ECEI Spectrogram Denoising and Feature Extraction using a Multi-Wavelet Convolutional Neural Network Model

By

NAVEED OROUMCHIAN

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Neville C. Luhmann Jr. (Committee Chair)

Anh-Vu Pham

William Putnam

Committee in Charge

Year of Degree 2023

Acknowledgments

This work is supported by U.S. Department of Energy grant number DE-FG02-99ER54531 and DE-SC0023500

This Work is dedicated to my family, partner, friends, and all those who were patient with me.

A special Thanks to Ranil, Braden and Brianna for lending me their friendship and couches.

Abstract

The advent of high-precision plasma diagnostics tools has created vast amounts of data to be processed and, as a result, increased demand for tools that automate data analysis. One common task in analyzing plasma data is processing spectrogram data to find structures that correspond to physical processes that have occurred, one such example being the toroidal Alfven eigenmode. Due to the extreme environments of the plasma, the data collected are often noisy, thus making automatic detection of structures in the data difficult.

This thesis presents a pipeline that takes in raw data from a channel of an Electron Cyclotron Emission Imaging (ECEI) system, which then detects and processes any excited Alfven eigenmodes. Each detected mode is tagged and individually filtered out in the pipeline. The main feature of the pipeline is a deep denoising model. The deep denoising model is created to combat the noise in the spectrograms and enhance the detected signal. A deep denoising architecture is chosen because of its strong ability to denoise images without blurring, unlike traditional denoising models. The architecture used is a Multi-Wavelet Convolutional Neural Network which was then trained using synthetic data. Synthetic data were used due to limited availability of clean measured ECEI data. When applied to real ECEI data, the denoising model achieves significant background denoising, at the cost of occasional hallucinated or spurious structures.

The pipeline, in total, has nine steps to process and filter the raw ECEI data. The steps of the pipeline are: time domain low-pass filtering, short-time Fourier transform, spectrogram denoising, thresholding, morphological thinning, trace clustering, trace separation, filter mask creation, mask filtering, and inverse short-time Fourier transform. The combination of the

operations in the pipeline allows for the automatic detection and filtering of signals in the spectrogram with a strong rejection of noise.

Table of Contents

I. Introduction

Investigating plasma physics phenomena typically requires manual labor in sifting through the vast amounts of data modern plasma diagnostics tools generate. Tools that automate aspects of the research process are desirable to allow the user to sift through the data quickly. One aspect of study in plasma physics is excited "modes" that occur in the plasma for various physical reasons. In studying plasma behavior in tokamaks, one class of signals of interest are toroidal Alfven eigenmodes[1]. Typically, the modes are detected using spectrograms generated from data collected from plasma diagnostic tools such as Electron Cyclotron Emission Imaging (ECEI)[2]. In the DIII-I tokamak, the modes are measured using an Electron Cyclotron Emission Imaging system [3] that measures 22 poloidal and 8 radial channels, creating a 22x8 pixel image at each moment with a time resolution of one microsecond. Each image represents plasma temperature in a cross-sectional patch of the plasma column. Individual excited modes in the tokamak data can be found by investigating the channel data from the ECEI measurements. Due to the non-stationary nature of the signals, a time-frequency analysis can be used for signal detection. Inherent to the plasma environment, the recorded data are plagued by corruption of the data. As a result, techniques for detecting signal events should have a strong level of noise rejection built into them.

The field of time-frequency analysis is rich with many different tactics to approach the problem of signal detection. The usual process in signal detection in the time-frequency domain is divided into filtering, tracking, and enhancement. Traditionally, Wiener filtering[4] and spectral subtraction[5] can generate good denoised signals, which can be extracted by tracking end points of detected structures. However, traditional methods perform poorly in noisy environments with low Signal-to-Noise Ratio (SNR) and non-stationary random noise. Many other modern

techniques can be used to extract signals, such as empirical mode decomposition (EMD)[6], blind source separation [7], [8], independent component analysis [9], Maximum Likelihood estimators [10], [11], particle filter tracking [12], or gaussian mixture probability hypothesis density filter tracking [13].

Another method of signal detection is to convert the signals into the time-frequency domain in the form of spectrogram images and directly process the images. The spectrograms can visually show the signals' time and frequency characteristics, making it easier to distinguish between signals. Additionally, image processing techniques can enhance and extract critical features in the spectrogram. General techniques include noise filters, image reconstruction, segmentation, and morphological operations. Some examples of these image-processing tools applied to spectrogram signal extraction include leveraging mathematical morphology [14], edge detectors [15], ridge detectors [16], and active contour methods [17]. A benefit of directly processing spectrogram images is that after the enhancement and detection of desired signals in the time-frequency domain, they can be reconstructed back into the time domain resulting in the signal of interest in the time domain.

With the vast amounts of data collected, modern machine-learning methods can be used to achieve significant results. If large amounts of labeled data corresponding to signals of interest in the spectrogram images exist, then supervised machine learning can be used to detect and classify the spectrogram images. Convolutional neural networks (CNN) have been used to classify signals in a variety of fields, from marine acoustics [18] to ECG processing [19]. Deep recurrent neural networks (RNN), a type of neural network architecture designed to handle time-series data, have also been used to detect signals from time-series data directly. A novel RNN

architecture [20] has been used to classify Alfven eigenmodes from ECE diagnostics. However, the downside of supervised machine learning is that it requires manually labeled data.

Machine learning, more specifically deep learning, is also used in denoising or enhancing images and signals. Deep learning models trained using noisy images and their corresponding clean images can learn how to remove noise from the original noisy images. Models can also be trained to sharpen images. Many different architectures are used for enhancement; for example, a standard CNN was used for spectrogram enhancement using a multi-metric training approach [21]. More advanced methods use deep autoencoders' compressing abilities to learn the data's latent representations and use that to separate noise from inputted data [22].

This work presents a pipeline to detect and separate individual Alfven eigenmodes from ECEI data. The pipeline uses a mix of image processing, filtering, clustering, and deep denoising techniques to achieve the task of signal detection. The pipeline takes one channel of ECEI data in the form of voltage versus time and outputs all detect modes in the form of voltage versus time. The pipeline can be applied to all channels to create a 2D visualization of the mode. The pipeline uses minimal parameters, and the deep denoising model is trained using only synthetic data. Synthetic data is utilized as it is not practically feasible to manually generate a considerable quantity of clean data samples, which are essential for training neural networks.

The pipeline has nine steps: time domain low-pass filtering, short-time Fourier transform, spectrogram denoising, thresholding, morphological thinning, trace clustering, trace separation, filter mask creation, mask filtering, and inverse short-time Fourier transform. The novel part of this pipeline is the Spectrogram denoising using a multi-level wavelet convolutional neural network to remove different types of noise from the spectrogram and enhance the structure in the spectrogram.

II. Image Denoising and Restoration

Image restoration or denoising is one of the fundamental problems in computer vision and has been deeply studied and remains an active field. Generally, noise is injected into the final image due to the influence of the environment, transmission, measurement, or other factors. The noisy images can significantly hamper the analysis of the images, especially when the SNR is low. Thus, it is of great interest to separate or reconstruct the true image from the noisy image.

The problem of image denoising can be formulated as follows:

$$y = x + n$$

where y is the noisy image, x is the true uncorrupted image, and n is the added noise. The noise can take on any form but is usually considered a Gaussian. The goal becomes to solve for x given y which is an inverse problem and thus many possible solutions exist. Additional challenges for denoising are that edges should be preserved without blurring, flat areas should be smooth, textures should be preserved, and new artifacts should not be generated [23]. Image denoising techniques can be classified into two categories: spatial methods and transform methods.

II.a. Spatial Filters

The first class of traditional filters is spatial filters. Spatial filters act as low-pass filters on image patches taking advantage of the fact that noise is usually distributed over higher frequency ranges. To the extent of lowering noise in image patches, spatial filters are reasonable. However, they come at the cost of blurring, which means losing information in higher frequencies, such as edges and textures.

Spatial filters modify a pixel value using information in the local region around the specified pixel. The spatial filters can be further divided into linear and nonlinear filters.

A spatial filter consists of a defined neighborhood and the operation that occurs centered at that pixel. For linear filters, mathematically, they can be formulated as the convolution of image $f(x, y)$ with impulse response $h(x, y)$ to give the output image $g(x, y)$ at each pixel. The impulse response $h(x, y)$ is also called the kernel. Usually, a square kernel with size $R \times R$ is used but other shapes could also be used. Every pixel in the filtered output image is given by the relation

$$g(x, y) = \sum_i \sum_j f(i, j) \cdot h(x - i, y - j) \quad [24]$$

One primary example of a linear spatial filter is the mean filter which takes the average neighborhood pixel value for each pixel to create the filtered image. This class of filter can reduce gaussian noise at the cost of over-smoothening sharp features. Additionally, linear spatial filters do not handle impulse-like noises. Nonlinear filters can be used to achieve better filtering while balancing noise smoothening and detail retention. Bilateral filtering is a popular nonlinear spatial filter used for denoising images while preserving edges and details [25]. The bilateral

filter works by determining the pixel value of the filtered image using a nonlinear combination of neighboring pixel values.

Another method to improve denoising is to formulate the problem as an energy minimizing problem where a noisy image has a high energy. These methods are called "Variational" denoising methods. The clean image $\hat{x}$ is defined as

$$\hat{x} = \arg\min E(x)$$

$$E(x) = F(x) + \lambda R(x)$$

where $E(x)$ is the "energy" of the image which is the sum of the fidelity term $F(x)$ and regularization term $R(x)$ with λ being the regularization parameter. The fidelity term denotes the difference between a clean image and a noisy image, while the regularization term is an image prior, which narrows the solution space in the optimization to the type of desired solutions, such as edge or texture preservation. In variational denoising methods, the goal is to design the image prior $R(x)$ to achieve the desired effect, some examples of image priors are gradient priors, non-local self-similarity (NSS) priors, sparse priors, and low-rank priors. [23]

One popular and successful variational method is called "Total Variational" (TV) denoising [26]. TV methods aim to achieve image detail preservation and edge sharpness by using an image prior based on the local gradient of the image. TV regularization is based on the fact that images tend to be locally smooth, and that intensity gradually varies. The gradient regularization thus penalizes sharp changes in pixel intensity which may be due to noise.

II.b. Transform Based Filters

As image-denoising techniques matured, new methods were created to transform the image into another basis that allows filtering while maintaining higher-order details. A basis is chosen or created so that when the image is transformed into the new domain, differentiating between noisy and clean images becomes substantially more straightforward. Denoising procedures are applied in the new domain, and then the result is transformed into the original domain to obtain a clean image. The transformation based denoising started with Fourier transform and developed into cosine transforms, wavelet transforms, and other sophisticated transforms. Transform-based filtering methods can be categorized based on the basis functions they use, and whether they are data-adaptive or non-data-adaptive [27]. One of the most widely used and successful transform-based models is Block Matching and 3D filtering (BM3D) [28]. BM3D is a two-stage filter that extends the idea of the non-local means (NLM) variational denoising technique [29]. First, similar regions are grouped using a block-matching algorithm and then arranged into 3D data structures called "group". A 3D linear transformation transforms each group, then a transform domain shrinkage such as a Wiener filter is applied, and finally, the group is inverted back into the original domain. Groups are returned into 2D image segments by weight-averaging overlapping segments. This procedure ensures noise is filtered, but structures are not blurred out.

In the class of data-adaptive transform-based filters, many different statistical and machine-learning methods have been used. Some classic methods involve using the Principal Component Analysis (PCA) [30] or Independent Component Analysis (ICA) [31] to identify the main features that comprise the image and separate it from the noise. Other techniques involve learning a global dictionary of features using the K-SVD algorithm from a training set to use in denoising [32] or using singular value decomposition to obtain local and global basis functions to

denoise an image [33]. These methods generally perform poorer than the BM3D algorithm at general image denoising. Newer deep learning-based methods are now on-par or surpass the BM3D algorithm.

II.c. Deep Denoising

Deep learning methods aim to solve the task of image denoising in an end-to-end fashion by learning image priors from a clean image dataset. Neural networks for image processing tasks achieve the best performance using convolutional layers, which perform a trainable convolution operation on the input data to the layer. Neural networks with convolutional layers are called CNNs; stacked convolution layers learn convolutional filters to extract hierarchical structures in the inputs. In denoising data, these models are often dubbed "Denoising Convolutional Neural Networks" (DnCNN).

Another method is the use of deep autoencoders. Deep autoencoders consist of two components: encoder and decoder. The encoder compresses the input feature down to a lower dimensional latent space representation by successively down-sampling between layers. The decoder then learns how to reconstruct the input from the latent representations and, in the process, up-sampling back into the original dimensions. Figure 1 shows the general architecture of an autoencoder.

