures and applications.

Potential applications of the proposed method include health monitoring [46], [47], machine-human interface [48], activity recognition [45], machinery fault diagnosis [17], [18], [19], audio signal processing [10], [20], [27], radar signal processing [36], automatic modulation recognition [49], [50], wireless sensor networks and internet of things [51], [52], and communication systems [37], [38], to name a few. In these applications, data is collected as a time series and fed to an ML model to perform a specific task such as classification, regression, or forecasting. Data preprocessing pipelines, including noise reduction and domain transformations, are applied to extract representative features for the ML task. The DSP-based CLs advanced in this study can replace the corresponding preprocessing pipelines and create self-contained DNN models. Data denoising can be implemented using the CL-based FIR filter presented in this work. The frequency spectral and time-frequency features of time series data typically computed in separate preprocessing pipelines can be implemented inside the DNN model using the DTFT, STFT, and CWT CLs proposed in this work.

III. 1D CONVOLUTIONAL LAYER INTERPRETATION AND IMPLEMENTATIONS

The discrete convolution operation $*$ between two sequences $h[n]$ and $x[n]$ is defined by 1 whereas the discrete correlation operation $\star$ is defined by 2 [53]. Convolution is equivalent to correlation with the time-reversed sequence defined by 3.

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=-\infty}^{\infty} x[n-k]h[k] \tag{1}$$

$$x[n] \star h[n] = \sum_{k=-\infty}^{\infty} x[k]h[k-n] = \sum_{k=-\infty}^{\infty} x[k-n]h[k] \tag{2}$$

$$x[n] \star h[n] = x[n] * h[-n] \tag{3}$$

Sequential data are converted to parallel segments and fed to the Conv1D layer to perform concurrent correlation with layer kernels and produce the output segments. Temporal data segmentation is equivalent to rectangular windowing in DSP [53]. Other windowing functions can be readily adopted by pointwise multiplying the input segment by a window function of the same segment width.

A Conv1D layer comprises several 1D filters or kernels that extract various feature maps from an input signal. By cross-correlating layer kernels with an input sequence, a Conv1D layer generates a 1D feature map per kernel. The number of strides parameter governs the correlation shift amount, whereas the dilation rate controls the distance between correlation points. A fine-tuning bias parameter is used to improve the correlation results. With the stride length and dilation rate set to 1, the output of a standard Conv1D layer is defined as follows:

$$y_{c_{out}}[n] = f_a(b_{c_{out}} + \sum_{i=1}^{C_{in}} w^i_{c_{out}}[n] \star x_i[n]) \tag{4}$$

where $y[n]$ is the kernel output, b is the bias term, $w[n]$ is the kernel weight, $x[n]$ is the kernel input, c_{out} denotes the output channel, and C_{in} denotes the total number of input channels, and $f_a()$ is the activation function. Unless otherwise stated, the bias term b will be set to zero for all DSP implementations presented in this work, and CLs with linear activation (an inactivated kernel) will be considered throughout this work.

The last equation describes the operation of a typical multi-input, multi-output 1D CL in which an output channel is computed as the sum of the cross-correlation between input channels and channel-specific kernels. There are other CL variants, such as separable and depth-wise CLs, in which each input channel is correlated with a different kernel, independent of other channels. For a standard CL with a single channel input, the CL output is computed as the cross-correlation between the input channel and multiple CL kernels, resulting in distinct output channels or feature maps. This study will focus on the standard single-input, multi-output CL. Figure 1 illustrates a block diagram of the Conv1D layer with N_F filters each of N_K kernel width. For a typical Conv1D layer with a single stride shift and dilation rate, an input signal segment of N_S-samples is fed to the layer and correlated with N_F filters to produce N_F output segments each of length N_S.

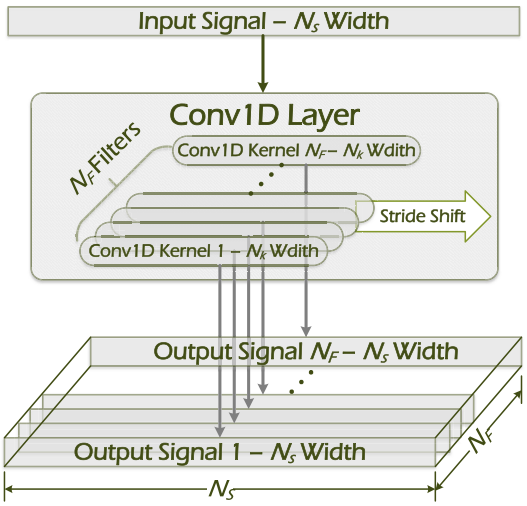


FIGURE 1. Block diagram of the 1D CL (Conv1D).

A prominent feature of CLs is their automatic feature extraction capability. In automatically trained CNNs, CL kernel parameters are automatically learned during the model training phase using the gradient descent and

backpropagation algorithms to minimize the loss function and optimize the model performance in a specific task. Kernels learned during model training represent salient features manifested by the training examples and captured by the CL correlation process. However, despite being the most common approach, automatic training of DNN models conceals various CL potentials that can significantly improve the model performance and cut down on its computational cost. In this study, we seek to unveil some of these potentials, inspired by the close link between CLs and signal processing.

A. 1D CONVOLUTIONAL LAYER AND MATCHED FILTER

The first interpretation of CLs is based on the correlation operation. Correlation is the main operation performed by MFs. An MF is an optimal linear filter for detecting signals contaminated by additive white Gaussian noise (AWGN) by maximizing the signal-to-noise ratio [54]. MFs are extensively deployed in wireless communication receivers for optimal signal detection. The MF receiver block diagram is illustrated in Figure 2. An MF correlates the noisy input signal with a template of the signal to be detected, samples the maximum correlator output, and decides whether the signal matches the template via thresholding. The MF correlator output is defined as:

$$y[n] = x[n] \star h[n] = \sum_{i=1}^{N} x[i]h[i-n] \tag{5}$$

where $y[n]$ is the correlator output, $x[n]$ is the input signal, and $h[n]$ is the template signal.

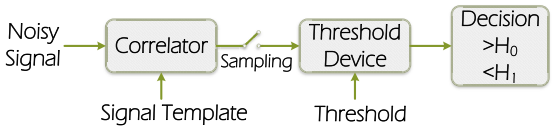


FIGURE 2. Block diagram of the matched filter receiver.

Comparing 3 and 5 demonstrates that a Conv1D kernel can implement the MF correlation operation with the template signal assigned to the kernel weights. Signals are sequentially fed to the MF correlator but concurrently fed to the CL kernel as segments. The correlation shifting operation is carried out using the CL stride parameter. The MF sampling operation is implemented using a global max pooling (GMP) layer to sample the Conv1D kernel maximum output. The thresholding operation is implemented using a nonlinear activation function such as Relu. The Conv1D layer comprises multiple kernels, as shown in Figure 1, which can be used to build multiple MFs for parallel feature detection. This stack of Conv1D, nonlinear activation, and pooling layers work together as a typical MF with multiple templates $h_i[n] = w_i[n]$, where i represent the i^{th} kernel of the Conv1D layer. The total number of MF CL parameters is $N_F * N_K$.

The MF interpretation of CLs can be used to build optimal ML classifiers with minimum computational requirements. A single Conv1D layer with a large receptive field and a GMP layer are required to build an optimally resource-efficient classifier model. The MF CL kernels can be pre-assigned if the template signals are already known or automatically learned during model training to extract featured patterns from the training data. A fully connected dense layer is then inserted and trained to map MF outputs to relevant classifier outputs by employing the weighted sum functionality of the dense layer. The MF-based CNN has been presented in our previous works with applications to ECG classification and HAR. The proposed models achieve remarkable classification and real-time performance compared to the state-of-the-art methods while minimizing the computational cost [45], [47]. The achieved results qualify model usage for edge inference on resource-constrained microcontroller devices without sacrificing the model performance, which overcomes concerns of cloud inference such as availability, privacy, and connectivity.

The MF interpretation of CLs is not only beneficial for building MF CNN classifiers, but it is also handy for analyzing and visualizing automatically trained DNN CLs. A trained model kernel weights $w_i[n]$ can be extracted from CLs in the model and plotted against time to visualize the MF templates extracted by the CL and link these templates to dataset features. Such an approach would help develop a deeper understanding of the CL operation, especially in applications emphasizing time domain analysis. Such an approach for visualizing convolutional kernels and activation maps has been adopted in the computer vision and machine learning literature for 2D CNNs since 2013 [55, Chapter 5].

B. 1D CONVOLUTIONAL LAYER AND FIR FILTERS

This interpretation of CLs is based on the convolution operation. An FIR filter is a DSP filter with a finite impulse response [53]. Outputs of a causal FIR filter are the weighted sum of the most recent inputs, as depicted in Figure 3. The output of an FIR filter of order N is defined as:

$$y[n] = h[n] * x[n] = \sum_{i=0}^{N} x[i]\, h[n-i] \tag{6}$$

where $x[n]$ is the input signal, $y[n]$ is the output signal, and N is the filter order; an N^{th}-order filter has $N+1$ taps, $h[n]$ is the FIR filter impulse response, and $*$ is the 1D convolution operator. For a direct-form FIR filter, $h[n]$ is the filter coefficients or taps.

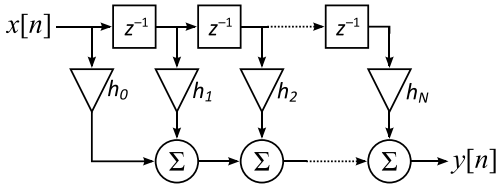


FIGURE 3. A direct form discrete-time FIR filter of order N. Each unit delay is a z^{-1} operator in $\mathcal{Z}$-transform notation.

Comparing 4 and 6 reveals that an inactivated Conv1D kernel with $b = 0$ is equivalent to an FIR filter, where the kernel input is convolved with the kernel time-reversed version. The stride shift of a Conv1D kernel performs the time delay of an FIR filter. Unlike the FIR filter with a sequential input, a Conv1D kernel is fed with an input segment and applies parallel convolution to produce the output segment. Accordingly, a Conv1D kernel is equivalent to the impulse response of an FIR filter, where the kernel weights are the coefficients of the FIR impulse response. In other words, the Conv1D kernel is a direct form FIR filter with $h[n] = w_i[-n]$ applied to a 1D input signal $x[n]$. A Conv1D kernel of length N_K has an order of $N = N_K - 1$ as an FIR filter.

The Fourier transform of the convolution between two signals is equivalent to the pointwise product of their Fourier transforms [53]. Accordingly, the FIR filter output $y[n]$ for an input sequence $x[n]$ is defined in the frequency domain as:

$$\underbrace{\mathcal{F}(x * h)}_{Y(\omega)} = \underbrace{\mathcal{F}(x)}_{X(\omega)} \cdot \underbrace{\mathcal{F}(h)}_{H(\omega)},$$
$$\text{and} \quad y[n] = x[n] * h[n] = \mathcal{F}^{-1}\{X(\omega) \cdot H(\omega)\} \tag{7}$$

where operators $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the discrete-time Fourier transform (DTFT) and its inverse, respectively. The complex-valued, multiplicative function $H(\omega)$ is the filter frequency response, and $X(\omega)$ and $Y(\omega)$ are the Fourier transform of the input and output sequences, respectively. Consequently, the Conv1D kernel applies an FIR frequency-selective filter to the input signal.

To implement an FIR filter using a CL, the direct form taps of the FIR filter are computed using the frequency response specifications of the filter; a single kernel Conv1D layer is instantiated with the kernel size set to $N_K = N + 1$, where N is the filter order, and N_K is the kernel size; the Conv1D layer kernel weights are initialized to the time-reversed version of the FIR filter taps; the bias is set to zero, and the stride length and dilation rate are set to 1. If CL training is disabled, the CL kernel will perform the pre-designed filtering operation, while enabling layer training would enable fine-tuning the filter parameters to optimize the model performance for the training data. The total number of FIR CL parameters is N_K since a single kernel is used.