Deep autoencoders are great at nonlinear dimensionality reductions to learn complex features from high-dimensional inputs such as images. Deep autoencoders are trained by minimizing the loss between the reconstructed and original images. For image processing tasks, the performance of convolutional layers can be used to improve the compression and reconstruction of the

autoencoders. One successful implementation of a deep learning model using autoencoder architecture with a convolutional layer for denoising is called the Residual Encoder-Decoder network (RED-net). RED-net achieves performance by introducing residual learning, skip-connections and batch normalizations to improve learning rate and learning convergence [34].

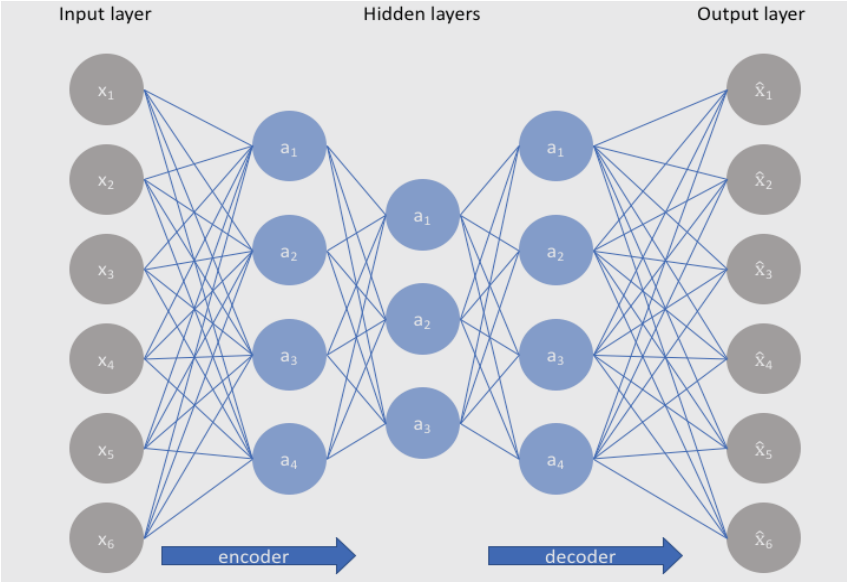

Figure 1 Example of an autoencoder [52]

The task of denoising images can be achieved by using denoising autoencoders [35]. Denoising autoencoders use the same architecture as the normal autoencoder but use a different training procedure. Instead of feeding the true images as inputs, a corrupted version of the input is fed through. Then, the reconstructed image is compared with the true uncorrupted image. By minimizing the loss between the reconstructed image resulting from a corrupted input with the original uncorrupted image, the network learns the latent representation of the possible true structures in the images. The resulting model can then, in essence, denoise and enhance given noisy and corrupted images.

Many modifications have been made to deep autoencoders to improve their reconstruction and learning convergence performance. One successful architecture is called the U-Net [22], a Fully Convolutional Network (FCN) autoencoder consisting of contracting and expanding portions. The contracting segment learns the context of the image and consists of groups of convolutional layers with a kernel size of (3x3) followed by (2x2) max pooling operations with a stride of 2 for down-sampling, which doubles the number of features at each time. The convolutional layers are each followed by a ReLU activation function. The expanding partition is almost symmetric to the contracting layer, consisting of the same groups of convolutional networks followed by a (2x2) up-convolution that halves the features. After each up-convolution, features from the corresponding contracting layer are concatenated with up-sampled features. The residual connections aid the model in maintaining the structure of the image fed into it. The final layer is a convolutional layer with a kernel size of 1x1 and a filter number of 1 to reconstruct the grayscale input image. Figure 2 [22] shows the architecture of the U-Net model.

The up-sampling procedure allows the network to learn to not only reconstruct the input image, but to sharpen it as well. The U-Net architecture is used primarily for image segmentation, but can easily be modified for denoising purposes by changing the training method.

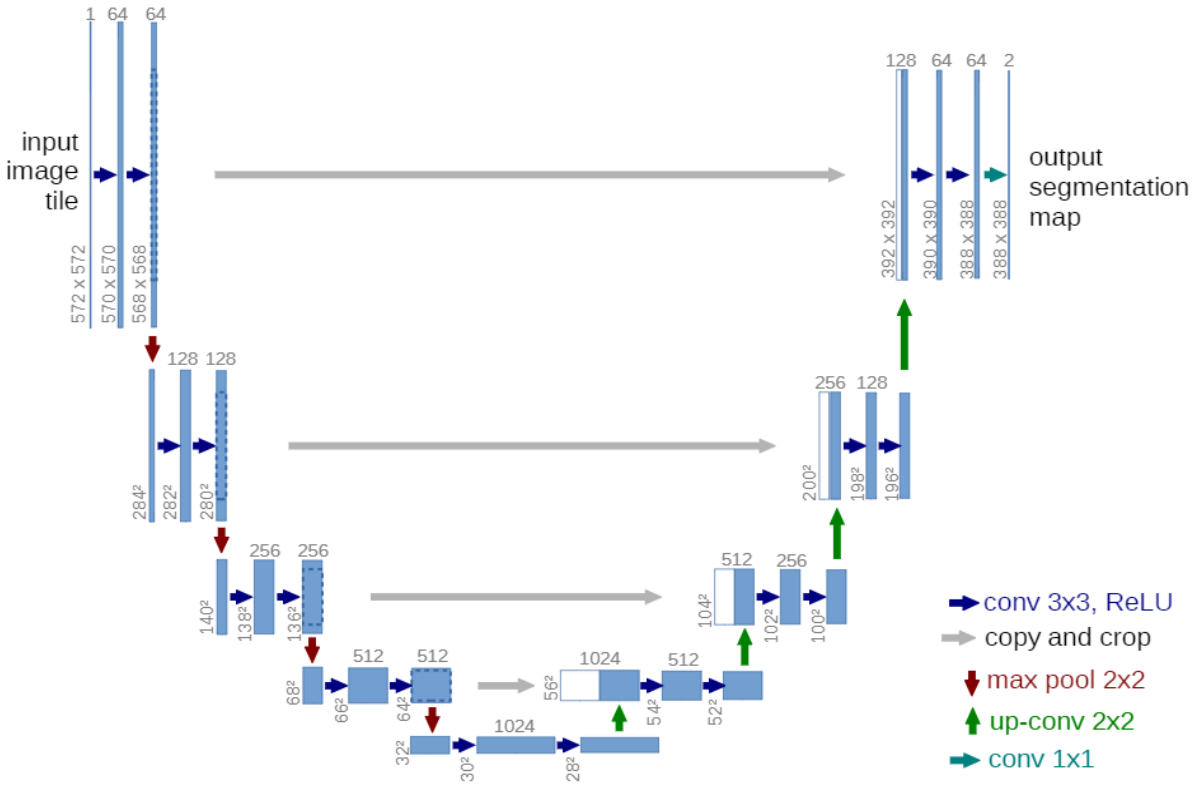

Figure 2 U-Net architecture. Each blue block corresponds to the feature map of the convolutional layers. The blue arrows correspond to the convolutional operations followed by ReLU activation function. The red and green arrows correspond to the max pooling and up [22]

The receptive field of a model, in the context of CNNs, is the area of a patch on the input that affects features further down the network. For image denoising, a sizeable receptive field is desired to capture images' small- and large-scale features. For an FCN without a max pooling layer, to increase the receptive field of the network, either the network depth can be increased, or the number of convolutional filters used can be increased. Both methods increase the computational cost dramatically in training and live deployment. Architectures like the U-Net can achieve large receptive fields without sacrificing too much computation cost by using

dilating or down-sampling layers. However, due to the gridded sampling, the down-sampling procedure comes at the cost of inherent checkerboard patterns. In the dilation step, instead of a standard NxN kernel, the kernel will sample NxN points but with a gap between the sampled points specified by how much down-sampling is desired. The artifacts introduced by dilation through the increasing of the receptive field may lower the SNR of the model. As a result, a balance should be met while increasing the dilation.

A novel model dubbed "multi-level wavelet CNN" (MWCNN) was introduced [36] to achieve a larger receptive field while avoiding the computational cost and artifact generation of the U-Net architecture. The model uses the same architecture as U-Net, but instead of down-sampling using a pooling layer in the contracting portion of the model, they employ a discrete wavelet transform (DWT). The DWT has the benefit of being invertible through an inverse wavelet transform (IWT) and thus does not lose information in the down-sampling. Additionally, wavelet transforms capture spatial and frequency information, which can aid in better representation of the image in feature space.

The DWT for a 2D image consists of four filters, $f_{LL}, f_{LH}, f_{HL}, and\ f_{HH}$, convolved with an image X to result in a decomposition into four sub-banded images $X_{LL}, X_{LH}, X_{HL}, and\ X_{HH}$. The filters correspond to the horizontal and vertical filtering by high-pass and lowpass filters; Fig. 3 shows the DWT decomposition. The filters have fixed parameters based on the wavelet basis and a convolutional stride 2 used in the transformation. An important property of the wavelet's filters is that they are orthogonal to each other. In general, the DWT operation can be written as:

$$X_{LL} = (f_{LL} \otimes X) \downarrow_2 , X_{LH} = (f_{LH} \otimes X) \downarrow_2 , X_{HL} = (f_{HL} \otimes X) \downarrow_2 , X_{HH} = (f_{HL} \otimes X) \downarrow_2$$

where $\downarrow_2$ refers to down sampling by a factor of 2 and $\otimes$ refers to convolution operation.

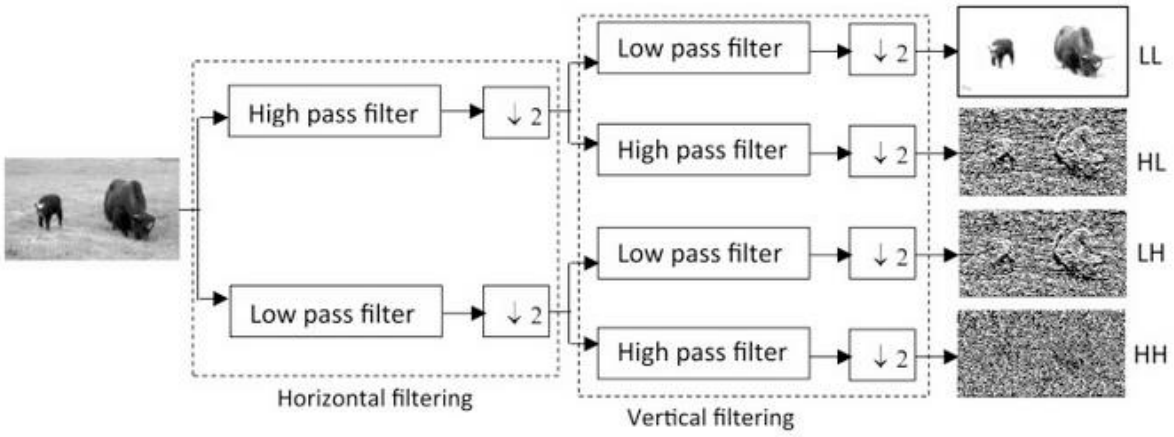

Figure 3 Decomposition of the image by 2D wavelet transform [34]

The original image can be reconstructed from the decomposed images without loss of information using the Inverse wavelet transform (IWT) such that $X = IWT(X_{LL}, X_{LH}, X_{HL}, X_{HH})$. The Haar wavelet [37], one of the simplest and most common wavelets, is used in this work but in theory any other wavelet can be used. The Haar wavelet filters are given by:

$$\mathbf{f}_{LL} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{f}_{LH} = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix},$$
$$\mathbf{f}_{HL} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{f}_{HH} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

After the 2D transform using the Haar wavelets, for each sub-band image the (i, j)-th value is given by:

$$\begin{cases} \mathbf{x}_{LL}(i,j) = \mathbf{x}(2i-1,2j-1) + \mathbf{x}(2i-1,2j) \\ \qquad\qquad +\mathbf{x}(2i,2j-1) + \mathbf{x}(2i,2j) \\ \mathbf{x}_{LH}(i,j) = -\mathbf{x}(2i-1,2j-1) - \mathbf{x}(2i-1,2j) \\ \qquad\qquad +\mathbf{x}(2i,2j-1) + \mathbf{x}(2i,2j) \\ \mathbf{x}_{HL}(i,j) = -\mathbf{x}(2i-1,2j-1) + \mathbf{x}(2i-1,2j) \\ \qquad\qquad -\mathbf{x}(2i,2j-1) + \mathbf{x}(2i,2j) \\ \mathbf{x}_{HH}(i,j) = \mathbf{x}(2i-1,2j-1) - \mathbf{x}(2i-1,2j) \\ \qquad\qquad -\mathbf{x}(2i,2j-1) + \mathbf{x}(2i,2j). \end{cases}$$

Furthermore, the IWT can be written in terms of the sub-banded images as the following:

$$\begin{cases} \mathbf{x}(2i-1,2j-1) = (\mathbf{x}_{LL}(i,j) - \mathbf{x}_{LH}(i,j) \\ \qquad\qquad -\mathbf{x}_{HL}(i,j) + \mathbf{x}_{HH}(i,j))/4, \\ \mathbf{x}(2i-1,2j) = (\mathbf{x}_{LL}(i,j) - \mathbf{x}_{LH}(i,j) \\ \qquad\qquad +\mathbf{x}_{HL}(i,j) - \mathbf{x}_{HH}(i,j))/4, \\ \mathbf{x}(2i-1,2j) = (\mathbf{x}_{LL}(i,j) + \mathbf{x}_{LH}(i,j) \\ \qquad\qquad -\mathbf{x}_{HL}(i,j) - \mathbf{x}_{HH}(i,j))/4, \\ \mathbf{x}(2i,2j) = (\mathbf{x}_{LL}(i,j) + \mathbf{x}_{LH}(i,j) \\ \qquad\qquad +\mathbf{x}_{HL}(i,j) + \mathbf{x}_{HH}(i,j))/4. \end{cases}$$

The DWT can be recursively applied to each sub-banded image for further processing in a multi-level wavelet packet transform (WPT). The IWT is applied symmetrically as the DWT path to reconstruct the original image. The multi-level WPT can be connected to the CNN layers because the wavelet filters could be considered a fully convolutional network (FCN) layer with pre-set weights and no nonlinearity. Thus, multi-level WPT can be extended by having FCN after each DWT. In image processing applications, such as denoising or segmentation, nonlinear operations like thresholding or filling are applied to post-transformed images to further aid in the

image processing task. Convolutional Layers in the multi-level WPT structure act like tailor-made functions for the image-processing tasks learned during the network training.

The benefits of employing learnable structures such as the FCNs are that any potential function can be automatically learned based on the learning phase's requirements. The complete architecture of the FCNs in the multi-level WPT is dubbed the multi-level wavelet CNN (MWCNN) [36]. Figure 4 shows a comparison between the multi-level WPT and the MWCNN. The MWCNN architecture can be modified in many of the same ways that other neural networks are modified, such as adding different types of layers or regularization methods. However, the core idea of the MWCNN is having a convolutional layer between each level of DWT.

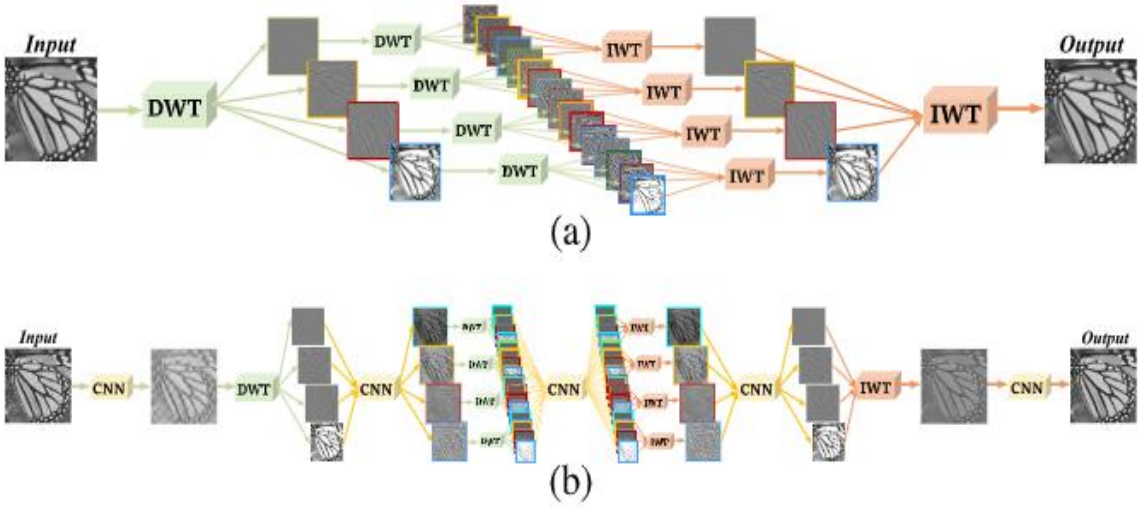

Figure 4 Comparison between Multi-level WPT (a) and a MWCNN (b). It can be seen that the MWCNN is a generalization of the multi-level WPT. [36]

As mentioned before, the MWCNN architecture used for image restoration was based on the U-Net architecture [22]. The main changes from the U-Net Architecture are the wavelet transforms for down-sampling and up-sampling layers and element-wise summation of features between contracting and expanding networks instead of concatenation. After each DWT down-sampling, the four sub-banded images are concatenated and fed to an FCN layer with several convolutional filters to match the next set of convolutional Blocks. Each CNN block consists of a 3-layer of FCN without any pooling layers. Each FCN layer in the block has a 3x3 filter followed by a ReLU activation function. In the expanding network before the IWT up-sampling layer, an FCN is placed with a number of filters to create a feature map that matches the next set of CNN blocks after up-sampling. Figure 5 shows the template for the MWCNN architecture.

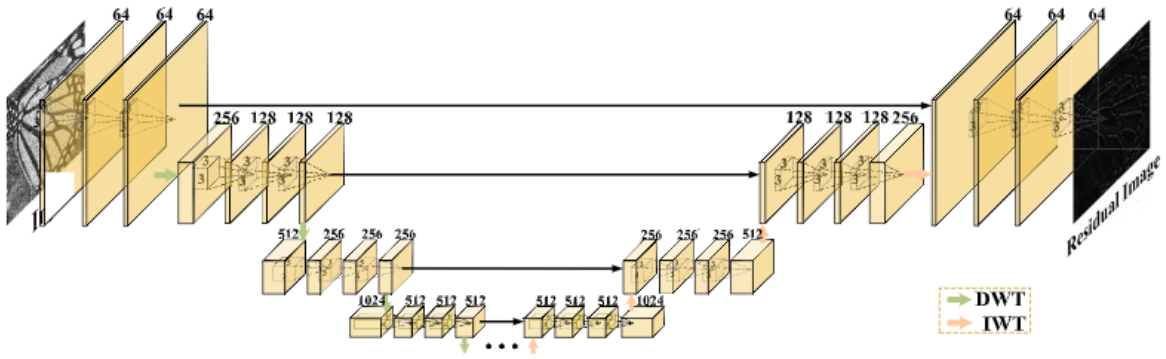

Figure 5 MWCNN architecture [36]

III. Training Deep Denoising Model

Two main methods of training deep denoising models using pairs of clean and corrupted images are to train directly for reconstruction accuracy or to train for residual learning. For deep denoising, pairs of clean ground truth images and noisy corrupted images are used for training the model. As defined before, the input to the model is the noisy image $y_i = x_i + n$ where y_i is the i-*th* noisy image, x_i is the i-*th* ground truth image and n is the noise or corruption added to the clean image. For standard denoising models, the goal is to find a function mapping $f(y_i, \Theta) = x_i$ where Θ are the parameters of the function. For neural networks, this is achieved through the appropriate loss function.

The loss function determines how far the output of a neural network is from the correct answer and is used for updating the parameters of the network. During training, the goal is to minimize the loss function. A widely used loss function is the average mean squared error (MSE) which can be written as

$$\mathcal{L}(\Theta) = \frac{1}{2N}\sum_{i=1}^{N}\|f(y_i, \Theta) - x_i\|^2$$

In some image processing tasks, it is more efficient and worthwhile to learn the residual mapping instead of directly predicting a clean image[38][38]. The residual is defined as noise that has been added to the clean images. The goal is to learn the function mapping $f(y_i, \Theta)$ such that $x_i = y_i - f(y_i, \Theta)$ which can be achieved with a slight modification to the MSE loss function as shown below

$$\mathcal{L}(\Theta) = \frac{1}{2N}\sum_{i=1}^{N}\|f(y_i, \Theta) - (y_i - x_i)\|^2$$

III.a. Generating Training Data Set

With limited available data for training deep neural networks, creating synthetic datasets becomes desirable. Creating a synthetic dataset requires adequate modeling of the real dataset's properties. If features that show up in real data are not modeled, then the trained model may not be able to handle them correctly. For denoising models, first, the ground truth image must be generated and then the corruptions must be added to create the image pairs for training. The corruption added to the clean images should also match similar corruption found in a realistic scenario.

In the task of denoising spectrogram images, generating synthetic data sets is desired; otherwise, manual labor is needed to process and generate the clean data needed for training. Deep learning models need thousands of samples; thus, automatic generation is helpful for training models adequately. Additional benefits to creating synthetic data arise from the nature of the signal we want to detect. In the ideal case, the spectrogram only contains the signal we want and adds no other corruptions, which can be simulated in the synthetic dataset. Then, any arbitrary corruption is added to the clean image. This procedure can force the model to train for only extracting the desired signal in the spectrogram, as the ground truth image only contains exactly the signal we want and ignores everything else. In essence, the denoising model has turned into a signal-extracting model.

III.b. Modeling Alfven Eigenmode Spectrograms

The Alfven eigenmodes are characterized by up or down chirping modes. Multiple Alfven eigenmodes can also be excited close to each other in time, creating clusters of modes in time and frequency. Figure 6 is a spectrogram from a shot from the DIII-D tokamak shot 185787 showing excited toroidal Alfven eigenmodes. The image shows that the excited modes appear as up-chirping curves clustered close to each other. A method to generate structures similar to the actual physics found in the spectrogram and possible corruptions is needed to create a synthetic dataset to match the actual features of data.

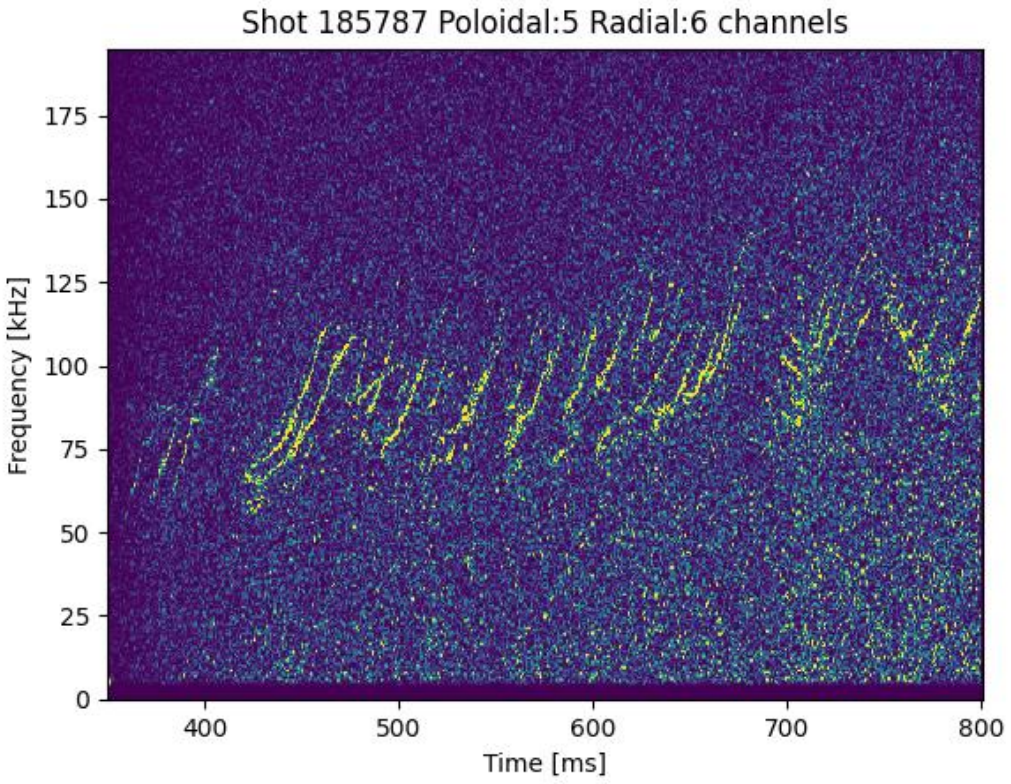

Figure 6 Spectrogram of ECEI data of a region where Alfven Eigenmodes exist from the DIII-D tokamak shot 185787

This work uses a simple approach to model structures generated by Alfven eigenmodes. Simple functions are used as proxies to generate similar-looking structures in the spectrogram. The assumption is that if the structures are similar enough to actual structures, then the model can generalize to detect other continuous curves that can show up. Four functions were used: linear, square, cubic, and logistic. Each function was parameterized to control its length, position, and shape. A random number of curve-generating functions is chosen per spectrogram, with random parameters chosen for each function. The figure (Fig. 7) below shows a sample of generated ground truth spectrograms. Each image has a size of 256 x 256 pixels which, while arbitrary, reduces the model size and, thus, training time. The spectrogram generated is divided into 256 x 256 chunks, fed to the model, and then further processed. The results generated from each chunk are then stitched back together. Chunking the spectrogram into smaller images allows the

processing of large spectrograms while using a relatively modest model size. The chunks should not be smaller than the features we want to process, which are the Alfven eigenmodes.