This interpretation of a CL has an extensive range of applications as it enables building frequency-selective filters inside a DNN model at an arbitrary depth. One of the most common applications of frequency selective filtering is noise reduction. Noise removal filters with specific pass and stop band characteristics can be implemented using an FIR CL and instantiated as an input layer inside a DNN model. Such a layer will eliminate the need for a preprocessing noise removal stage and enable the development of self-contained models. Another application of the CL FIR layer is building moving average filters by instantiating a CL kernel with a specific length matching the filter width and assigning fixed weights to the kernel parameters. Implementation and validation of the FIR-based CL filter are provided in Appendix A.

The FIR interpretation of CLs does not only enable the development of frequency-selective filters inside a DNN model but also allows developers to analyze and visualize the frequency response of CL kernels for automatically trained models. Kernel weights of a trained model are extracted from CLs and time reversed to get the FIR filter impulse response $h_i[n]$. The kernel frequency response $H_i(f)$ is obtained by applying FFT to the impulse response $h_i[n]$. Such an approach can completely characterize the CL operation, especially in applications emphasizing frequency domain analysis such as machinery fault diagnosis [17], [18], [19].

C. 1D CONVOLUTIONAL LAYER AND STFT ALGORITHM

Some DNN models work better when fed the frequency spectrum of the input signal. Many feature engineering approaches have been proposed to use the 2D time-frequency domain representations of the 1D time series as an input feature to the model [10], [15], [16], [17], [18], [19], [20]. Among the most commonly used time-frequency representations are STFT and CWT. The STFT is a window-based FT in which the FT of a long sequence is repeatedly computed over separate short segments to produce the signal spectrogram, which contains the signal frequency spectra versus time [56]. The discrete-time STFT is defined as:

$$STFT\{x[n]\}(m,f) = X(m,f) = \sum_{n=-\infty}^{\infty} x[n]W[n-m]e^{-j2\pi f} \tag{8}$$

where $x[n]$ is the input sequence, $X(m,f)$ is the STFT of the sequence, and $W[n]$ is the window function.

To avoid using complex numbers inside the CL to compute the STFT as defined by 8, we propose an alternative approach based on the FIR implementation of the CL to compute a spectrogram-equivalent representation of the input signal. A Conv1D layer is instantiated with multiple kernels N_F each of size $N_K = N + 1$ to implement a bank of adjacent FIR bandpass filters (BPFs) of order N. Filter banks are commonly used for signal frequency decomposition because they enable extracting spectral components of the signal while providing efficient implementations [56]. Collectively, the filter bank works as the FT, where the output of each filter reveals a specific frequency component of the signal. Since the Conv1D kernel slides a window along the time axis, the filter bank output spectrum captures the change in signal frequency components with respect to time. Therefore, the Conv1D layer output is a time domain signal indicating the existence of specific frequency components in the signal at specific time instants to replicate the STFT operation.

To implement the STFT filter bank algorithm, N_F adjacent FIR filters are developed to cover a specific frequency range with a pre-set resolution. The kernel weights and Conv1D layer parameters will be assigned following the FIR filter design procedures presented in the last section. Increasing the

number of Conv1D kernels N_F extends the frequency range of the spectrum while increasing the kernel size N_K (FIR filter order) boosts spectrum resolution at the expense of increasing the number of layer parameters. Higher order FIR filters have steeper roll-off rates and higher quality factors [53]. The 1D kernel outputs are concatenated and reshaped into a 2D image representing the STFT-like spectrogram of the signal. The resulting spectrogram can then be handled as a single-channel 2D image and fed to Conv2D layers for further processing. The total number of STFT CL parameters is $N_F \cdot N_K$. Implementation and validation of the STFT-based CL are provided in Appendix B.

The STFT-based CNN has been presented in our previous work with the application to ECG classification. The ECG signal is processed inside the CNN model using the STFT CL to produce a 2D spectrogram signal fed to a 2D CNN for feature extraction and classification. The proposed self-contained model exhibits high classification performance, a shorter training time, low computational complexity, and superior real-time performance compared to the state-of-the-art rival models [46].

D. 1D CONVOLUTIONAL LAYER AND DTFT ALGORITHM

In various applications, such as machinery fault diagnosis, frequency domain representations are preferred over time domain features [17], [18], [19]. Some DNN models are fed with the input signals' magnitude spectrum or power spectral density, whereas others use a lambda layer to compute the fast Fourier transform (FFT) inside the model. The former approach requires a preprocessing step, while the latter is not amenable to quantization. The DTFT $X(f)$ of a discrete time signal $X[n]$ is defined as:

$$DTFT\{x[n]\} = X(f) = \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi f} \tag{9}$$

We propose a slight modification to the STFT CL to implement the DTFT. A GMP layer is instantiated after the STFT CL to pool the maximum value of each output channel of the filter bank. The pooling layer output represents the maximum frequency components of CL filters along the time axis. Consequently, the pooling layer output is equivalent to the magnitude spectrum of the input signal. A global average pooling (GAP) layer can be an alternative to GMP. In this case, the layer output represents the average frequency components of CL filters along the time axis. Similar to the STFT CL, increasing the number of Conv1D kernels N_F extends the frequency range of the spectrum, while increasing the kernel size N_K (FIR filter order) boosts the spectrum resolution at the expense of increasing the number of layer parameters. The total number of DTFT CL parameters is $N_F \cdot N_K$. Implementation and validation of the DTFT-based CL are provided in Appendix C.

At the analysis level, most existing CNN models use global pooling layers for dimensionality reduction in deep hierarchical models. According to the DTFT-CL perspective, the outputs of global pooling layers represent the maximum frequency components passed through a stack of cascaded CL FIR filters and local pooling layers, which perform rate downsampling. This interpretation can explain the domain shift problem arising in models with training and testing sets drawn from different domains from the deterministic system point of view rather than the statistical domain shift interpretations that dominate the ML literature. Changing the testing dataset distribution implies changing its spectral contents, which results in performance degradation of the model trained on data with different spectral contents. This interpretation is particularly important for CNN applications to machinery fault diagnosis because it precisely explains why such models trained on a specific dataset do not generalize well to other datasets with different faults and machine working conditions due to the frequency spectral variation between the training and testing sets.

E. 1D CONVOLUTIONAL LAYER AND CWT ALGORITHM

In STFT, a trade-off arises between the time and frequency resolutions. CWT overcomes this limitation by correlating dilated versions of a mother wavelet with the signal to extract a high-resolution time-frequency 2D image called a scalogram of the signal [56]. The real-valued CWT is defined as:

$$X_\omega(a,b) = \frac{1}{|a|^{1/2}} \int_{-\infty}^{\infty} x(t)\Psi(\frac{t-b}{a})dt = x(t) \star \Psi_a(t) \tag{10}$$

where $x(t)$ is the input signal, $\Psi(t)$ is the real-valued mother wavelet function, a and b are the real-valued scale and translational values of the wavelet, respectively, and $\Psi_a(t) = |a|^{-1/2}\Psi(\frac{t}{a})$ is the scaled wavelet function.

As depicted by 10, the CWT is computed by correlating the signal with dilated versions of the mother wavelet. The main operation performed by a CL is correlation, and hence assigning suitable weights embodied by the dilated wavelets to the layer kernels would enable computing the CWT of the signal. The implementation steps are pretty straightforward, which include selecting the required mother wavelet from the set of real-valued wavelet families such as Morlet or Mexican hat wavelets, computing N_F amplitude-scaled time-dilated versions of the mother wavelet $\Psi_a(t)$ each of a sequence length N_K according to the required time and frequency resolutions and frequency range; instantiating and Conv1D layer with N_F kernels each of N_K size; and assigning the dilated wavelet values to the Conv1D layer kernels. The wavelet family and dilation rates are determined according to the frequency spectral properties of the input data, desired scalogram characteristics, and DNN model requirements. The 1D kernel outputs are concatenated and reshaped into a 2D image representing the CWT scalogram of the signal, which can be handled as a 2D image of a single channel and fed to Conv2D layers for processing. The total number of CWT CL parameters is $N_F \cdot N_K$. Implementation and validation of the CWT-based CL are provided in Appendix D.

IV. APPLICATION EXAMPLE: MACHINERY FAULT DIAGNOSIS

This section will provide an illustrative example of using the proposed method to design and analyze DNNs in a practical application. This part aims not to benchmark the proposed method but to illustrate its potential in practical applications. For detailed benchmark results and comparative analysis, we refer to our previous works [44], [45], [46], which presented application and implementation of the presented CLs in full CNNs, evaluated their classification and real-time performance against state-of-the-art models in the ECG classification and HAR literature, and demonstrated their superiority and unique features.

Intelligent machinery fault diagnosis is AI-enabled research aiming to enable early detection and diagnosis of faults in different machine parts by monitoring machine conditions in real-time [17], [18], [19]. Various signals, including vibration, acoustic, and electric measurements, enable machine condition monitoring, of which vibration analysis is the most dominant method due to the rich information available in the vibration signals [57]. However, raw vibration signals are heavily contaminated by noise from several sources, including sensor noise, electromagnetic interference, and vibration components from other machine parts, making it difficult to use them directly in ML models. Therefore, a noise removal preprocessing stage is required to condition the vibration signal for ML settings. The FIR noise removal CL filter presented in Section III can be designed inside a DNN model, eliminating the need for the preprocessing stage.

On the other hand, machinery vibrations are non-stationary signals in which their spectral contents vary with time. Time-frequency analysis is an effective method for feature extraction from non-stationary signals [58]. A preprocessing stage is needed to extract the time-frequency features from the vibration signal before feeding them to the DNN model. CWT- or STFT-based CL proposed in Section III can be instantiated inside a DNN model to extract time-frequency features of the vibration signal, eliminating the need for the corresponding preprocessing stages.

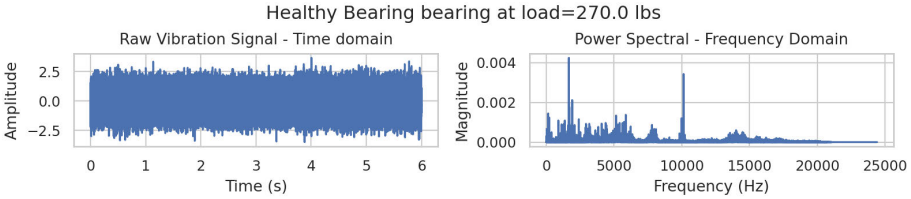
The frequency spectrum of vibration signals provides detailed information about the machine conditions, but it also contains resonance and natural high-frequency components that complicate diagnosing machine conditions. Envelope analysis is one of the most commonly used methods for effective fault diagnosis [59]. Faults and their harmonics can be easily spotted in the envelope spectral, which facilitates fault diagnosis. In envelope analysis, the vibration envelope is computed by applying the Hilbert transform and computing the DTFT of the absolute envelope [60]. The resulting envelope has a much lower bandwidth than the raw vibration signal due to demodulating high-frequency signal components and using an LPF and downsampler to reduce the envelope sampling rate and, consequently, the computational complexity of the subsequent stages.

The computation of the envelope spectral requires a pipeline of several preprocessing stages, including the Hilbert transform, downsampler, and LPF. The proposed DSP-based CLs can be used for envelope spectral estimation, eliminating the need for the preprocessing pipeline. In this section, we will design four CNN models for intelligent fault diagnosis using vibration signals that incorporate the DSP CLs presented in Section III for different purposes and illustrate their usage methodology. Two models are based on raw vibration analysis, whereas the remaining are based on envelope spectral analysis. The selected application illustrates the proposed method typical use case and potential. Moreover, we will provide a typical automatically trained CNN model and demonstrate their analysis methodology based on the interpretations presented in Section III. Again, we emphasize that the purpose of this work is neither benchmarking a specific ML model nor proving its superiority to existing models for a specific application but illustrating features of the proposed method on a practical application.