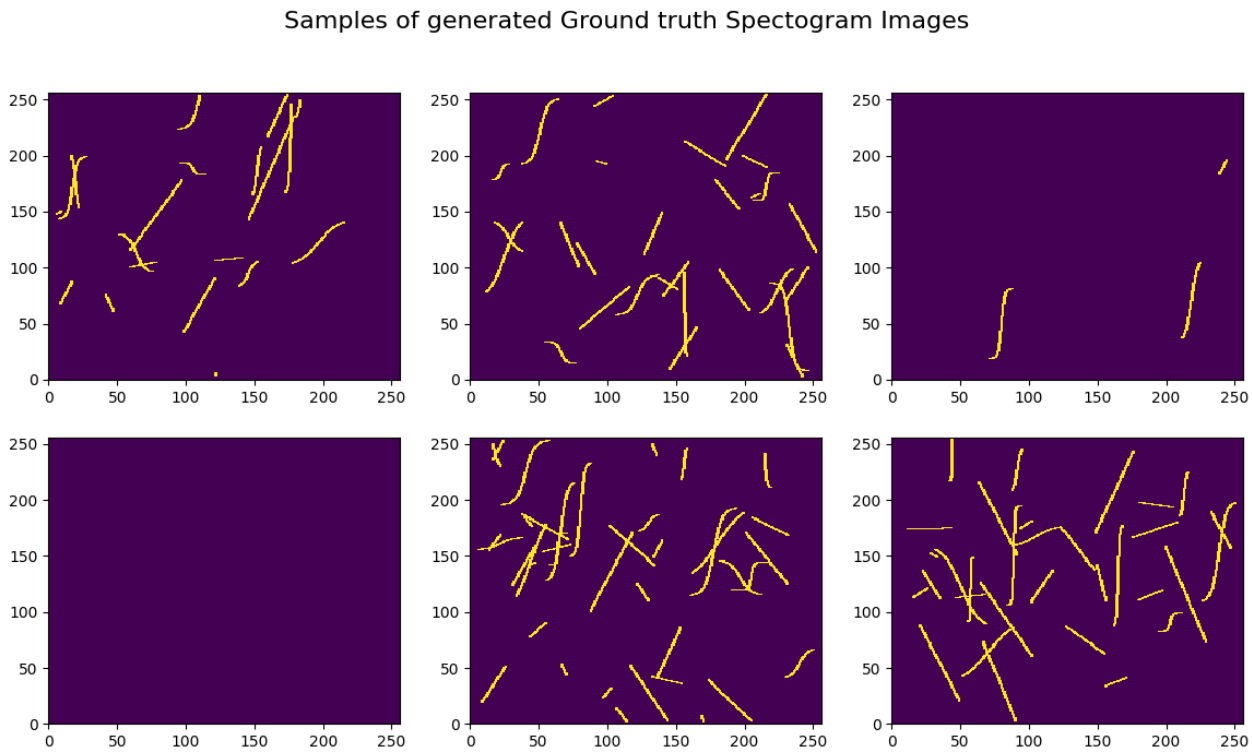

Figure 7 Examples of generated Ground Truth Images

While not precisely matching a realistic scenario, the images generated imitate features that signals in a spectrogram would exhibit, mainly continuity and a sharp contrast from the background. The generated images also include extreme cases of densely clustered signals to push the limit of the model's ability to extract individual signals out of the cluster. Blank images were also included in the ground truth training set to train the model to expect patches with no signal and to prevent the model from hallucinating signals from random noise.

The ground truth images act as the target we want to achieve, and the noisy images are the model's input. Thus, the noisy images must be generated from the clean images by adding noise and corruption. The corruption added to the images should match the types of processes desired

to be removed, which can be gaussian noise, salt and pepper noise, more exotic types of noise or even structures caused by other processes that are different enough from the target. In this case, four types of corruptions were added: gaussian noise, salt and pepper noise, noisy columns, and vertical spikes.

The gaussian noise is random values added to the whole image sampled from a gaussian distribution parameterized by its mean and standard deviation. The salt and pepper noise is when a random number of pixels from the image is set to the highest or lowest value. The added noisy columns represent noise from external processes happening in some time interval. The noisy columns follow a gaussian distribution and are parameterized by the width of the column and position of the column in addition to the mean and standard deviation of the distribution. The vertical spikes represent the result of impulses picked up by the sensors and are columns of pixels set to the highest value at random time intervals. Figure 8 shows the corresponding images in Fig. 7 injected with the corruptions mentioned above. Similarly, the ground truth images, and the noisy images are extreme representations that may be found in actual data and are meant to push the model to its limits of denoising. Overall, the noise added together creates a non-stationary noise distribution that is hard to remove without losing information about the signals present.

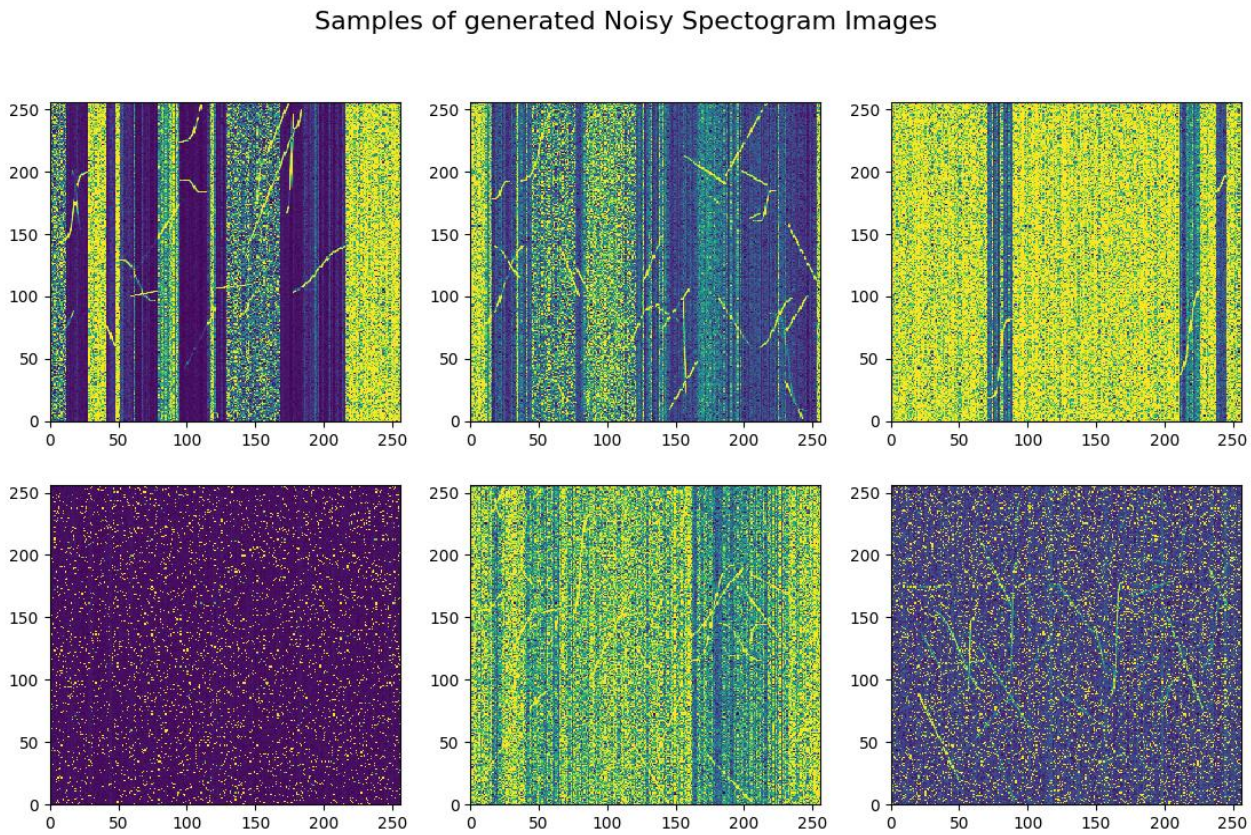

Figure 8 Sample of Synthetic Corrupted Images

III.c. Spectrogram Denoising Model and Training Parameters

The spectrogram denoising model used follows the MWCNN architecture shown in Fig. 9. The network consists of three down-sampling/up-sampling operations with eight "convolutional blocks", eight additional 2D convolutional layers and skip-addition connections between corresponding points in the encoder and decoder path. All convolutional layers except the last layer use a 3x3 filter kernel and are followed by a ReLU operation. The last convolutional layer is used to recreate the denoised image and thus uses a 1x1 filter kernel size and no ReLU

operation. The convolutional blocks consist of three layers of 2D convolutional layers, each followed by a ReLU and Batch Normalization.

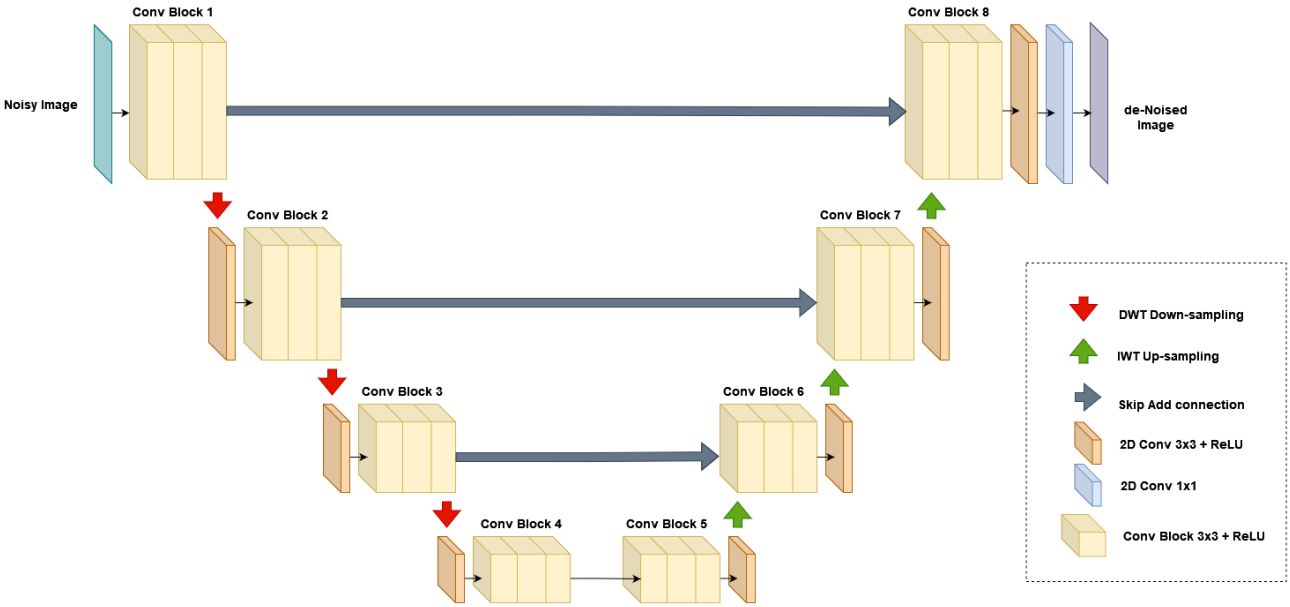

Figure 9 MWCNN architecture for proposed Spectrogram Denoising

The model weights were optimized by the widely used Adam optimizer [39] and an MSE loss function. The loss is calculated between the model output and the ground truth image. The training set consists of 10,000 samples of synthetic data generated in the manner shown in section (III.ii.). The three primary hyper-parameters of the training are the optimizer learning rate, batch size, and epoch. The optimizer learning rate controls the step size the optimization algorithm takes in updating the weights of neural at each update step. The batch size is the number of samples from the entire training set fed to the network for each forward pass through the network before the update step. The epoch parameter controls the number of total passes through the training data to complete. The final hyper-parameters used in training the model are

shown in Table 1. Better results can be achieved with more hyper-parameter tuning using grid search or other advanced techniques [40].

Hyper-parameter	Value
Learning Rate	0.0001
Batch size	2
Number of Epochs	10

Table 1 Hyper-parameters of Neural Network Training

III.d. Evaluation Metrics

In the space of image processing and denoising, the two standard metrics are the Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index Measure (SSIM) [41], which measure the quality of an image compared to a reference image. These metrics are used to quantify a model or algorithm's performance at various tasks, such as image reconstruction or image denoising. Both metrics are classified as full reference metrics, meaning they require the uncorrupted or original image to calculate the metrics.

In the context of images, the PSNR measures the ratio of the maximum possible pixel value to the mean square error of the corrupted/processed image to the true image in decibel scale. The higher the value, the better the image stands out from the noise. The higher the PSNR, the less corrupted the image, meaning a better compressing or denoising model. The equation of the PSNR is shown below.

$$PSNR = 10 \cdot \log_{10} \frac{MAX^2}{MSE}$$

The SSIM is a metric to measure the corruption of an image based on the changes in the structural information between the true image and the processed image. The SSIM considers an image's luminance, contrast, and structure and then performs a comparison to generate a quality assessment measure, as shown in Fig. 10. In general, the SSIM between two images x and y can be expressed as follows

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}$$

where the functions $l(x,y), c(x,y), s(x,y)$ measure the luminance, contrast, and structure similarity between the two images. The parameters $\alpha, \beta, and\ \gamma$ determine the relative importance of the components. With equal weighting of each component, by setting the parameters $\alpha = \beta = \gamma = 1$, the general formula shown below can be derived.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where:

- μ_x , μ_y are the mean pixel value of images x, y
- σ_x^2 , σ_y^2 are the variance of pixel value of images x, y
- σ_{xy} is the covariance between $x\ and\ y$
- $C_1 = (k_1 L)^2$, $C_2 = (k_2 L)^2$ are constants to stabilize the expression
- L is the dynamic range of pixel values of the images
- $k_1 = 0.01$ and $k_2 = 0.03$ is used in the default implementation of SSIM

In general, local structures in images vary spatially throughout an entire image resulting in different statistics between parts of the image. Calculating the SSIM over a moving window across the image, the local statistics of the image are preserved. Different window sizes and weighting schemes can be used, but the default implementation uses an $N \times N$ circular-symmetric Gaussian weighting scheme with a standard deviation of 1.5 normalized to unity. Then, the average SSIM of all the patches is taken to obtain the final overall metric. The value of the SSIM metric for the whole image ranges between -1 to 1, where 1 refers to the two images as completely similar while -1 refers to complete dissimilarity.

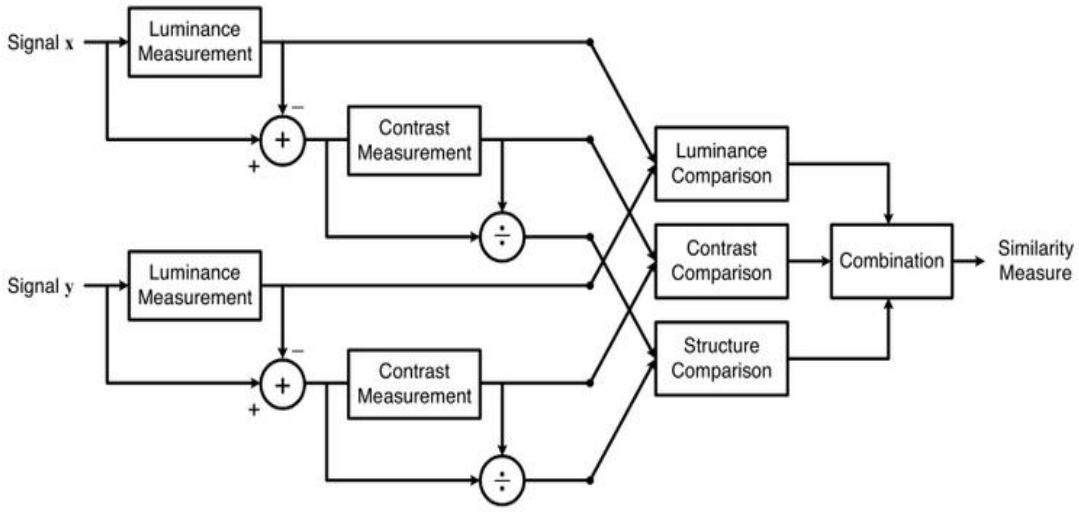

Figure 10 Diagram of SSIM computation [41]

IV. Denoising Model Results

A batch of an additional 1000 synthetic spectrograms is used to calculate the evaluation metrics PSNR and SSIM. The MWCNN was evaluated on the test set along with various denoising models or filtering techniques to achieve a comparison. Two groups of denoising techniques were used: non-data-adaptive and data-adaptive methods. The non-data-adaptive methods models are the gaussian filter, median filter, bilateral filter, TV filter, and BM3D. The adaptive data methods used are deep neural network architectures RED-net, U-Net, and MWCNN. Small and large versions of each architecture were used, which for U-Net and MWCNN mean larger convolutional filters at each block, and RED-net means more total layers. The results of all the methods are shown in Table 2. As a control, the PSNR and SSIM were calculated across the test set between the noisy images and ground truth.

Name	PSNR	SSIM	Name	PSNR	SSIM
Control	9.8	0.037	RED-Net10	17.44	0.072
Gaussian Filter	10.89	0.039	RED-Net20	28.00	0.729
Median Filter	11.32	0.048	U-Net	27.22	0.946
Bilateral Filter	10.89	0.033	U-Net (Large)	24.92	0.888
TV Filter	10.72	0.051	MWCNN	27.69	0.948

BM3D	9.8	0.052	MWCNN(Large)	28.88	0.959
Wavelet	10.24	0.04			

Table 2 Comparison of Denoising Methods

The results show that the data-adaptive methods perform significantly better than the non-data-adaptive methods. The poor performance of the non-data adaptive methods is because they are not suited for large global non-stationary noises. They can clear out some of the added gaussian noise, but not the more considerable corruption added. While removing some noise, most filtering methods performed worse in the SSIM metric due to the smoothing/blurring effect they cause to the image. On the other hand, the data-adaptive methods perform better because they have spent a considerable amount of computational time learning the statistics of the dataset and thusly perform well on this task.

In the class of data-adaptive methods, the MWCNN overall performs better than the others. The RED-Net20, U-Net, and MWCNN have the same approximate number of parameters (10 million). While the RED-Net20 has the higher PSNR, it has the lowest SSIM amongst the three, which means some structure information is lost. In this case, the difference between the U-Net and MWCNN is marginal.

A point of interest is among the large versions (40 million parameters) of the U-Net and MWCNN in which the U-Net performs worse than its smaller version, PSNR 24.92 vs 27.22 and SSIM 0.946 vs 0.888. The weaker performance is due to a lack of generalization in the training phase, as larger neural networks are prone to overfitting the dataset. On the other hand, the large

MWCNN performs the best among all the models tested, meaning the network architecture performs better at generalization than the other models tested.

Figure 11 shows the results of each neural network denoising models from a noisy sample image. The RED-net20, while removing significant amounts of noise, still has large amounts of noise. The U-Net and MWCNN perform about the same, with some distortions around the signals and a few false "hallucinated" signals, at reducing most of the noises. The larger U-Net model achieves sharp resulting signals with some artifacts spread throughout the image. The larger MWCNN

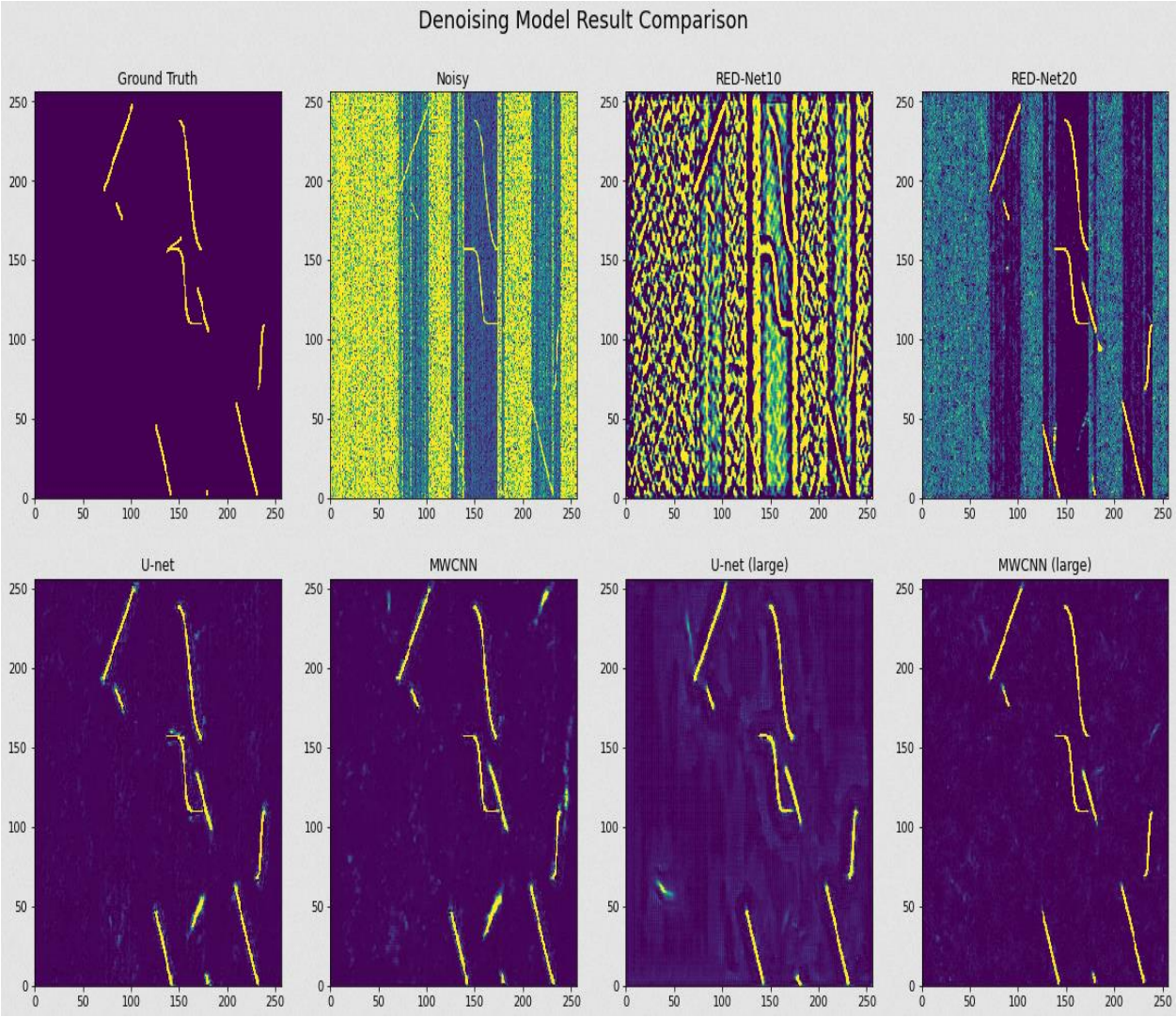

Figure 11 Denoising Model Comparisons between different Deep Denoising Architecture

model performs the best, creating sharp signals with the least amount of background artifacts which can be seen in Fig. 12.