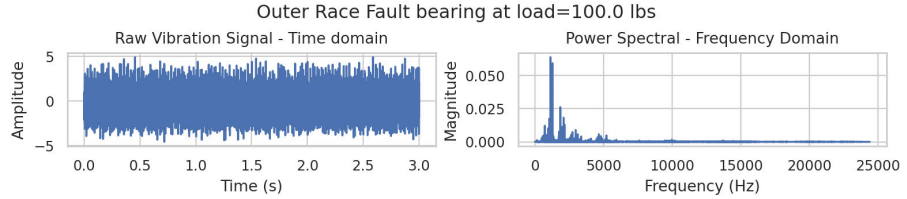
A. DATASET

Rolling element bearings are critical components in rotating machinery, and bearing faults account for 40% to 90% of problems in mechanical equipment [61]. Machinery Failure Prevention Technology (MFPT) is a bearing fault dataset provided by Society for Machinery Failure Prevention to push the machinery condition-based monitoring (CBM) research [62]. The test rig is equipped with a NICE bearing with healthy, outer race, and inner race fault conditions running at 1500 rpm speed and different workload conditions ranging from 0 to 300 lbs with sampling frequencies of 48,828 and 97,656 Hz. The bearing vibration signal is captured and recorded in mat files for different bearing conditions and working loads for 3 s at 48,282 sample/s and 6 s at 97,656 Hz sample/s.

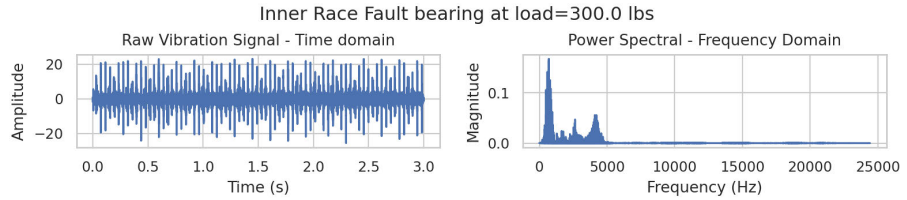
Figure 4 shows the raw vibration signals of randomly selected examples of different bearing conditions under different working loads in the time domain and their power spectra in the frequency domain. Figure 5 shows vibration envelopes of the selected examples. As illustrated by the figures, the raw vibration power spectrum comprises various frequency components ranging from 0 to 10 kHz and is contaminated by white noise. Spectral analysis of raw vibration signals indicates that an LPF of a cutoff frequency $f_{c1} = 10kHz$ can be used to remove high-frequency noise. The envelope power spectral reveals different single-frequency components at different frequencies and their harmonics ranging from 0 to 1 kHz, directly related to the fault class and severity [60]. It also reveals that the maximum frequency component of the envelope signal is in the range of 1 kHz, which instructs for using a second LPF of $f_{c2} = 1kHz$ to filter out undesired high-frequency components of the envelope and downsampling the envelope by a factor of $f_{c1}/f_{c2} = 10$. The DSP-based CL filter will be used to implement the needed filters inside the CNN model instead of using a preprocessing pipeline.



(a) Raw vibration signal of a healthy bearing in the time and frequency domains.

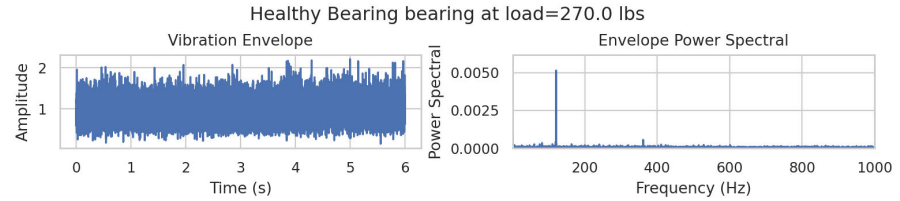


(b) Raw vibration signal of a bearing with an inner race fault in the time and frequency domains.

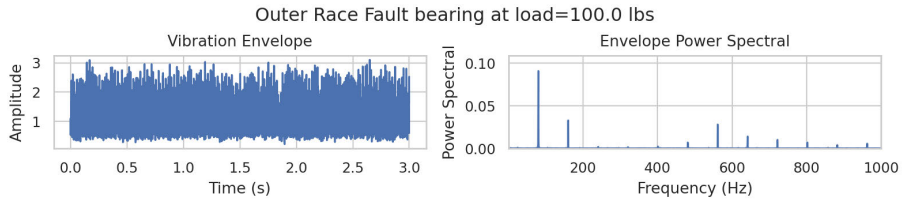


(c) Raw vibration signal of a bearing with an outer race fault in the time and frequency domains.

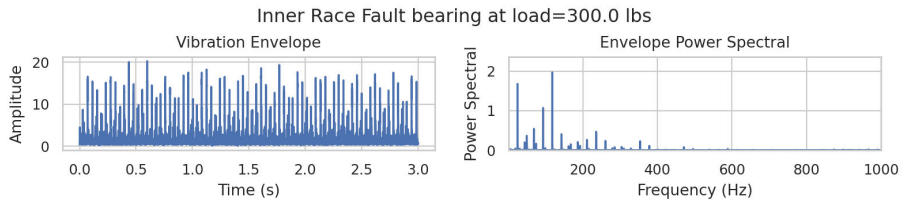
FIGURE 4. Visualization of random dataset examples of the raw vibration signals in the time and frequency domains.



(a) Envelope of the vibration signal of a healthy bearing in the time and frequency domains.



(b) Envelope of the vibration signal of a bearing with an inner race fault in the time and frequency domains.



(c) Envelope of the vibration signal of a bearing with an outer race fault in the time and frequency domains.

FIGURE 5. Visualization of random dataset examples of the raw vibration signals in the time and frequency domains.

The sampling frequency of all vibration records is unified at 48,828 by downsampling files with higher sampling frequencies. Each vibration record is partitioned into segments, each containing 4096 samples equivalent to 0.084 s,

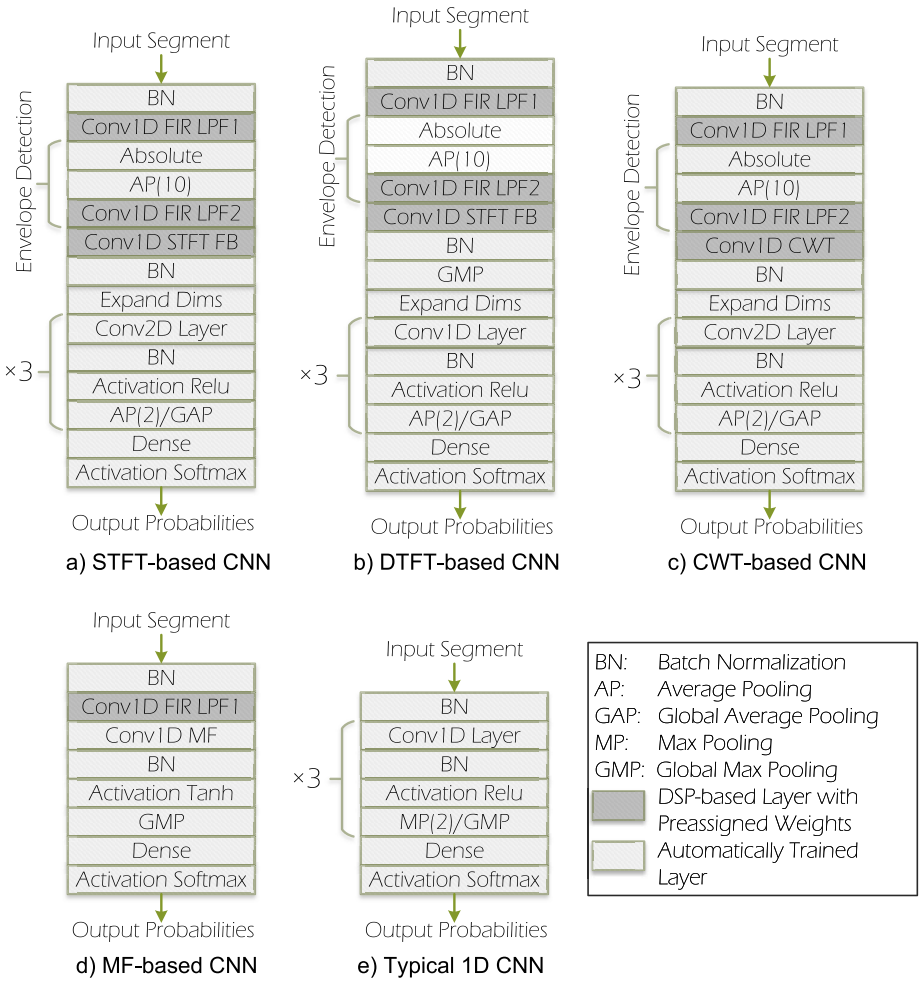


FIGURE 6. The developed models: a) STFT-based CNN classifier, b) DTFT-based CNN classifier, c) CWT-based CNN classifier, d) MF-based CNN classifier, e) Typical 1D CNN classifier.

and labeled according to the bearing condition into three classes: healthy bearing, inner race faulty bearing, and outer race faulty bearing. Segments with a number of samples less than 4096 are zero-padded to produce unified length sequences, which is required by the CNN model. The bearing rotation speed of 60 Hz is considered in selecting the segmentation frequency, which equals 11.92 Hz, *i.e.* each segment covers around five rotations of the bearing. A total of 748 examples, each contains 4096 samples, are produced and partitioned into three classes: 173 healthy examples, 374 outer race fault examples, and 201 inner race fault examples. The dataset is randomly split and stratified into training, validation, and holdout testing sets of 598, 150, and 188 examples, respectively. Vibration segments are fed directly to the developed CNN models without applying preprocessing or feature engineering stages to the raw vibration signals.

B. DESIGN OF SELF-CONTAINED CNNs

Five CNN classifier models for bearing fault diagnosis are developed and analyzed using the proposed DSP-based interpretations of the 1D CL presented in this work. Figure 6 depicts the developed models, which are: STFT-based CNN, DTFT-based CNN, CWT-based CNN, MF-based CNN, and a typical automatically trained 1D CNN. All models are built using standard layers provided by TensorFlow and trained on the MFPT bearing dataset with the categorical cross-entropy loss function and Adam optimizer with a learning rate of 0.001. Models are trained on a cloud workstation featuring 8 CPU cores, 30 GB of RAM, and an Nvidia RTX A4000 GPU with 16 GB of VRAM. The training and validation sets are used for model training on mini-batches of size 64 with 100 epochs. Trained models are tested on the holdout testing set, and the testing accuracy is reported. The main objective of the developed models is to illustrate features and application mechanisms of the proposed methods; thus, model hyper-parameters are selected to enable visualization of the model kernels in the time and frequency domains, not to maximize the model classification performance. Nevertheless, all presented models achieve acceptable classification results, as will be illustrated.

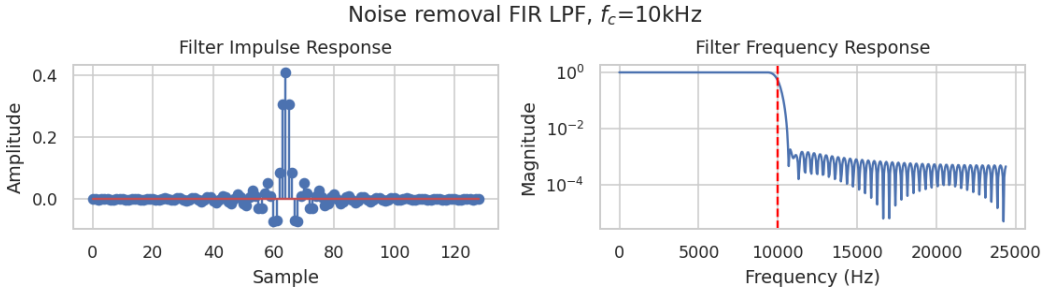


FIGURE 7. Impulse and frequency responses of the noise removal FIR LPF 1 with order $N = 128$ and $f_c = 10$ kHz.

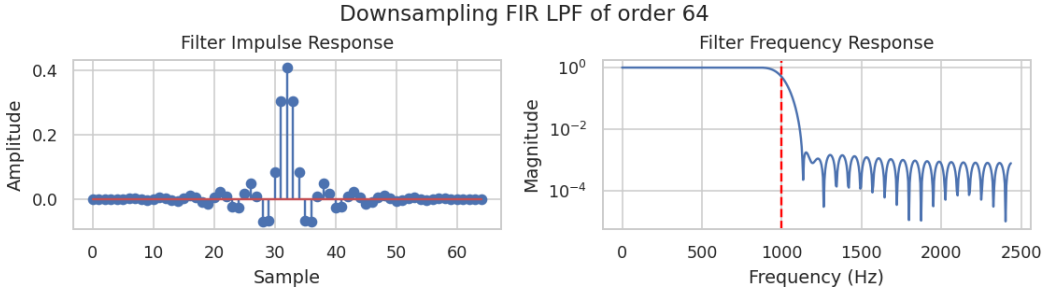


FIGURE 8. Impulse and frequency responses of the downsampling FIR LPF 2 with order $N = 64$ and $f_c = 1$ kHz.