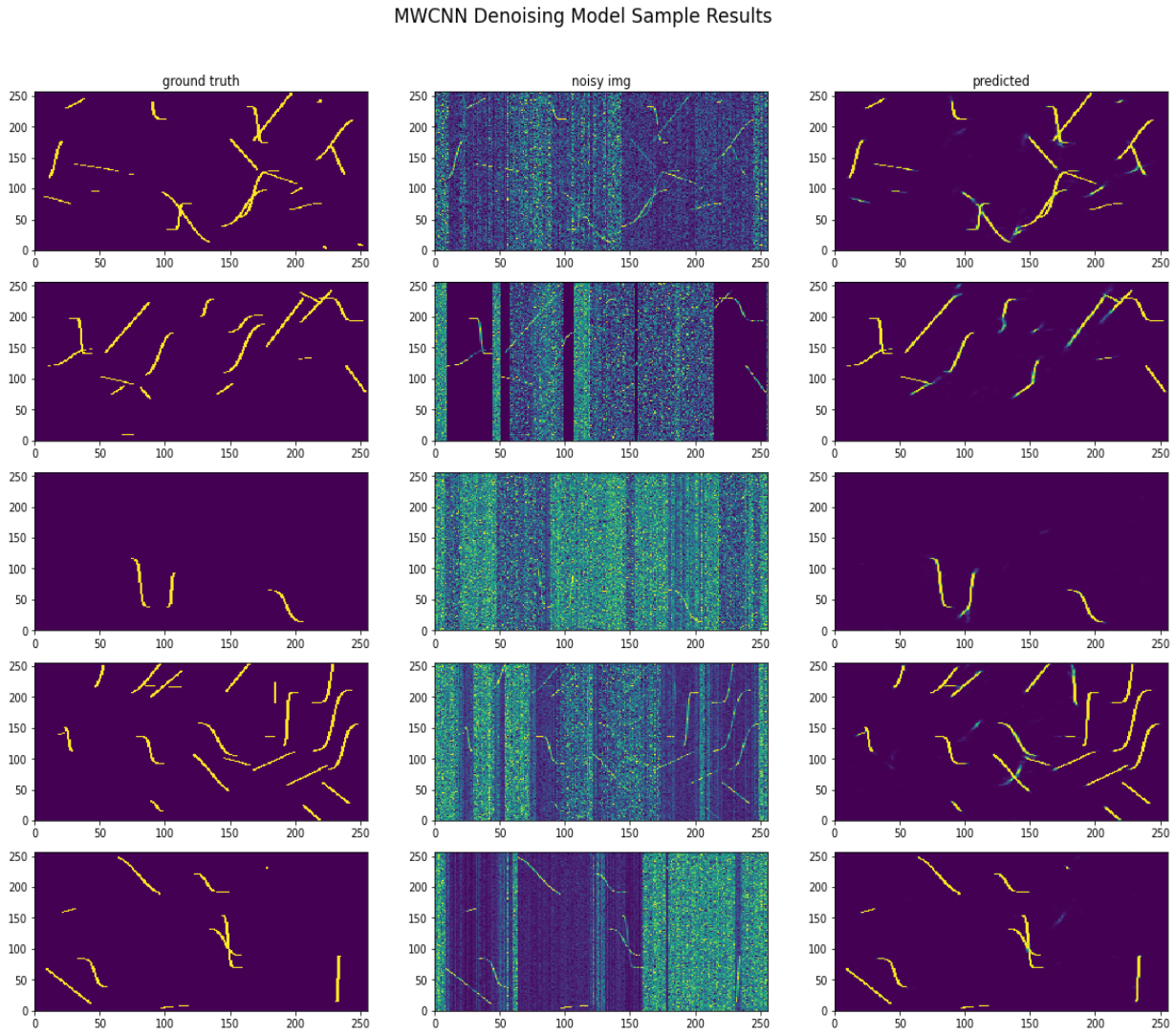

Figure 12 MWCNN sample results

So far, the denoising models have only been trained and evaluated with synthetic data emulating similar structures that could be found in real spectrogram data. ECEI diagnostic data from the DIII-D tokamak is used to generate a spectrogram from one channel to evaluate the denoising

model. The ECEI channel data comes in a time series, with each step corresponding to 1 microsecond. The spectrograms were generated using a sampling frequency of 1000 and a window size of 2048 samples. Figure 13 shows the results of denoising using the MWCNN denoising model on a 256x256 pixel patch corresponding to a frequency range of 125 kHz and a time interval of 612.15 milliseconds. The result shows that the vast majority of background noise has been removed, and structures have been highlighted. Some artifacts remain, and some "hallucinated" signals also appear. Additionally, in parts of the spectrogram where the SNR is low, or ambiguities exist, the model can not denoise properly, resulting in a blurry patch.

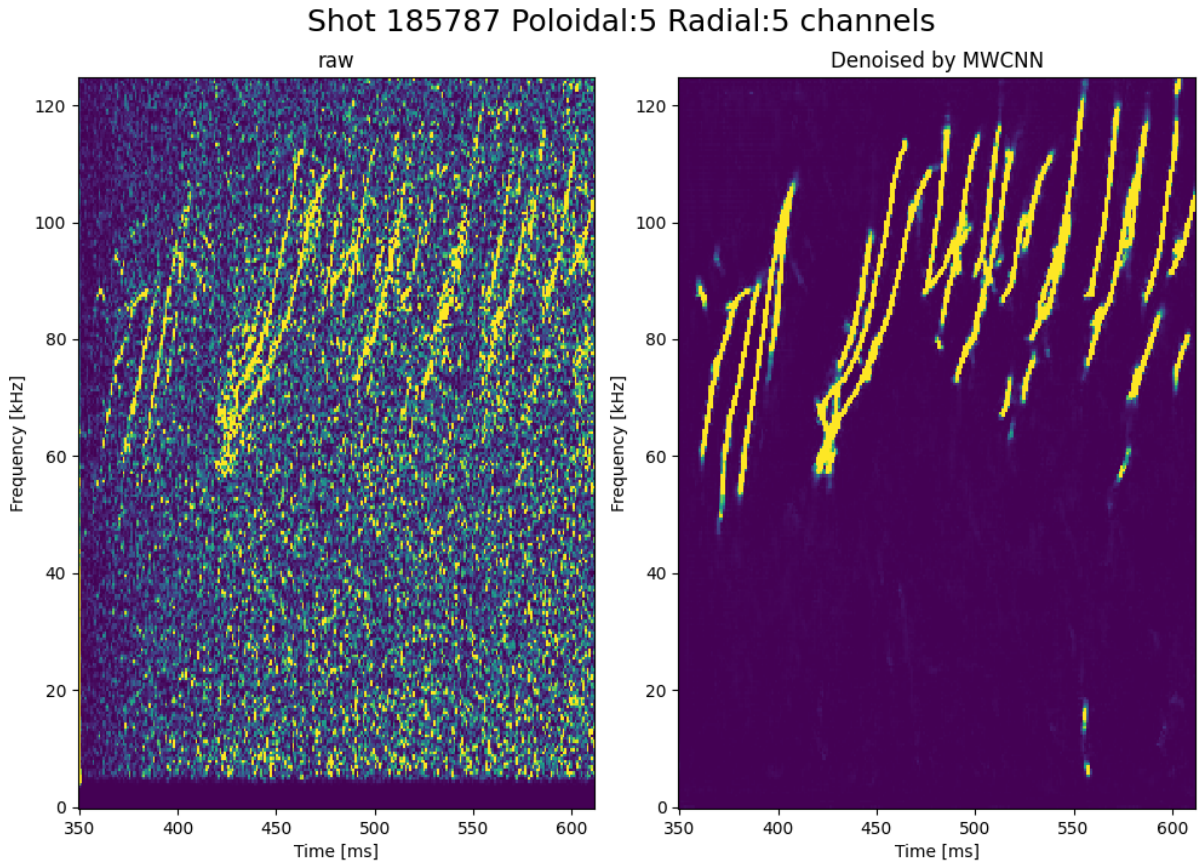

Figure 13 Results of MWCNN denoising model on ECEI data

V. Signal Extraction Pipeline

The results of the denoising models create images that often have unwanted artifacts that make disambiguating between different signals more difficult. A processing pipeline is created incorporating the denoising model and other techniques to detect the different excited Alfven eigenmodes from the time series and separate them into distinct signals. The pipeline takes in a time series of data containing possible Alfven eigenmodes and outputs a list of time series corresponding to each individual detected Alfven eigenmode. The pipeline has 9 steps: time domain low-pass filtering, short-time Fourier transform, spectrogram denoising, thresholding, morphological thinning, trace clustering, trace separation, filter mask creation and mask filtering, and inverse short-time Fourier transform.

V.a. Signal Extraction Pipeline Description

Step 1: Low-pass filtering:

The ECEI data comes from a voltage signal with a strong DC offset. The DC aspect of the signal can overpower all other high-frequency signals when converted into spectrogram form. To remove the DC bias, a simple 5th-order Butterworth lowpass filter is used with a cutoff at 10 kHz. The cutoff was arbitrarily chosen since the feature we care to observe occurs in higher frequency ranges.

Step 2: Spectrogram generation

First, a Short-Time Fourier Transform (STFT) of the filtered signal is taken to obtain the spectrogram. The STFT has two main parameters, the sampling rate and the window length, with

values of 1000 and 2048, respectively. The magnitude of the STFT is taken to create the spectrogram. For ease of use, the spectrogram bin values are scaled to be between 0-100.

Step 3: Spectrogram denoising

The spectrogram is then denoised using the deep denoising models. In this case, the MWCNN denoising model is used to denoise the spectrogram. The resulting image is rescaled again for ease of use.

Step 4: Thresholding

To remove any low amplitude noise or stray artifacts, a thresholding process is performed, turning the image into a binary image. To create the binary image, any pixel value above a threshold is set to one and any value below the threshold is set to zero. The denoised image strongly contrasts the detected signals and background noise; thus, algorithmically finding an optimal threshold is feasible. Multiple algorithms exist to find an optimal threshold, but the "Li" method [42] is used for this pipeline.

Step 5: Morphological thinning

In the domain of image processing, morphological operations are operations that "morph" an image based on the underlying structure of the image. Morphological processes determine the resulting pixel value using the neighborhood pixel values. Examples of basic morphological operations are dilations (addition of pixels based on neighboring pixels) and erosion (deletion of pixels based on neighboring pixels). The base operations can be chained together to create more complex morphological operations.

Currently, the binarized spectrogram has a thickness and roughness without any consistency, which is not desirable when making filter masks. To get to the centerline of each structure in the image, the morphological operation called thinning/skeletonizing is used [43]. The resulting "skeletonized" image consists of one-pixel wide lines corresponding to the center frequencies of filter masks that will be created. The skeletonized traces also allow for easier trace separation down the pipeline.

Figure 14 shows the effects of the thresholding and skeletonizing on the original denoised image. As seen in the middle plot, the thresholding has removed some of the stray weak artifacts. With the skeletonizing, everything has been reduced to one-pixel structures.

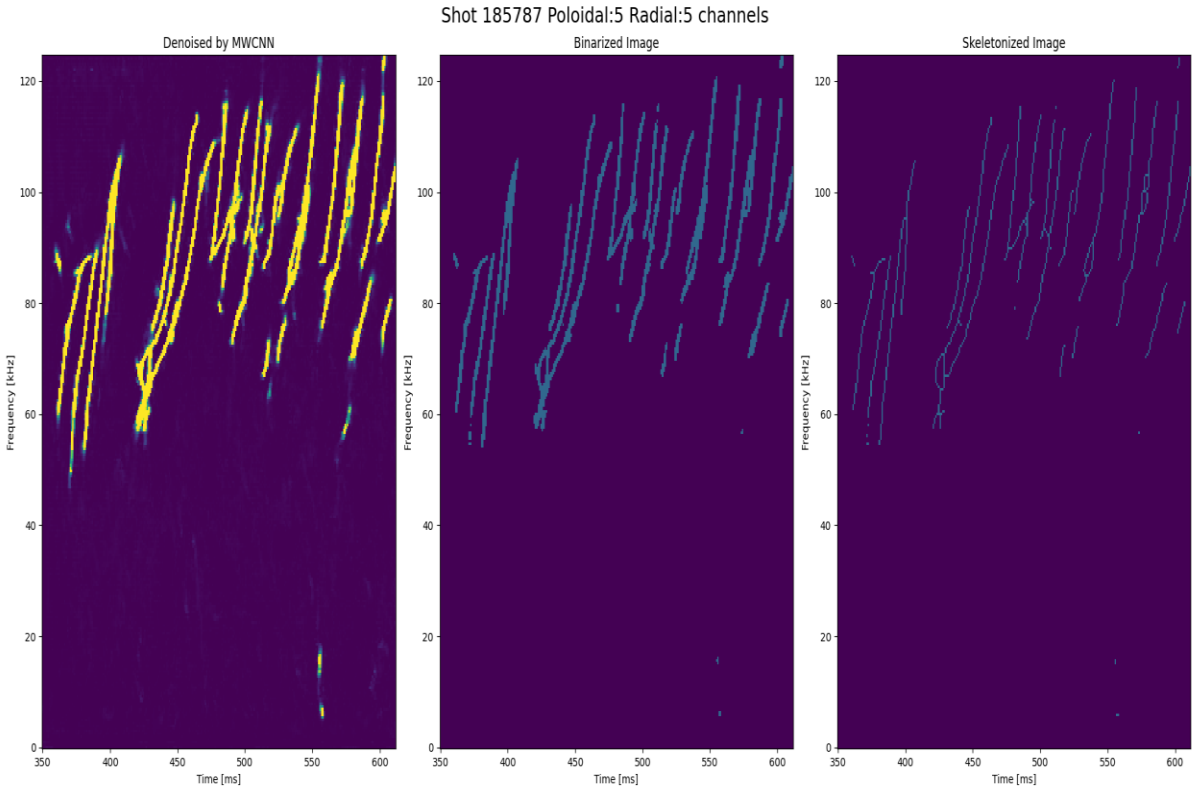

Figure 14 Processing of Denoised Image by Thresholding and Skeletonizing

Step 6: Trace clustering

The intended goal of the pipeline is to have separated and tagged filtered signals that correspond to different Alfven eigenmodes; thus, it is required to separate each unique trace in the image. The image can be converted into a point cloud where each pixel in a trace corresponds to a 2D point, with the x-axis being time and the y-axis being frequency. Different types of clustering algorithms exist, which can be categorized using different methods such as centroid, density, hierarchical or distribution methods, each of which has pros and cons. For example, the K-means clustering [44] method is a centroid-based clustering method that is very efficient but requires knowledge of the number of existing clusters in the data you want to cluster. Additionally, centroid-based clustering methods do not work well when the "shape" of the data is not geometrically "flat", such as in our case of packed curving traces. On the other hand, density-based clustering methods such as DBSCAN [45] are better suited to non-flat geometries at the cost of computational efficiency.

For our use case, we use an extension of DBSCAN called Hierarchical-DBSCAN (HDBSCAN) [46], which overcomes some of the limitations of the regular DBSCAN, such as requiring clusters to be uniform density. In addition to clustering segments, the clustering algorithm acts as another stage of denoising by removing stray or small groups of points that are not in any sizable existing cluster. The clustering algorithm will have two outcomes: a cluster containing one unique trace or intersecting traces. In the first case, the task is complete, and the unique trace is separated. In the case of intersecting traces in one cluster, we need to separate them further into separate traces.

Step 7: Trace separation

The trace separation algorithm used in this work is a tracing algorithm from another more extensive spectrogram processing pipeline used to detect whale chirps in spectrograms [47]. The tracing algorithm takes in a binary image where each "on" pixel corresponds to a point on the traces. From the binary image, for each active pixel a "connection degree" is calculated based on the number of active neighbors. Based on each point's connection degree, they are categorized into Basis points (BP), Connection points (CP), and Node points. The Basis points are the endpoints of the traces, the Connection points are the points along a unique trace, and the Node points are the points of intersection between traces. The algorithm starts with a BP and collects each adjacent CP into one trace until it reaches a node point. At the node point, the direction of the current trace is calculated and then CP in the same direction as the current trace is collected until a BP is reached. After a BP is reached, the collected points are removed, and the algorithm continues until no other points are left.

Each cluster containing multiple possible traces in this pipeline is converted into a binary image and fed to the trace-separating algorithm described. The result is a list of points belonging to a unique potential cluster. A "Savgol" smoothing filter is applied for each set of points to smoothen each trace. The results of the trace separation are overlaid over the spectrogram in Fig. 15. In the results, we can generally see the traces are separated into unique traces; however, for

some traces, deformations are apparent, and in some cases, the traces were "deleted". The deletion error is likely due to this work's implementation of the algorithm cited in the paper.

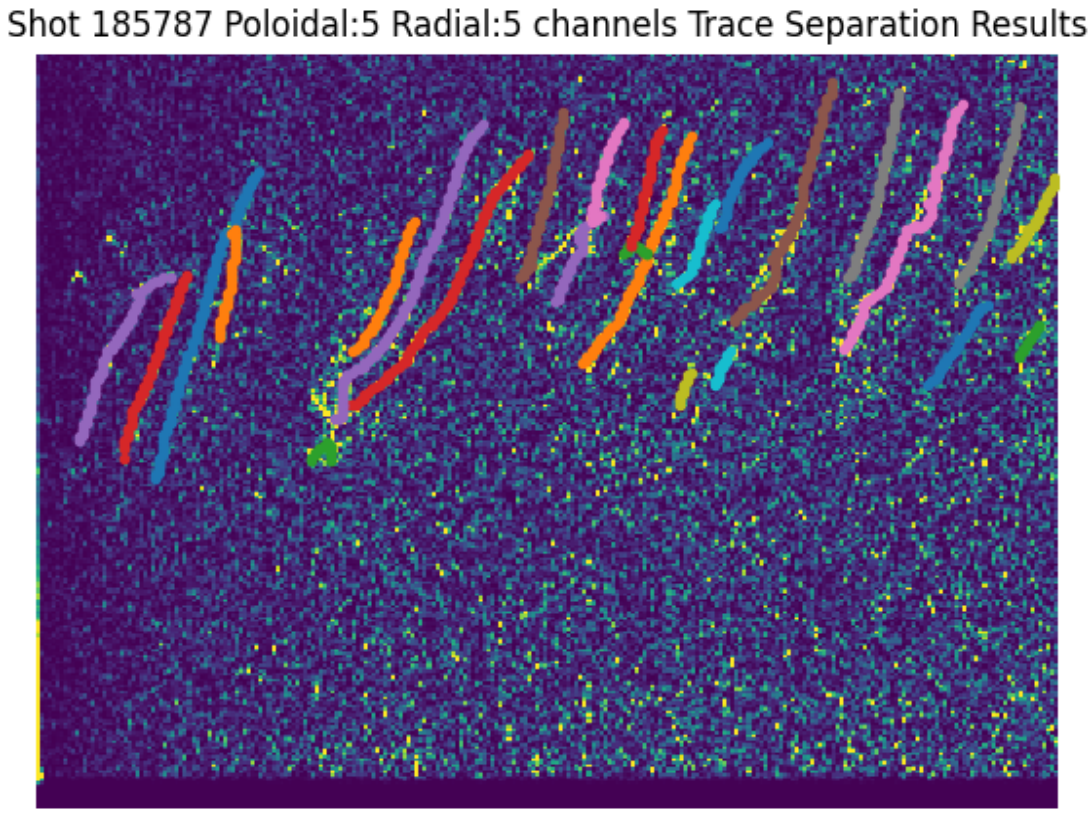

Figure 15 Results of trace separation algorithm in pipeline. Each unique trace is overlayed on the original spectrogram.

Step 8: Filter mask creation

Using the individual traces filter mask can now be created. Spectrogram filter masks are used as a method of filtering spectrogram data. The principle of mask filtering is that by zeroing out all bins except where the desired signal is, by element-wise multiplication of the filter mask with the original spectrogram, only the desired signal remains, which can then be converted back into the time domain. Zeroing out time-frequency bins removes the signal contribution of that bin when

reverted to the time domain. Since the spectrogram has inherent uncertainty in both time and frequency, the filter mask should be wider than one bin/pixel so that the chances we filter the desired signal become higher. To achieve this, the filter is expanded above and below the center line by some number of pixels corresponding to a desired bandwidth. Each filter mask is a matrix the size of a spectrogram with regions where the desired signal should be set to unity while everywhere else set to zero. The filter mask's sharp boundary could cause a small ringing phenomenon in the time domain. A more gradual windowing around the filter mask can combat the ringing.

With each filter mask created, the filtering process can now be performed. The original complex-value short-time Fourier transformed matrix is used as the object of filtering because the phase information is required to accurately reconstruct the desired signals using the inverse short-time Fourier transform. Each filter mask is applied to the STFT matrix element-wise to create a list of "filtered" matrices corresponding to each filtered trace.

Step 9: Inverse short-time Fourier transform

To get the final individual detected Alfven eigenmode in the time domain the inverse short-time Fourier transform is performed on the filtered matrices with the same parameters used in the original short-time Fourier transform. The result is a time series spanning the entire length of the original data with only the active signal at the time corresponding to the filtered Alfven eigenmode. Some post-processing is performed to only get the segment of data corresponding to the detected mode, which is then saved. Figure 16 shows the result of the pipeline for one specific existing mode. The mode is filtered using the pipeline and then returned to the time-frequency domain to be compared with the original unfiltered data. We can see that only a single trace remains in the filtered spectrogram.

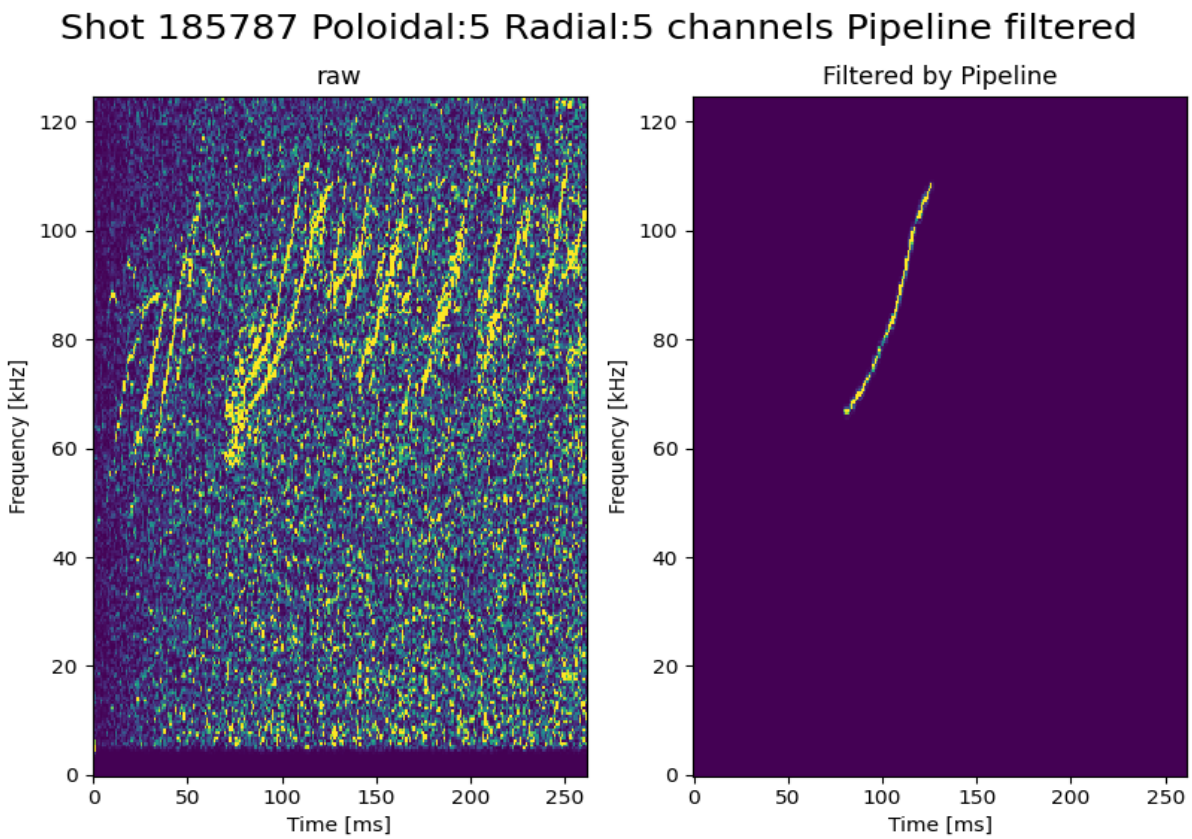

Figure 16 Filtering Pipeline results compared to raw spectrogram

V.b. Application of Signal detecting pipeline

Currently, the pipeline takes only one data channel and detects and filters out the Alfven eigenmodes. In actuality, the Alfven eigenmode is a rotating structure that can be visualized as a 2D cross-sectional image generated by all the channels of the ECEI system. The pipeline can be modified to be applied to all the channels to filter out detected modes by using the filter masks from an initially selected channel. If a mode does not appear in the channel used to generate the filter masks, it will not be detected and filtered out to generate the image. Figure 17 is an example of a detected Alfven eigenmode in a 2D image generated from all the channels at a specific time.

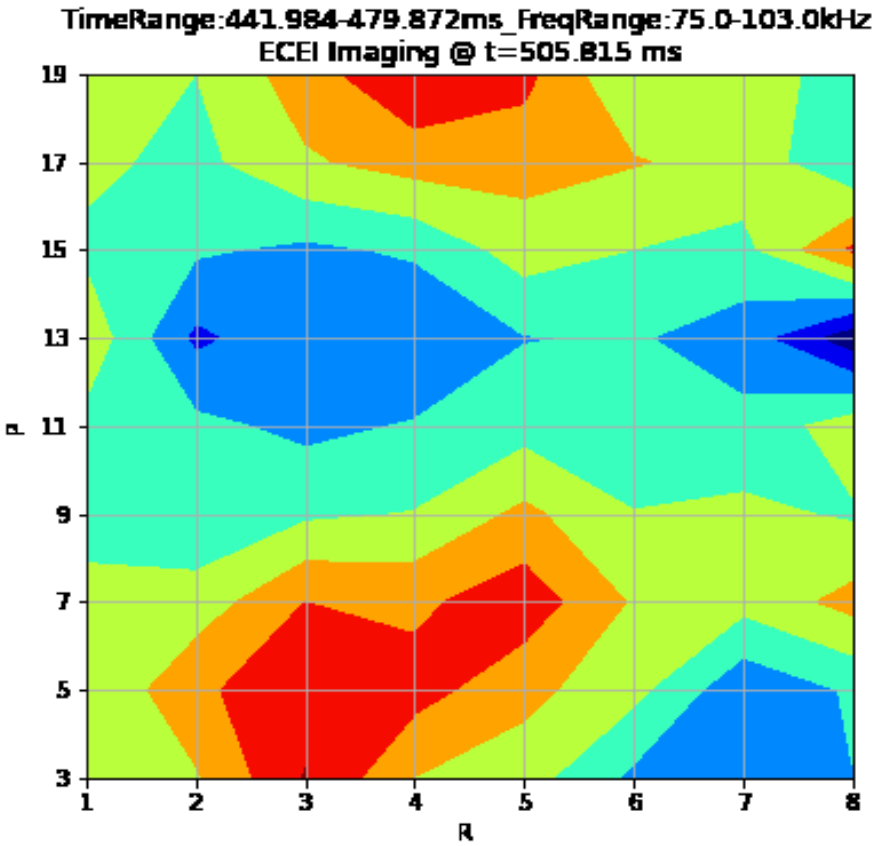

Figure 17 2D image of a filtered Alfven Eigenmode in Shot 185787

VI. Conclusion

Modern Plasma physics research typically involves investigating vast amounts of data generated by cutting-edge plasma diagnostics. Any tool that speeds up or automates the process is valuable to a researcher. One avenue of study in plasma physics is the excited modes that occur in different plasma experiments. The excited modes are often detected using spectrograms generated from plasma diagnostic tools such as Electron Cyclotron Emission Imaging. Typically, the recorded data are very noisy due to the plasma environment making it difficult to manually and automatically detect points of interest in the data.