We will start by presenting the design steps of CLs used in all models and then discuss each model in detail. The noise removal layer is designed with an FIR LPF of $f_{c1} = 10kHz$. Figure 7 shows the impulse and frequency responses of the noise removal FIR filter with order = 128. Filter taps are time reversed and assigned to an inactivated linear Conv1D FIR LPF1, the bias term is set to zero, and the layer training is disabled as instructed in Section III. This layer is instantiated to filter out high-frequency noise from raw vibration signals after a batch normalization (BN) layer. The BN layer is instantiated as an input stage in all models, which is used to normalize model inputs inside the CNN model and helps to mitigate the covariate shift between training set batches.

Three models are developed to classify the vibration signal based on envelope analysis, resulting in better classification performance due to filtering out irrelevant high-frequency components and reducing the computational cost due to downsampling the signal by a significant factor. The envelope segments are downsampled to 409 samples compared to the raw vibration segments of 4096 samples, significantly reducing the CL computation time. Employing the vibration envelope for classification is critical due to the high sampling rates of the vibration signals, which results in a prolonged segment size that incurs increased computational cost.

A stack of three layers is used to estimate the vibration envelope. A lambda layer is instantiated to compute the absolute vibration value, an average pooling (AP) layer with a pool size of 10 is used to downsample the signal, and an FIR LPF of $f_{c2} = 1kHz$ and order $N = 64$ is designed using a CL similar to the noise removal LPF 1.

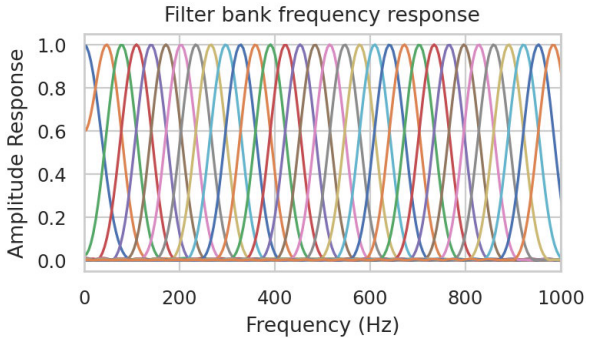


FIGURE 9. Frequency response of the STFT layer filter bank of equally-spaced 32 FIR BPFs and $f_c = 1KHz$.

The proposed method for envelope estimation is based on the heuristic approach presented in [63], which does not require the Hilbert transform for envelope detection, with a slight modification to use the standard neural layers. We verified the accuracy of the envelope estimation method against the Hilbert transform-based computation method for all examples of the dataset, and the resulting mean square error does not exceed 10%.

1) STFT-BASED CNN

This model extracts time-frequency spectrogram images of the vibration envelope and feeds them to a 2D CNN for classification. A bank of 32 equally-spaced filters is designed between 0 and 1 kHz as shown in Figure 9, and the filter taps are time-reversed (flipped) and assigned to an inactivated linear Conv1D layer. The layer bias term is set to zero, and

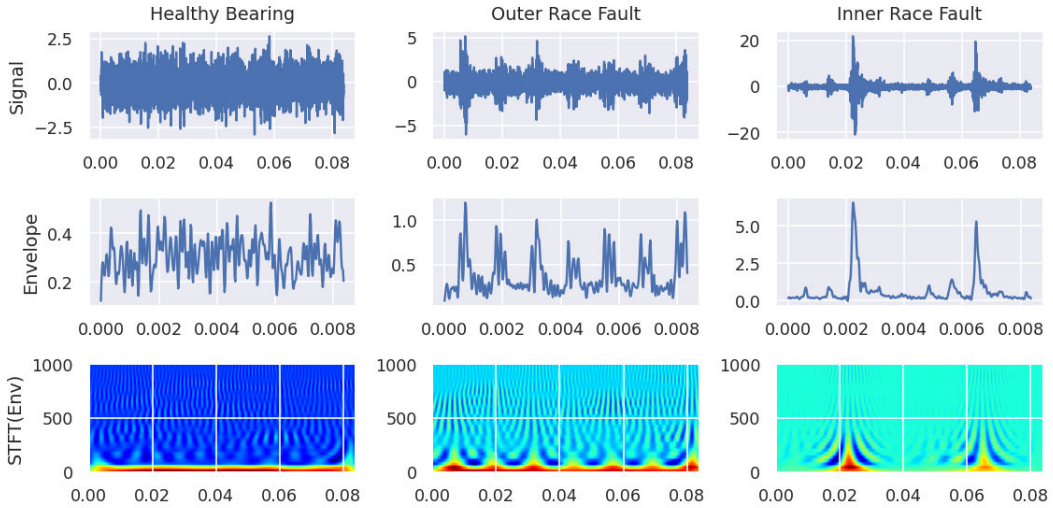


FIGURE 10. Visualization of the envelope estimation stack and STFT layer feature map outputs for random examples of different classes.

TABLE 1. STFT-based CNN model (envelope) layers and parameters.

Layer	Function	Trainable	Output Shape	Params	(N_F, N_K)
Input	Shape	FALSE	(4096, 1)	0	
BN	Normalization	TRUE	(4096, 1)	4	
Conv1D	LPF 1	FALSE	(4096, 1)	129	(1, 129)
Lambda	Absolute	FALSE	(4096, 1)	0	
AP(10)	Downsampling	FALSE	(409, 1)	0	
Conv1D	LPF 2	FALSE	(409, 1)	65	(1, 65)
Conv1D	STFT	FALSE	(409, 32)	4096	(32, 128)
BN	Normalization	TRUE	(409, 32)	128	
Lambda	Expand Dims	FALSE	(409, 32, 1)	0	
Conv2D	Layer 1	TRUE	(409, 32, 10)	260	(10, 5)
BN	Normalization	TRUE	(409, 32, 10)	40	
Activation	Relu	FALSE	(409, 32, 10)	0	
AP(2)	Downsampling	FALSE	(205, 16, 20)	0	
Conv2D	Layer 2	TRUE	(205, 16, 20)	5020	(20, 5)
BN	Normalization	TRUE	(205, 16, 20)	80	
Activation	Relu	FALSE	(205, 16, 20)	0	
AP(2)	Downsampling	FALSE	(103, 8, 20)	0	
Conv2D	Layer 1	TRUE	(103, 8, 40)	20040	(40, 5)
BN	Normalization	TRUE	(103, 8, 40)	160	
Activation	Relu	FALSE	(103, 8, 40)	0	
GAP	Pooling	FALSE	(40)	0	
Dense	SoftMax	TRUE	(3)	123	

the layer training is disabled as instructed in Section III. This layer is instantiated after the envelope detection layer to extract the time-frequency spectrogram of the envelope signal and feed them to a typical hierarchical stack of Conv2D, BN, Relu activation, and AP layers of a pool size of 2. The Conv2D stack output is fed to a GAP layer and dense output layer with Softmax activation for the classification task. Table 1 presents the layers and parameters used in the STFT-based CNN model shown in Figure 6a.

Figure 10 visualizes the envelope estimation stack outputs and STFT Conv1D layer feature maps for randomly selected examples of the three bearing classes. As demonstrated by this figure, the envelope estimation stack produces a representative estimation of the signal envelope. On the other hand, the STFT CL produces high-resolution spectrograms of (409 × 32) size with distinguishable features between different classes. The DC average of the envelope signal is salient in the first two classes, the periodicity of the last two classes is identified in their spectrograms, and even the modulation effect featuring the inner race fault is spotted.

2) DTFT-BASED CNN

This model extracts the frequency spectra of the vibration envelope and feeds them to a 1D CNN for classification. The same STFT filter bank depicted in Figure 9 is used for extracting the time-frequency spectrogram of the vibration envelope of a (409 × 32) resolution. A GAP layer is instantiated after the STFT CL to compute the layer average along the time axis to estimate a 32-point DTFT of the envelope signal. The computed envelope spectral is then fed to a typical 1D CNN composed of stacks of Conv1D, BN, GAP, and Relu activation layers. The Conv1D stack output is fed to a GAP layer followed by a dense layer with Softmax activation for the classification output. Table 2 presents the layers and parameters used in the DTFT-based CNN model shown in Figure 6b.

Figure 11 visualizes the envelope estimation stack outputs, STFT Conv1D layer feature maps, and the DTFT layer outputs for randomly selected examples of the three bearing classes. Due to disabling layer training, the envelope detection stack and STFT layer work similarly to the STFT-based CNN. The DTFT layer computes an estimate of the envelope spectral with distinguishable patterns between various classes. The 32-point spectral resolution can be enhanced by increasing the number of filters of the STFT CL at the expense of increasing the layer number of parameters and computation load.

3) CWT-BASED CNN

This model extracts time-frequency scalogram images of the vibration envelope and feeds them to a 2D CNN for

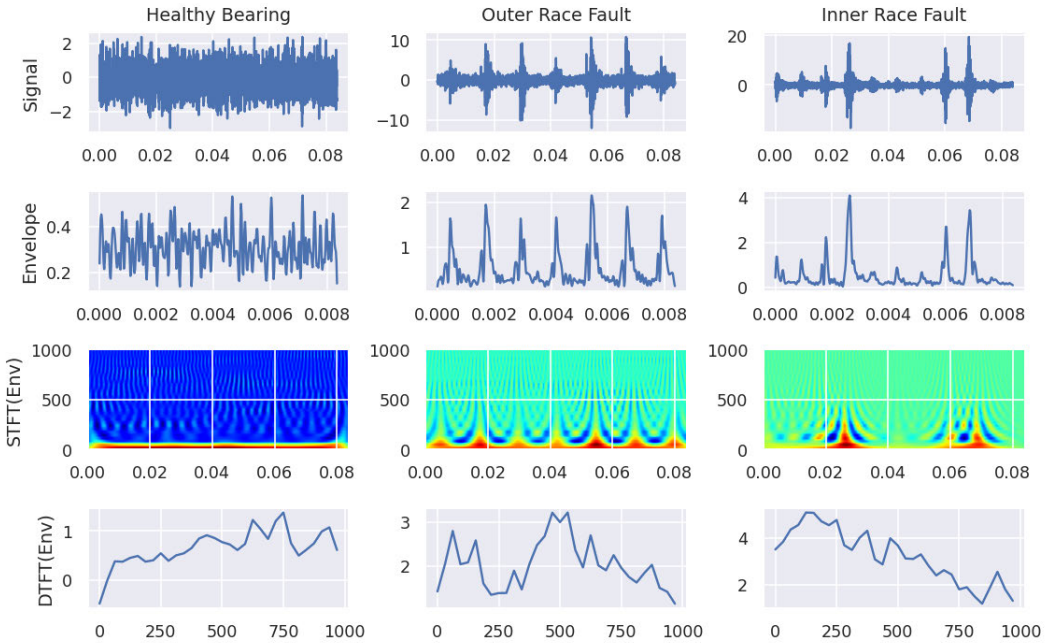


FIGURE 11. Visualization of the DTFT layer feature map outputs for random examples of different classes.

TABLE 2. DTFT-based CNN model (envelope) layers and parameters.

Layer	Function	Trainable	Output Shape	Params	(N_F, N_K)
Input	Shape	FALSE	(4096, 1)	0	
BN	Normalization	TRUE	(4096, 1)	4	
Conv1D	LPF 1	FALSE	(4096, 1)	129	(1, 129)
Lambda	Absolute	FALSE	(4096, 1)	0	
AP(10)	Downsampling	FALSE	(409, 1)	0	
Conv1D	LPF 2	FALSE	(409, 1)	65	(1, 65)
Conv1D	STFT	FALSE	(409, 32)	4096	(32, 128)
BN	Normalization	TRUE	(409, 32)	128	
GMP	AlongTimeAxis	FALSE	(32)	0	
Lambda	Expand Dims	FALSE	(32, 1)	0	
Conv1D	Layer 1	TRUE	(32, 10)	330	(10, 32)
BN	Normalization	TRUE	(32, 10)	40	
Activation	Relu	FALSE	(32, 10)	0	
AP(2)	Downsampling	FALSE	(16, 10)	0	
Conv1D	Layer 2	TRUE	(16, 20)	3220	(20, 16)
BN	Normalization	TRUE	(16, 20)	80	
Activation	Relu	FALSE	(16, 20)	0	
AP(2)	Downsampling	FALSE	(8, 20)	0	
Conv1D	Layer 3	TRUE	(8, 40)	6440	(40, 8)
BN	Normalization	TRUE	(8, 40)	160	
Activation	Relu	FALSE	(8, 40)	0	
GAP	Downsampling	FALSE	(40)	0	
Dense	SoftMax	TRUE	(3)	123	

classification. A Morlet mother wavelet is used to extract 64 dilated wavelets, each of a 409 sample length (length of the envelope segment), as shown in Figure 12. Wavelet scales are calculated to produce filters covering the frequency range of envelope signals (0 – 1 kHz). The dilated wavelets are assigned to an inactivated linear Conv1D layer, the layer bias term is set to zero, and the layer training is disabled as instructed in Section III. This layer is instantiated after the envelope detection layer to extract the time-frequency scalogram of the envelope signal and feed them to a typical hierarchical stack of Conv2D, BN, Relu activation, and AP

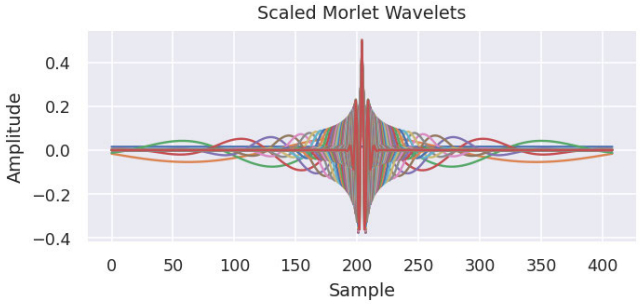


FIGURE 12. Impulse response of the CWT layer kernels of 64 scaled Morlet wavelets each of 409 samples.

layers of a pool size of 2. The Conv2D stack output is fed to GAP and a dense output layer with Softmax activation for the classification task. Table 3 presents the layers and parameters used in the CWT-based CNN model shown in Figure 6c.

Figure 13 visualizes the envelope estimation stack outputs and CWT Conv1D layer feature maps for randomly selected examples of the three bearing classes. The CWT CL produces high-resolution scalograms of (409 × 64) size with distinguishable features between different classes.

C. ANALYSIS OF 1D CONVOLUTIONAL LAYERS

Two models are developed to illustrate the analytic power of the proposed method: the MF-based CNN and a typical automatically trained 1D CNN. Unlike the three preceding classifiers, which incorporate 2D CLs, both classifier models are developed using only 1D CLs and work on the raw vibration signal rather than the envelope due to the relaxed computational requirements of 1D CLs. Both models

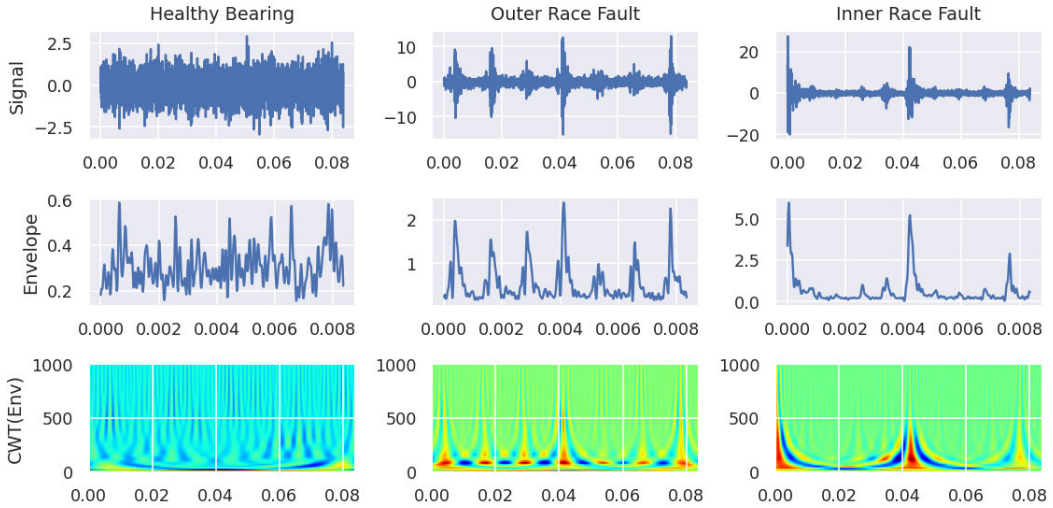


FIGURE 13. Visualization of the CWT layer feature map outputs for random examples of different classes.

TABLE 3. CWT-based CNN model (envelope) layers and parameters.

Layer	Function	Trainable	Output Shape	Params	(N_F, N_K)
Input	Shape	FALSE	(4096, 1)	0	
BN	Normalization	TRUE	(4096, 1)	4	
Conv1D	LPF 1	FALSE	(4096, 1)	129	(1, 129)
Lambda	Absolute	FALSE	(4096, 1)	0	
AP(10)	Downsampling	FALSE	(409, 1)	0	
Conv1D	LPF 2	FALSE	(409, 1)	65	(1, 65)
Conv1D	CWT	FALSE	(409, 64)	26176	(64, 409)
BN	Normalization	TRUE	(409, 64)	256	
Lambda	Expand Dims	FALSE	(409, 64, 1)	0	
Conv2D	Layer 1	TRUE	(409, 64, 10)	260	(10, 5)
BN	Normalization	TRUE	(409, 64, 10)	40	
Activation	Relu	FALSE	(409, 64, 10)	0	
AP(2)	Downsampling	FALSE	(205, 32, 10)	0	
Conv2D	Layer 2	TRUE	(409, 32, 20)	5020	(20, 5)
BN	Normalization	TRUE	(409, 32, 20)	80	
Activation	Relu	FALSE	(409, 32, 20)	0	
AP(2)	Downsampling	FALSE	(103, 16, 40)	0	
Conv2D	Layer 1	TRUE	(103, 16, 40)	20040	(40, 5)
BN	Normalization	TRUE	(103, 16, 40)	160	
Activation	Relu	FALSE	(103, 16, 40)	0	
GAP	Pooling	FALSE	(40)	0	
Dense	SoftMax	TRUE	(3)	123	

automatically learn CL weights in the model training phase using the backpropagation and gradient descent algorithms. After model training, learned CL kernels are extracted from the model and analyzed in the time and frequency domains by treating the CL as an FIR filter described in Section III.

1) MF-BASED CNN

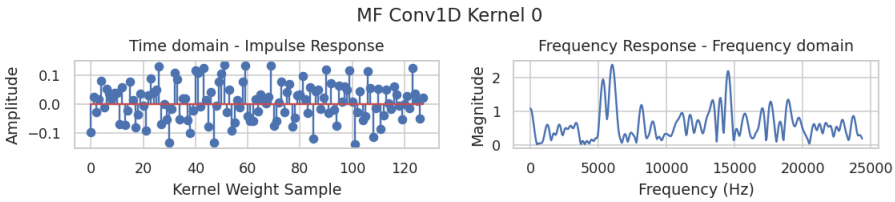
This model is built to work as an MF classifier, as shown in Figure 6d. The pre-designed FIR filter LPF1 is instantiated after the initial BN layer for noise removal. An automatically trained Conv1D MF layer is then inserted, followed by a second BN layer, a Tanh activation layer, a GMP layer, and a dense layer with Softmax activation for the classification task. This model is a shallow CNN with a single Conv1D layer correlator, a GMP layer for selecting maximum correlation outputs, a Tanh activation layer for maximum thresholding

TABLE 4. MF-based CNN model (raw signal) layers and parameters.

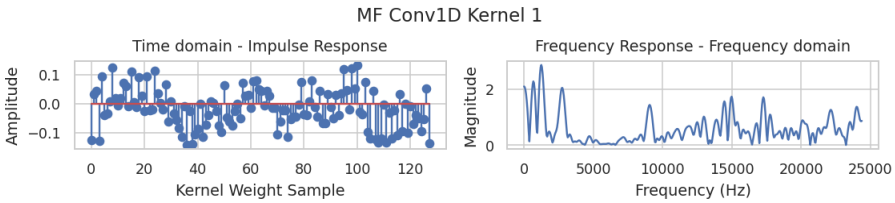
Layer	Function	Trainable	Output Shape	Params	(N_F, N_K)
Input	Shape	FALSE	(4096, 1)	0	
BN	Normalization	TRUE	(4096, 1)	4	
Conv1D	LPF 1	FALSE	(4096, 1)	129	(1, 129)
Conv1D	MF	TRUE	(4096, 3)	384	(3, 128)
BN	Normalization	TRUE	(4096, 3)	12	
Activation	Tanh	FALSE	(4096, 3)	0	
GMP	Pooling	TRUE	(3)	0	
Dense	SoftMax	TRUE	(3)	9	

correlations, and a Dense layer for mapping the threshold outputs to the corresponding class outputs. Table 4 presents the layers and parameters used in the MF-based CNN model. The MF Conv1D layer comprises three kernels, each of length 128; the bias term is set to zero. The number of kernels is set to three to match the number of classes in the MFPT dataset, which indicates that each kernel should learn a distinguishable pattern corresponding to one of the classes. The MF Conv1D parameters can be tuned to learn more distinguishable patterns for each class, but we opted to select minimal parameters for better visualizations.

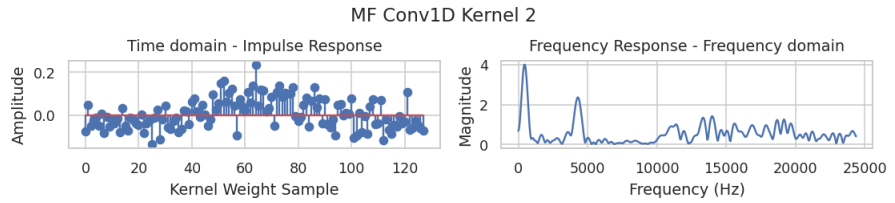
Next to model training, learned MF Conv1D kernel weights are extracted from the model and flipped to get their time-reversal, equivalent to the impulse response of an FIR filter as described in III. Afterwards, the FFT of the FIR impulse response is computed to get the frequency response of the FIR filter. Figure 14 depicts the impulse and frequency responses of the three MF kernels. time domain analysis of the trained kernels demonstrates the MF templates learned by the CNN, which discriminate between different classes of the training set. Learned templates are directly affected by the kernel length, number of kernels, training set size, weight initialization method, and other factors; however, they still provide an effective means to understand the learning process of the CNN. Frequency domain analysis



(a) MF Conv1D Kernel 0 in the time and frequency domains.



(b) MF Conv1D Kernel 1 in the time and frequency domains.



(c) MF Conv1D Kernel 2 in the time and frequency domains.

FIGURE 14. Visualization of the MF CNN Conv1D automatically-learned kernels in the time and frequency domains.

of the frequency response functions learned by the Conv1D kernels depicts frequencies passed and stopped by each filter. As shown by Figure 14a, Kernel 0 is a BPF passing frequencies between 5-6 kHz and 14-15 kHz, Kernel 1 is BPF passing selective frequencies between 0-3kHz and some higher range frequencies, and Kernel 2 is a BPF passing frequencies between 0-1 kHz and 4-5 kHz. These frequency ranges are tightly related to bearing fault frequencies, indicating that the MF Conv1D layer learns distinguishable fault frequencies during model training. This analysis also explains why a CNN-based fault diagnosis model trained using a dataset collected under specific working conditions does not generalize well for other datasets of different working conditions due to changing the learned fault frequencies.