This work created a pipeline to detect and extract the Toroidal Alfven Eigenmodes automatically. The pipeline uses ECEI channel data, and through a pipeline of filtering, denoising, clustering and tracing, any detected Toroidal Alfven Eigenmodes are tagged and filtered out into separate signals.

One of the main aspects of the pipeline is a Deep Denoising model used to remove noise and enhance features in the spectrograms under analysis. The architecture used for the denoising model is a multi-level wavelet convolutional neural network. Due to limited data, the model was trained using synthetic data designed to recreate similar features of real Alfven Eigenmodes. Nevertheless, the trained denoising model adequately enhanced the noisy spectrograms with minimal false positives.

Many aspects of the pipeline can be improved, but here we can classify them into two categories: Deep denoising model and pipeline improvements. To improve the Deep denoising model, two avenues can be pursued: better architectures or better data. The model used in the current pipeline is a relatively straightforward extension of the already simple U-Net model architecture.

The current state of the art in deep denoising uses much more complex architectures, which are either extension of the CNN U-Net architecture [48], [49] or use the newer Transformer architectures [50], [51].

In practice in deep learning, most of the gains come from better data. Thus, the same architecture can achieve better results if the models are trained on better data. In this case, that would either mean a hand-cleaned dataset of the structures we want to denoise, or data generated from a direct simulation. Both methods add additional labor to human labeling or computations.

The clustering method can improve different aspects of the overall pipeline, but one primary way is the clustering method. Clustering could be applied to all the channels so that features could be cross-correlated to remove hallucinated structures and detect modes across all the channels in one go.

With the development of modern diagnostics tools, vast amounts of high-resolution data are being created rapidly every year. Advanced machine learning techniques that require large amounts of data can now be used to aid researchers in their work. Automated pipelines can be built to clean or classify desired features saving countless human hours and allowing rapid iterations through the research process to increase the speed of science.

VII. References

[1] K.-L. Wong, “A review of Alfvén eigenmode observations in toroidal plasmas,” *Plasma Phys Control Fusion*, vol. 41, no. 1, pp. R1–R56, Jan. 1999, doi: 10.1088/0741-3335/41/1/001.

[2] M. A. van Zeeland *et al.*, “Radial Structure of Alfvén Eigenmodes in the DIII-D Tokamak through Electron-Cyclotron-Emission Measurements,” *Phys Rev Lett*, vol. 97, no. 13, p. 135001, Sep. 2006, doi: 10.1103/PhysRevLett.97.135001.

[3] Y. Zhu *et al.*, “W-band system-on-chip electron cyclotron emission imaging system on DIII-D,” *Review of Scientific Instruments*, vol. 91, no. 9, p. 093504, Sep. 2020, doi: 10.1063/5.0018082.

[4] J. Benesty, J. Chen, and Y. Huang, “Study of the widely linear Wiener filter for noise reduction,” in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 205–208. doi: 10.1109/ICASSP.2010.5496033.

[5] M. Yektaeian and R. Amirfattahi, “Comparison of spectral subtraction methods used in noise suppression algorithms,” in *2007 6th International Conference on Information, Communications & Signal Processing*, 2007, pp. 1–4. doi: 10.1109/ICICS.2007.4449542.

[6] N. E. Huang *et al.*, “The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis,” *Proceedings*

of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, vol. 454, no. 1971, pp. 903–995, Mar. 1998, doi: 10.1098/rspa.1998.0193.

[7] M. Pal, R. Roy, J. Basu, and M. S. Bepari, "Blind source separation: A review and analysis," in *2013 International Conference Oriental COCOSDA held jointly with 2013 Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE)*, Nov. 2013, pp. 1–5. doi: 10.1109/ICSDA.2013.6709849.

[8] A. Belouchrani and M. G. Amin, "Blind source separation based on time-frequency signal representations," *IEEE Transactions on Signal Processing*, vol. 46, no. 11, pp. 2888–2897, 1998, doi: 10.1109/78.726803.

[9] M. T. Pourazad, Z. Moussavi, F. Farahmand, and R. K. Ward, "Heart Sounds Separation From Lung Sounds Using Independent Component Analysis," in *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, 2005, pp. 2736–2739. doi: 10.1109/IEMBS.2005.1617037.

[10] D. Rife and R. Boorstyn, "Single tone parameter estimation from discrete-time observations," *IEEE Trans Inf Theory*, vol. 20, no. 5, pp. 591–598, Sep. 1974, doi: 10.1109/TIT.1974.1055282.

[11] R. A. Altes, "Detection, estimation, and classification with spectrograms," *J Acoust Soc Am*, vol. 67, no. 4, pp. 1232–1246, Apr. 1980, doi: 10.1121/1.384165.

[12] Yu Shi and E. Chang, “Spectrogram-based formant tracking via particle filters,” in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP ’03).*, 2003, pp. I-168-I–171. doi: 10.1109/ICASSP.2003.1198743.

[13] P. Gruden and P. R. White, “Automated tracking of dolphin whistles using Gaussian mixture probability hypothesis density filters,” *J Acoust Soc Am*, vol. 140, no. 3, pp. 1981–1991, Sep. 2016, doi: 10.1121/1.4962980.

[14] X. Mankun, P. Xijian, L. Tianyun, and X. Mantian, “A New Time-Frequency Spectrogram Analysis of FH Signals by Image Enhancement and Mathematical Morphology,” in *Fourth International Conference on Image and Graphics (ICIG 2007)*, Aug. 2007, pp. 610–615. doi: 10.1109/ICIG.2007.154.

[15] W. B. Hussein, “Spectrogram Enhancement By Edge Detection Approach Applied To Bioacoustics Calls Classification,” *Signal Image Process*, vol. 3, no. 2, pp. 1–20, Apr. 2012, doi: 10.5121/sipij.2012.3201.

[16] A. Kershenbaum and M. A. Roch, “An image processing based paradigm for the extraction of tonal sounds in cetacean communications,” *J Acoust Soc Am*, vol. 134, no. 6, pp. 4435–4445, Dec. 2013, doi: 10.1121/1.4828821.

[17] T. A. Lampert and S. E. M. O’Keefe, “An active contour algorithm for spectrogram track detection,” *Pattern Recognit Lett*, vol. 31, no. 10, pp. 1201–1206, Jul. 2010, doi: 10.1016/j.patrec.2009.09.021.

[18] J. Jiang *et al.*, “Whistle detection and classification for whales based on convolutional neural networks,” *Applied Acoustics*, vol. 150, pp. 169–178, Jul. 2019, doi: 10.1016/j.apacoust.2019.02.007.

[19] 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, doi: 10.1109/ACCESS.2019.2928017.

[20] A. Jalalvand *et al.*, “Alfvén eigenmode classification based on ECE diagnostics at DIII-D using deep recurrent neural networks,” *Nuclear Fusion*, vol. 62, no. 2, p. 026007, Feb. 2022, doi: 10.1088/1741-4326/ac3be7.

[21] S.-W. Fu, T. Hu, Y. Tsao, and X. Lu, “Complex spectrogram enhancement by convolutional neural network with multi-metrics learning,” in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, Sep. 2017, pp. 1–6. doi: 10.1109/MLSP.2017.8168119.

[22] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” May 2015.

[23] L. Fan, F. Zhang, H. Fan, and C. Zhang, “Brief review of image denoising techniques,” *Vis Comput Ind Biomed Art*, vol. 2, no. 1, p. 7, Dec. 2019, doi: 10.1186/s42492-019-0016-7.

[24] D. H. Rao and P. P. Panduranga, “A Survey on Image Enhancement Techniques: Classical Spatial Filter, Neural Network, Cellular Neural

Network, and Fuzzy Filter," in *2006 IEEE International Conference on Industrial Technology*, 2006, pp. 2821–2826. doi: 10.1109/ICIT.2006.372671.

[25] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pp. 839–846. doi: 10.1109/ICCV.1998.710815.

[26] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, no. 1–4, pp. 259–268, Nov. 1992, doi: 10.1016/0167-2789(92)90242-F.

[27] P. Jain and V. Tyagi, "Spatial and Frequency Domain Filters for Restoration of Noisy Images," *IETE Journal of Education*, vol. 54, no. 2, pp. 108–116, Jul. 2013, doi: 10.1080/09747338.2013.10876113.

[28] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007, doi: 10.1109/TIP.2007.901238.

[29] A. Buades, B. Coll, and J.-M. Morel, "A Non-Local Algorithm for Image Denoising," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pp. 60–65. doi: 10.1109/CVPR.2005.38.

[30] D. D. Muresan and T. W. Parks, "Adaptive principal components and image denoising," in *Proceedings 2003 International Conference on Image*

Processing (Cat. No.03CH37429), pp. I-101–4. doi: 10.1109/ICIP.2003.1246908.

[31] A. Hyvarinen, E. Oja, P. Hoyer, and J. Hurri, "Image feature extraction by sparse coding and independent component analysis," in *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, pp. 1268–1273. doi: 10.1109/ICPR.1998.711932.

[32] M. Elad and M. Aharon, "Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006, doi: 10.1109/TIP.2006.881969.

[33] Y. He, T. Gan, W. Chen, and H. Wang, "Adaptive Denoising by Singular Value Decomposition," *IEEE Signal Process Lett*, vol. 18, no. 4, pp. 215–218, Apr. 2011, doi: 10.1109/LSP.2011.2109039.

[34] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017, doi: 10.1109/TIP.2017.2662206.

[35] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008, pp. 1096–1103. doi: 10.1145/1390156.1390294.

[36] P. Liu, H. Zhang, W. Lian, and W. Zuo, “Multi-Level Wavelet Convolutional Neural Networks,” *IEEE Access*, vol. 7, pp. 74973–74985, 2019, doi: 10.1109/ACCESS.2019.2921451.

[37] S. G. Mallat, “A theory for multiresolution signal decomposition: the wavelet representation,” *IEEE Trans Pattern Anal Mach Intell*, vol. 11, no. 7, pp. 674–693, Jul. 1989, doi: 10.1109/34.192463.

[38] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[39] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014.

[40] T. Yu and H. Zhu, “Hyper-Parameter Optimization: A Review of Algorithms and Applications,” Mar. 2020.

[41] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image Quality Assessment: From Error Visibility to Structural Similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: 10.1109/TIP.2003.819861.

[42] C. H. Li and C. K. Lee, “Minimum cross entropy thresholding,” *Pattern Recognit*, vol. 26, no. 4, pp. 617–625, Apr. 1993, doi: 10.1016/0031-3203(93)90115-D.

[43] T. C. Lee, R. L. Kashyap, and C. N. Chu, “Building Skeleton Models via 3-D Medial Surface Axis Thinning Algorithms,” *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462–478, Nov. 1994, doi: 10.1006/cgip.1994.1042.

[44] S. Lloyd, “Least squares quantization in PCM,” *IEEE Trans Inf Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982, doi: 10.1109/TIT.1982.1056489.

[45] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” *Kdd*, vol. 96, no. No. 34, pp. 226–231, 1996.

[46] L. McInnes, J. Healy, and S. Astels, “hdbscan: Hierarchical density based clustering,” *The Journal of Open Source Software*, vol. 2, no. 11, p. 205, Mar. 2017, doi: 10.21105/joss.00205.

[47] X. Wang *et al.*, “A method for enhancement and automated extraction and tracing of Odontoceti whistle signals base on time-frequency spectrogram,” *Applied Acoustics*, vol. 176, p. 107698, May 2021, doi: 10.1016/j.apacoust.2020.107698.

[48] H. Huang *et al.*, “UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 1055–1059. doi: 10.1109/ICASSP40776.2020.9053405.

[49] C.-M. Fan, T.-J. Liu, and K.-H. Liu, “Selective Residual M-Net for Real Image Denoising,” Mar. 2022.

[50] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, "Restormer: Efficient Transformer for High-Resolution Image Restoration," Nov. 2021.

[51] Z. Wang, X. Cun, J. Bao, W. Zhou, J. Liu, and H. Li, "Uformer: A General U-Shaped Transformer for Image Restoration," Jun. 2021.

[52] Jeremy Jordan, "Introduction to autoencoders.," Mar. 19, 2018.