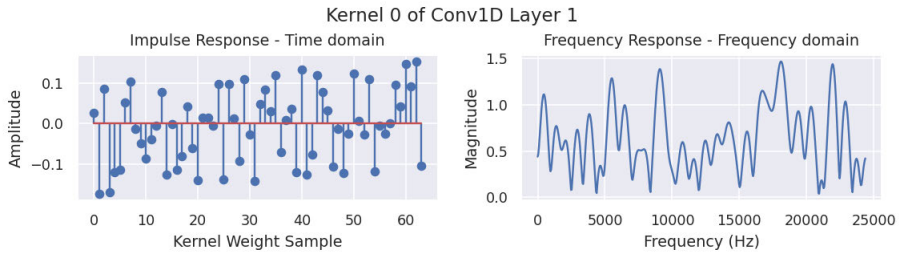
2) TYPICAL 1D CNN

This model is designed as an automatically trained model resembling a typical 1D CNN as shown by Figure 6e. The model comprises hierarchical stacks of Conv1D, BN, AP, and Relu activation layers and a dense output layer with Softmax activation for the classification task. Such a topology represents the prevalent usage scenario of CNNs in the ML literature. Model layers and parameters are presented in Table 5. The first layer of this model is a single-input, multi-output Conv1D layer with three kernels each of 64 kernel size, and the bias term is not zeroed. This layer can be analyzed based on the method presented in this work, which is confined to single-input, multi-output 1D CLs. The subsequent Conv1D layers are multi-input, multi-output CLs where each output channel is calculated as the sum of the correlation between all input channels and corresponding kernels. Currently, the FIR-based analysis method presented in this work does not address multi-input, multi-output CLs with merged-channel outputs, but it can be applied to separable and depthwise Conv1D CLs with separate channel-wise operation [45].

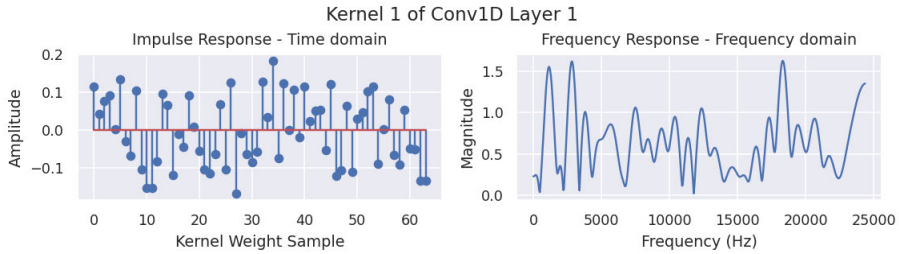
Next to model training, learned Conv1D kernel weights are extracted from the model and flipped to get their time-reversal, equivalent to the impulse response of an FIR filter as described in III. Afterwards, the FFT of the FIR impulse response is computed to get the frequency response of the

TABLE 5. Typical 1D CNN model (raw signal) layers and parameters.

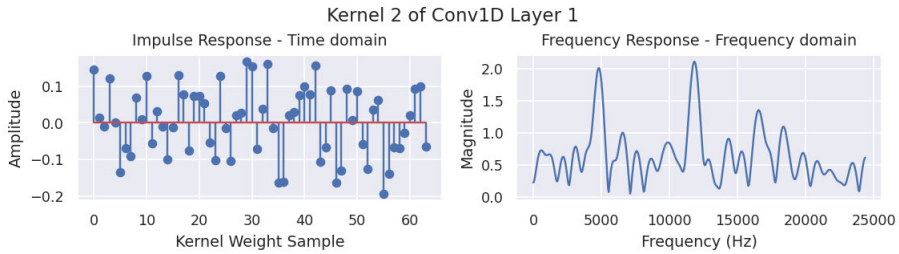
Layer	Function	Trainable	Output Shape	Params	(N_F, N_K)
Input	Shape	FALSE	(4096, 1)	0	
BN	Normalization	TRUE	(4096, 1)	4	
Conv1D	Layer 1	TRUE	(4096, 3)	195	(3, 64)
BN	Normalization	TRUE	(4096, 3)	12	
Activation	Relu	FALSE	(4096, 3)	0	
MP(2)	Downsampling	FALSE	(2048, 3)	0	
Conv1D	Layer 2	TRUE	(2048, 6)	582	(6, 32)
BN	Normalization	TRUE	(2048, 6)	24	
Activation	Relu	FALSE	(2048, 6)	0	
MP(2)	Downsampling	FALSE	(1024, 6)	0	
Conv1D	Layer 3	TRUE	(1024, 12)	1164	(12, 16)
BN	Normalization	TRUE	(1024, 12)	48	
Activation	Relu	FALSE	(1024, 12)	0	
GAP	Downsampling	FALSE	(12)	0	
Dense	SoftMax	TRUE	(3)	39	



(a) 1D CNN Layer 1, Kernel 0 in the time and frequency domains.



(b) 1D CNN Layer 1, Kernel 1 in the time and frequency domains.



(c) 1D CNN Layer 1, Kernel 2 in the time and frequency domains.

FIGURE 15. Visualization of the 1D CNN Conv1D layer 1 automatically-learned kernel in the time and frequency domains.

FIR filter. Figure 14 depicts the impulse and frequency responses of the three MF kernels. Each kernel resembles a frequency-selective filter with specific pass and stop frequencies related to the fault frequencies. As the first layer, filters learned by this layer directly affect subsequent layers and delineate frequencies passed to their enclosed kernels. Due to using a smaller kernel size compared to the MF kernels, the roll-off rate of the filter is reduced. Due to employing a deeper network topology, the pass and stop bands associated with each class cannot be determined entirely from the first layer but from the full cascaded stacks. Moreover, the nonlinear activation functions introduce nonlinear distortion components affecting the operation and decision mechanism of the entire network, yet this does not invalidate the interpretation presented for the single-input multi-output CL.

3) RESULTS AND DISCUSSION

Table 6 presents the training time, average inference time, total number of parameters, and testing accuracy results of the developed models. These results are obtained with a basic tuning of the model hyperparameters because the objective is to illustrate the proposed method potentials and application mechanisms, not to benchmark the developed models. Nevertheless, all models achieve good classification performance on the holdout testing set. The STFT-based CNN achieves the best accuracy of 100% and the 1D CNN achieves the lowest accuracy of 96.27%. The time-frequency and spectral envelope-based methods have better performance than the MF and 1D CNN models applied to the raw vibration signal, which indicates the superiority of envelope spectral analysis for bearing fault diagnosis. On the other hand, all DSP-based CNN models achieve better performance compared to the automatically trained 1D CNN, indicating the potential of the proposed method. It should be indicated that with slight fine-tuning of model hyperparameters, all models can achieve 100% accuracy, but performance is sacrificed for better visualizations.

TABLE 6. Testing results of the developed models.

Model	MF	1D CNN	CWT	STFT	DTFT
Total # of Params	541	2065	132,873	110,665	14,815
Training Time (s)	24.20	28.13	26.82	36.41	24.55
Inference Time (ms)	0.24	0.314	0.42	0.36	0.42
Testing Accuracy %	97.87	96.27	99.46	100.00	99.46

However, classification performance comes at the expense of increased model size, as indicated by the total number of parameters used by each model. At this level, the MF-based and 1D CNNs require only 541 and 2065 parameters, respectively, compared to the CWT-, STFT-, and DTFT-based models, which need 132,873, 110,665, and 14,815 parameters, respectively. The increased number of CWT- and STFT-based CNNs is attributed to using deeper stacks of 2D CLs applied to the scalogram and spectrogram output images produced by the 1D DSP-based CLs. Notwithstanding, the total number of parameters of these models is relatively small compared to existing models presented in the fault diagnostic literature [17], [18], [19].

The training time of all models does not exceed 1 minute due to the small size of the training set. Comparatively, the MF-based CNN has the lowest training time of 24.2 s due to the shallow nature of the model, while the STFT-based CNN has the highest training time of 36.41 s. The short training time of all models is attributed to the small size of the MFPT dataset and the reduced computational complexitiy of the developed models. Another factor affecting the training time is pre-assigning CL weights which reduces the number of model trainable parameters.

The average inference time is computed by calculating the total inference time of the testing set and dividing the result by the number of examples in the set. The MF-based CNN has the lowest inference time of 0.24 ms, while the CWT- and STFT-based CNNs have the highest inference time of 0.42 ms due to incorporating 2D CLs. Generally, the efficient resource usage and low computational time of the presented models enable their deployment on resource-constrained devices for edge inference, which has been experimentally established in our previous works [45], [46], [47].

Eventually, the proposed DSP-based CL design method complements the automatic feature extraction capabilities of CNNs by advancing a means to implement frequently used DSP preprocessing tasks inside the model using layers of the model. A single CL can be used to build DSP transformations such as STFT and CWT whereas a stack of neural layers such as pooling layers, non-linear activation functions, and dense layers can be stacked to build more complex DSP tasks such as MF, DTFT, and envelope detection. Self-contained CNN models are portable, resource-efficient, faster, and optimization-compatible compared to the separate preprocessing pipelines. The DSP-based CLs have full access to the model computational resources and accelerators, unlike separate preprocessing pipelines that are neither optimized nor accelerated using the model hardware delegates. DSP-inclusive models can be fully optimized and quantized for deployment and execution on edge devices, special-purpose accelerators, or specific instruction set architectures.

Nevertheless, the proposed method has some limitations at both the design and analysis levels. All DSP algorithms implemented using CLs are based on the correlation and equivalent convolution operations performed by CLs, limiting the scope of potentially developed algorithms. For example, the renowned FFT algorithm cannot be directly implemented using the proposed method because it is not based on convolution or correlation. Another limitation posed by the state-of-the-art machine learning development packages is the lack of support for complex number layers, which are commonly needed by many DSP algorithms. At the analysis level, the proposed method can only be used to analyze the single-input, multi-output, separable, and depth-wise CLs, which are addressed in the definitions of the convolution and correlations presented in Section III. The proposed method does not present an interpretation of the multi-input, multi-output CLs with mixed convolution across input channels.

V. CONCLUSION AND FUTURE WORK

In this work, we presented signal processing interpretations of the convolutional layer in neural networks and translated them into practical implementations of DSP algorithms that can be used for various preprocessing and feature extraction tasks inside a DNN model. Specifically, we exploited the correlation and equivalent convolution operations performed by a CL to implement FIR filters, MFs, STFT, DTFT, and CWT algorithms. The proposed interpretations of the CL are based on a straightforward understanding of the layer correlation and convolution signal processing operations. The implementation steps of each DSP algorithm were detailed, and validation visualizations were provided.

Afterwards, we advanced a comprehensive application example of mechanical bearing fault diagnostic to illustrate how to use the proposed method to design DSP-inclusive CNN models and analyze automatically trained CNNs. The STFT-, CWT-, and DTFT-based CNNs were designed using the proposed DSP-based CLs to extract the spectrogram, scalogram, and frequency spectral of the vibration envelope. An FIR-based LPF was designed for noise removal, and a stack of an absolute layer, AP layer, and downsampling FIR-based CL LPF were developed for envelope detection. MF-based and automatically trained 1D CNNs were advanced to illustrate how to analyze single-input, multi-output CLs in the time and frequency domains. All models are evaluated in terms of classification accuracy, total number of model parameters, training time, and average inference time.

The proposed approach alleviates the need for preprocessing and feature engineering steps and opens new frontiers in ML architectures and applications. Moreover, the presented CL interpretations provide a means to analyze and explain the operation of DNN models in both the time and frequency domains. Eventually, the CL signal processing interpretations advanced in this study will be immensely valuable for both the design and analysis of DNNs, opening new frontiers in ML research.

In future work, we will investigate CL implementations of other convolution-based DSP algorithms. We will study extending the interpretations presented in this work to Conv1D layers with multiple input channels and other types of convolutional layers, such as depth-wise and separable

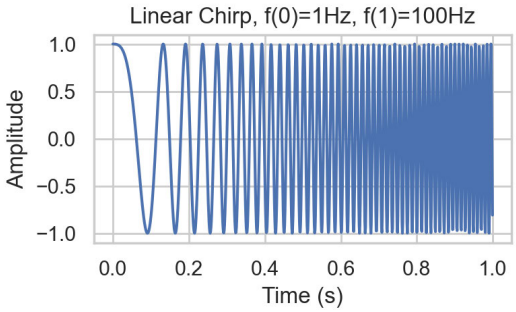
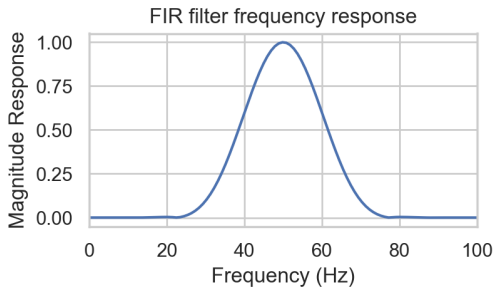
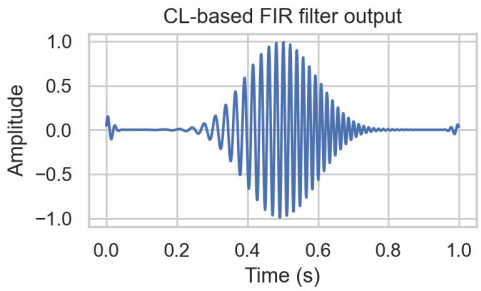


FIGURE 16. Linear chirp testing signal of a 1 s duration and a linear frequency range from 1 to 100 Hz.

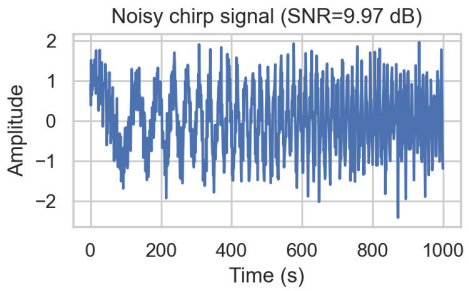


(a) Frequency response of the FIR BPF Filter of order 99 and passband between 40 and 60 Hz.

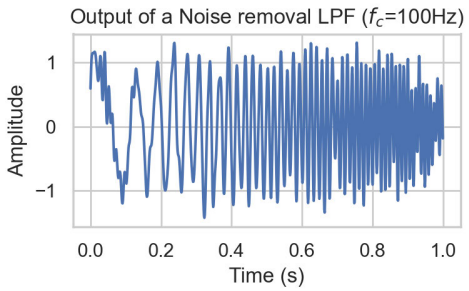


(b) FIR-based CL BPF layer output.

FIGURE 17. Visualization of the CL FIR filter applied to the chirp input signal.

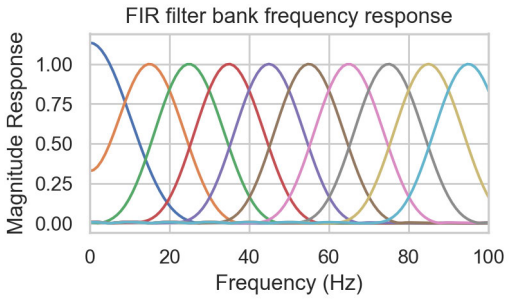


(a) Noisy input signal

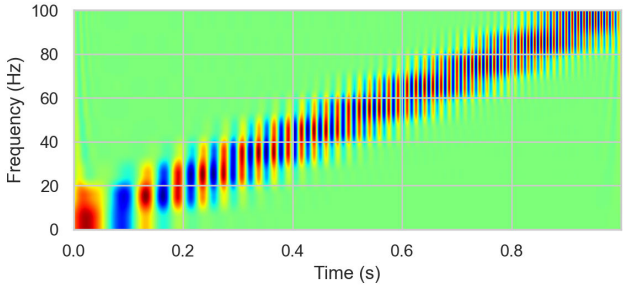


(b) Noise removal CL LPF output

FIGURE 18. Application of the CL FIR filter for noise removal.



(a) Frequency response of the FIR Filter Bank of 10 filters each of order 99.



(b) STFT-based CL layer output for $N_F = 10$ and $N_K = 100$.

FIGURE 19. Visualization of the 2D spectrogram produced by the CL STFT layer.

CLs. Also, we will study applying the CL interpretations presented in this work to 2D CLs. Moreover, we plan to study the effect of manipulating CL hyper-parameters such as stride length and dilation rate on the DSP algorithm functionality. Furthermore, interpretation and DSP implementations of other layers, such as dense, pooling, and activation layers, will be explored. Another research direction inspired by this study is developing custom DSP layers rather than relying on the standard set of layers provided by existing ML development packages. Finally, we will explore other potential applications of the proposed methodology.

APPENDIX

In this appendix, we will provide the development steps and validation results of the CL-based DSP algorithms presented in this article. The algorithm implementation details will be disclosed using Keras with the TensorFlow backend and Python programming language. Keras is an open-source Python interface for TensorFlow. TensorFlow is Google open-source machine learning framework, equipped with a rich ecosystem of tools, libraries, and community resources to help developers build and deploy ML-powered apps.

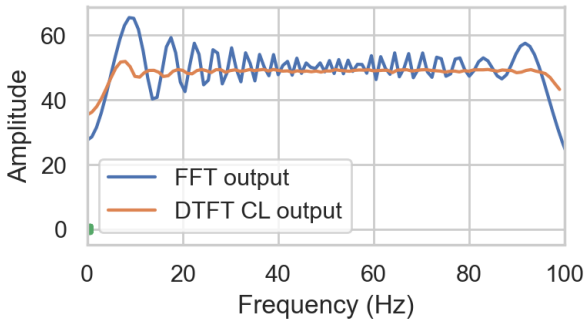
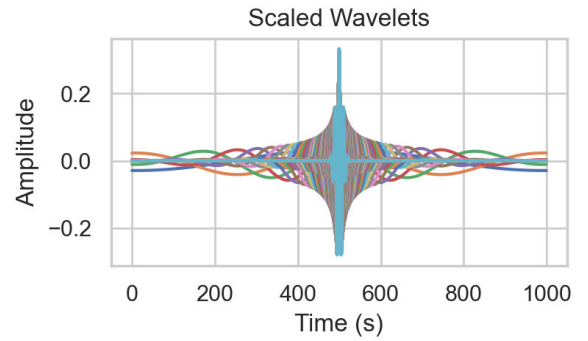
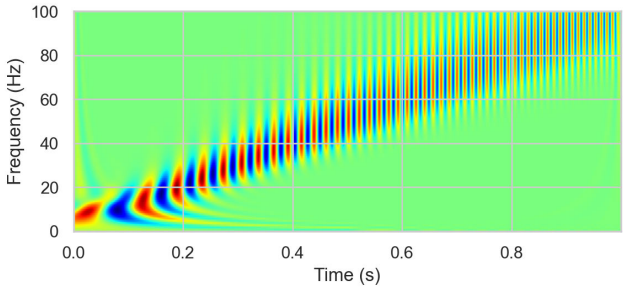


FIGURE 20. DTFT-based CL layer output for $N_F = 100$ and $N_K = 100$.



(a) Scaled Morlet wavelets with 100 scales each of 1000 samples.



(b) CWT-based CL layer output for $N_F = 100$ and $N_K = 1000$.

FIGURE 21. Visualization of the 2D spectrogram produced by the CL STFT layer.

The development and testing steps of a DSP-based CL are:

- Using the Python Scipy and Pywavelets packages to design the DSP algorithm and compute the filter taps.
- Instantiate a single Conv1D layer and assign the layer hyperparameters as instructed by Section III.
- Assign the computed filter weights to the Conv1D layer kernels and disable layer training.
- Apply a test signal to the CL and visualize its output.

A linear chirp signal is generated and applied to the CL-based DSP layer to test the developed algorithms. Figure 16 illustrates the testing signal of a 1 s duration and a linear frequency range from 1 to 100 Hz.

A. IMPLEMENTATION AND VALIDATION OF THE CL FIR FILTERS

The following Python script shows how to develop a CL-based FIR filter. Figure 17a demonstrates the frequency response of an example FIR BPF of order 99 with a passband between 40 and 60 Hz. Figure 17b illustrates the CL output for the chirp testing input in the time domain. As expected, the CL layer works as a BPF and passes frequencies from 40 to 60 Hz. Figure 18 depicts the application of the CL FIR filtering for noise removal. An AWGN of 10 dB SNR is added to the chirp signal in Figure 16 and applied to a CL FIR LPF with a 100 Hz cutoff frequency and order $N = 99$. As evidenced by the output, the CL filter significantly smoothens the input noisy signal by filtering out high-frequency noise.

```
# Import Python packages
import pywt
import numpy as np
import tensorflow as tf
from scipy.signal import freqz, stft, firwin
from keras.layers import Conv1D
import matplotlib.pyplot as plt
# Generate test signal
fs = 1000
t = np.arange(0,1,1/fs)
sig = scipy.signal.chirp(t, 1, 1, 100)
#  Design the Filter Filter
from scipy.signal import firwin, freqz
H = firwin(100, [40, 60], pass_zero=False,fs=fs)
f, a = freqz(H)
# Instantiate the Conv1D layer and assign weights
layer = Conv1D(1, H.shape[0], padding='same',
    use_bias=False, trainable=False)
inputs = tf.convert_to_tensor(np.expand_dims(sig
    ,(0,-1)), np.float32)
outputs = layer(inputs)
layer.set_weights([np.expand_dims(H,(0,-1)).
    swapaxes(0,1).swapaxes(1,2)])
# Apply the test signal to the CL and plot output
layer_out = np.squeeze(layer.call(inputs).numpy())
    .transpose()
plt.plot(t,layer_out)
plt.show()
```

```
# Design the Filter bank
conv1d_filters = 10
conv1d_kernel = 100
bw = 0.1*fs/(conv1d_filters)
H = np.ndarray(shape=(conv1d_filters,conv1d_kernel
    ))
for i in range(conv1d_filters):
    H[i] = firwin(conv1d_kernel, [bw*i+0.01, bw*(i
    +1)-0.01], pass_zero=False,fs=fs)
    f, a = freqz(H[i])
# Instantiate the Conv1D layer and assign weights
layer = Conv1D(H.shape[0], H.shape[1], padding='
    same',use_bias=False, trainable=False)
inputs = tf.convert_to_tensor(np.expand_dims(sig
    ,(0,-1)), np.float32)
outputs = layer(inputs)
layer.set_weights([np.expand_dims(H,axis=-1).
    swapaxes(0,1).swapaxes(1,2)])
# Apply the test signal to the CL and plot output
layer_out = np.squeeze(layer.call(inputs).numpy())
    .transpose()
plt.imshow(layer_out,cmap='jet',aspect='auto',
    origin='lower')
plt.show()
```

B. IMPLEMENTATION AND VALIDATION OF THE STFT CL

The following Python script demonstrates how to develop a CL-based STFT algorithm. Figure 19a demonstrates the frequency response of an example FIR filter bank of 10 filters each of order 99 Figure 19b illustrates the time-frequency output of the CL with $N_F = 10$ and $N_K = 100$ for the

chirp testing input. As expected, the CL layer produces an STFT-equivalent spectrogram image that a 2D DNN can handle.

C. IMPLEMENTATION AND VALIDATION OF THE DTFT CL

The following Python script depicts how to develop a CL-based DTFT algorithm. Figure 20 illustrates the FFT-based frequency spectral and spectral output of the CL with $N_F = 100$ and $N_K = 100$ for the chirp testing input. The CL layer produces a DTFT-equivalent spectrum of the input signal.

```
# Design the Filter bank
conv1d_filters = 100
conv1d_kernel = 100
bw = 0.1*fs/(conv1d_filters)
H = np.ndarray(shape=(conv1d_filters,conv1d_kernel
    ))
for i in range(conv1d_filters):
    H[i] = firwin(conv1d_kernel, [bw*i+0.01, bw*(i
    +1)-0.01], pass_zero=False,fs=fs)
    f, a = freqz(H[i])
# Instantiate the Conv1D layer and assign weights
layer = Conv1D(H.shape[0], H.shape[1], padding='
    same',use_bias=False, trainable=False)
layer2 = GlobalMaxPooling1D()
inputs = tf.convert_to_tensor(np.expand_dims(sig
    ,(0,-1)), np.float32)
outputs = layer2(layer(inputs))
layer.set_weights([np.expand_dims(H,axis=-1).
    swapaxes(0,1).swapaxes(1,2)])
# Apply the test signal to the CL and plot output
f, a = freqz(sig)
plt.plot((f/(2*np.pi))*fs, np.abs(a))
plt.plot(fs/20*np.squeeze(layer2.call(layer.call(
    inputs))))
plt.legend(['FFT output', 'DFT CL output'])
plt.show()
```

```
# Generate scaled mother wavelets (Morlet)
wavelet = pywt.ContinuousWavelet('morl')
freqs = np.linspace(1,100,100)/(fs)
scales = pywt.frequency2scale(wavelet,freqs)
mother_wavelet = lambda z : np.exp(-z*z/2)*np.cos
    (5*z)
def morl(n,scale):
    x = np.linspace(-n/2,n/2,n)
    return mother_wavelet(x/scale)/np.sqrt(scale)
wvlt_len = t.shape[0]
H = np.ndarray(shape=(len(scales),wvlt_len))
for i, scale in enumerate(scales):
    H[i]= morl(wvlt_len,scale)
    plt.plot(H[i])
    plt.title('Scaled Wavelets')
# Instantiate the Conv1D layer and assign weights
layer = Conv1D(H.shape[0], H.shape[1], padding='
    same',use_bias=False, trainable=False)
inputs = tf.convert_to_tensor(np.expand_dims(sig
    ,(0,-1)), np.float32)
outputs = layer(inputs)
layer.set_weights([np.expand_dims(H,axis=-1).
    swapaxes(0,1).swapaxes(1,2)])
# Apply the test signal to the CL and plot output
layer_out = np.squeeze(layer.call(inputs).numpy())
    .transpose()
plt.imshow(layer_out,cmap='jet',aspect=3,origin='
    lower')
plt.show()
```

D. IMPLEMENTATION AND VALIDATION OF THE CWT CL

The following Python script illustrates how to develop a CL-based CWT algorithm. Figure 21a demonstrates an example of scaled Morlet wavelets with 100 scales, each of 1000 samples. Figure 17b illustrates the time-frequency output of the CL with $N_F = 100$ and $N_K = 1000$ for the chirp testing input. As expected, the CL layer produces a CWT scalogram image that a 2D DNN can handle.

REFERENCES

[1] J. Gu, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, May 2018.
[2] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2, 1989, pp. 1–9.
[3] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022.
[4] S. A. Alasadi and W. S. Bhaya, "Review of data preprocessing techniques in data mining," *J. Eng. Appl. Sci.*, vol. 12, no. 16, pp. 4102–4107, 2017.
[5] C. Fan, M. Chen, X. Wang, J. Wang, and B. Huang, "A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data," *Frontiers Energy Res.*, vol. 9, Mar. 2021, Art. no. 652801.
[6] T. A. Alghamdi and N. Javaid, "A survey of preprocessing methods used for analysis of big data originated from smart grids," *IEEE Access*, vol. 10, pp. 29149–29171, 2022.
[7] (2022). *TensorFlow Lite: ML for Mobile and Edge Devices*. Accessed: Feb. 20, 2023. [Online]. Available: https://www.tensorflow.org/lite/
[8] (2022). *Quantization in Pytorch*. Accessed: Feb. 20, 2023. [Online]. Available: https://pytorch.org/docs/stable/quantization.html
[9] M. A. Little, *Machine Learning for Signal Processing: Data Science, Algorithms, and Computational Statistics*. London, U.K.: Oxford Univ. Press, 2019.
[10] G. Sharma, K. Umapathy, and S. Krishnan, "Trends in audio signal feature extraction methods," *Appl. Acoust.*, vol. 158, Jan. 2020, Art. no. 107020.
[11] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 2, pp. 56–70, May 2020.
[12] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Elect. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014.
[13] M. M. Richter, S. Paul, V. Këpuska, and M. Silaghi, *Signal Processing and Machine Learning With Applications*. Springer, 2022. [Online]. Available: https://books.google.com.sa/books?id=VuD8vQAACAAJ
[14] A. Papandreou-Suppappola, *Applications in Time-Frequency Signal Processing*. Boca Raton, FL, USA: CRC Press, 2018.
[15] T. Wang, C. Lu, Y. Sun, M. Yang, C. Liu, and C. Ou, "Automatic ECG classification using continuous wavelet transform and convolutional neural network," *Entropy*, vol. 23, no. 1, p. 119, Jan. 2021.
[16] J. Huang, B. Chen, B. Yao, and W. He, "ECG arrhythmia classification using STFT-based spectrogram and convolutional neural network," *IEEE Access*, vol. 7, pp. 92871–92880, 2019.
[17] G. Yu, "A concentrated time–frequency analysis tool for bearing fault diagnosis," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 2, pp. 371–381, Feb. 2019.
[18] R. X. Chen, X. Huang, L. X. Yang, X. Y. Xu, X. Zhang, and Y. Zhang, "Intelligent fault diagnosis method of planetary gearboxes based on convolution neural network and discrete wavelet transform," *Comput. Ind.*, vol. 106, pp. 48–59, Apr. 2019.
[19] W. Deng, S. Zhang, H. Zhao, and X. Yang, "A novel fault diagnosis method based on integrating empirical wavelet transform and fuzzy entropy for motor bearing," *IEEE Access*, vol. 6, pp. 35042–35056, 2018.
[20] M. H. Soni, N. Shah, and H. A. Patil, "Time-frequency masking-based speech enhancement using generative adversarial network," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 5039–5043.
[21] M. Kimlyk and S. Umnyashkin, "Image denoising using discrete wavelet transform and edge information," in *Proc. IEEE Conf. Russian Young Researchers Electr. Electron. Eng. (EIConRus)*, Jan. 2018, pp. 1823–1825.
[22] S. Saxena, R. Jais, and M. K. Hota, "Removal of powerline interference from ECG signal using FIR, IIR, DWT and NLMS adaptive filter," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2019, pp. 12–16.

[23] T. Tuncer, S. Dogan, and A. Subasi, "Surface EMG signal classification using ternary pattern and discrete wavelet transform based feature extraction for hand movement recognition," *Biomed. Signal Process. Control*, vol. 58, Apr. 2020, Art. no. 101872.
[24] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Trans. Signal Inf. Process.*, vol. 3, no. 1, p. e2, 2014.
[25] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, Apr. 2017.
[26] S. Pouyanfar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–36, Sep. 2018.
[27] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at Microsoft," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 8604–8608.
[28] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 2, pp. 206–219, Apr. 2019.
[29] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19143–19165, 2019.
[30] D. J. Hemanth and V. V. Estrela, *Deep Learning for Image Processing Applications*, vol. 31. Amsterdam, The Netherlands: IOS Press, 2017.
[31] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," 2017, *arXiv:1707.05589*.
[32] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 604–624, Feb. 2021.
[33] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.
[34] L. Deng and Y. Liu, *Deep Learning in Natural Language Processing*. Cham, Switzerland: Springer, 2018.
[35] G. Litjens, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, Dec. 2017.
[36] P. Lang, X. Fu, M. Martorella, J. Dong, R. Qin, X. Meng, and M. Xie, "A comprehensive survey of machine learning applied to radar signal processing," 2020, *arXiv:2009.13702*.
[37] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 4, pp. 648–664, Dec. 2018.
[38] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.
[39] F.-L. Fan, J. Xiong, M. Li, and G. Wang, "On interpretability of artificial neural networks: A survey," *IEEE Trans. Radiat. Plasma Med. Sci.*, vol. 5, no. 6, pp. 741–760, Nov. 2021.
[40] R. S. Srinivasamurthy, "Understanding 1D convolutional neural networks using multiclass time-varying signalss," Ph.D. dissertation, Dept. Comput. Eng., Clemson Univ., Clemson, SC, USA, 2018.
[41] Z. Jia, C. Bao, and K. Ma, "Exploring frequency domain interpretation of convolutional neural networks," 2019, *arXiv:1911.12044*.
[42] L. Stankovic and D. Mandic, "Convolutional neural networks demystified: A matched filtering perspective based tutorial," 2021, *arXiv:2108.11663*.
[43] D. Yu and L. Deng, "Deep learning and its applications to signal and information processing [exploratory DSP]," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 145–154, Jan. 2011.
[44] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, Mar. 2021.
[45] M. M. Farag, "Matched filter interpretation of CNN classifiers with application to HAR," *Sensors*, vol. 22, no. 20, p. 8060, Oct. 2022.
[46] M. M. Farag, "A self-contained STFT CNN for ECG classification and arrhythmia detection at the edge," *IEEE Access*, vol. 10, pp. 94469–94486, 2022.
[47] M. M. Farag, "A tiny matched filter-based CNN for inter-patient ECG classification and arrhythmia detection at the edge," *Sensors*, vol. 23, no. 3, p. 1365, Jan. 2023.
[48] L. Guo, L. Wang, J. Dang, Z. Liu, and H. Guan, "Exploration of complementary features for speech emotion recognition based on kernel extreme learning machine," *IEEE Access*, vol. 7, pp. 75798–75809, 2019.
[49] Z. Liang, M. Tao, L. Wang, J. Su, and X. Yang, "Automatic modulation recognition based on adaptive attention mechanism and ResNeXt WSL model," *IEEE Commun. Lett.*, vol. 25, no. 9, pp. 2953–2957, Sep. 2021.
[50] Q. Zhang, Z. Xu, and P. Zhang, "Modulation recognition using wavelet-assisted convolutional neural network," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Oct. 2018, pp. 100–104.
[51] Z. Sun, L. Zhou, and W. Wang, "Learning time-frequency analysis in wireless sensor networks," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3388–3396, Oct. 2018.
[52] I. Dey and S. Siddiqui, "Wavelet transform for signal processing in internet-of-Things (IoT)," in *Wavelet Theory*. London, U.K.: IntechOpen, 2021.
[53] A. V. Oppenheim, J. R. Buck, and R. W. Schafer, *Discrete-Time Signal Processing*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.
[54] R. E. Ziemer and W. H. Tranter, *Principles of Communications*. Hoboken, NJ, USA: Wiley, 2014.
[55] F. Chollet, *Deep learning With Python*. New York, NY, USA: Simon and Schuster, 2021.
[56] A. Mertins and D. A. Mertins, *Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications*. New York, NY, USA: Wiley, 1999.
[57] S. Zhang, S. Zhang, B. Wang, and T. G. Habetler, "Deep learning algorithms for bearing fault diagnostics—A comprehensive review," *IEEE Access*, vol. 8, pp. 29857–29881, 2020.
[58] W. Caesarendra and T. Tjahjowidodo, "A review of feature extraction methods in vibration-based condition monitoring and its application for degradation trend estimation of low-speed slew bearing," *Machines*, vol. 5, no. 4, p. 21, 2017.
[59] D. Wu, J. Wang, H. Wang, H. Liu, L. Lai, T. He, and T. Xie, "An automatic bearing fault diagnosis method based on characteristics frequency ratio," *Sensors*, vol. 20, no. 5, p. 1519, Mar. 2020.
[60] S. Kim, D. An, and J.-H. Choi, "Diagnostics 101: A tutorial for fault diagnostics of rolling element bearing using envelope analysis in MATLAB," *Appl. Sci.*, vol. 10, no. 20, p. 7302, Oct. 2020.
[61] F. Immovilli, C. Bianchini, M. Cocconcelli, A. Bellini, and R. Rubini, "Bearing fault model for induction motor with externally induced vibration," *IEEE Trans. Ind. Electron.*, vol. 60, no. 8, pp. 3408–3418, Aug. 2013.
[62] E. Bechhoefer. (2013). *Condition Based Maintenance Fault Database for Testing of Diagnostic and Prognostics Algorithms*. Accessed: Feb. 23, 2023. [Online]. Available: https://www.mfpt.org/fault-data-sets/
[63] C. Jarne, "A heuristic approach to obtain signal envelope with a simple software implementation," 2017, *arXiv:1703.06812*.

MOHAMMED M. FARAG (Member, IEEE) was born in Egypt. He received the B.Sc. and M.Sc. degrees in electrical engineering from Alexandria University, Egypt, in 2003 and 2007, respectively, and the Ph.D. degree in computer engineering from Virginia Polytechnic Institute and State University, in 2012. He was a Teaching Assistant with Alexandria University, from 2003 to 2009. From 2009 to 2013, he was a Research Assistant with Virginia Tech. From 2013 to 2017, he joined the Electrical Engineering Department, Alexandria University, as an Assistant Professor. Since 2018, he has been with the Electrical Engineering Department, College of Engineering, King Faisal University, Saudi Arabia. His research interests include machine learning, signal processing, VLSI design, and cyber-physical security.