ORNL/SPR-2021/2381

Oak Ridge National Laboratory

Power System Waveform Classification Using Time-Frequency and CNN

Christopher Sticht

1/2022

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/SPR-2021/2381

Project:

DarkNet – Cyber-physical Security for Energy Sector's Critical Infrastructure Advanced Communication Systems

TOPIC I:
Optimization of the DarkNet Environment, Edge Processing and Data Analytics

Task 3.
Data Analytics and Event Correlation

For

Office of Cybersecurity, Energy Security and Emergency Response

Cybersecurity for Energy Delivery Systems

FY 2021-2022

Power System Waveform Classification Using Time-Frequency and CNN

Author(s)
Christopher Sticht

Date Published:
1/2022

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

Contents

ABSTRACT

To address the need for a commercially viable solution that can classify waveform data, energies were directed to develop a universal neural network (NN) structure (deep learning algorithm) that works for a wide variety of system event types (such as; faults, voltage dips, harmonics, etc.). The structure that showed the most promise was one that included the use of spectrograms. The technique has shown positive results in audio engineering, particularly with respect to speech recognition.

A Power System Neural Network (PSNN) has been developed to use a CNN to classify events within waveform data for power systems. The waveform is converted to an array of values by way of spectrograms and interpreted as an image. This image is passed into the CNN. The test results on independent simulated test and validation datasets show greater than 99% accuracy.

Special thanks to:

- David Wells of the Department of Energy funded the work for the PSNN through the DarkNet Project
- Yarom Polsky, Thomas King and William Monday of Oak Ridge National Laboratory (ORNL) for managing the DarkNet Project
- Travis Smith, formerly of ORNL who supported some of the earlier work on the machine learning on power system waveform data
- Jim Glass and Bob Hay of EPB of Chattanooga who provided waveform data and explained how they currently process their waveform data as well as a distribution circuit model in CYMEdist format which was converted to MATLAB/Simulink
- Mike Marshall, formerly of ORNL and Neil Shepard who converted distribution circuit model in CYMEdist format to MATLAB/Simulink
- Isabelle Snyder and Emilio Piesciorovsky of ORNL who modified the MATLAB/Simulink and used the model to create simulated waveform datasets used for training and testing the various neural network models
- Aaron Wilson for helping to edit the report

1. EXECUTIVE SUMMARY

A Power System Neural Network (PSNN) has been developed to use a CNN to classify events within waveform data for power systems. The waveform is converted to an array of values by way of spectrograms and interpreted as an image. This image is passed into the CNN. The PSNN is very promising and should work for a wide variety of power system conditions of interest. Ultimately, much of the custom code and tools used today and much of the manual effort expended today may be automated using this PSNN.

Spectrograms are plots of short time Fourier transforms. Short time Fourier transforms are time sampled Fourier transforms of waveform data. In other words, a Fourier transform is performed on a snippet of waveform data (a time step). The next snippet of waveform data is also converted to a Fourier transform and so forth and so on until the entire waveform is converted to samples of Fourier transforms. Each Fourier transform results in a spectrum of intensities for each frequency that exists within that snippet.

Each of the snippets has a spectrum associated with it that describes all the frequency components of that snippet. The frequency spectrum of each snippet is horizontally stacked next to each other for all snippets of the overall waveform. This creates a two-dimensional array of data where each column represents a snippet of time within the waveform and each row represents a frequency (see Figure 13).

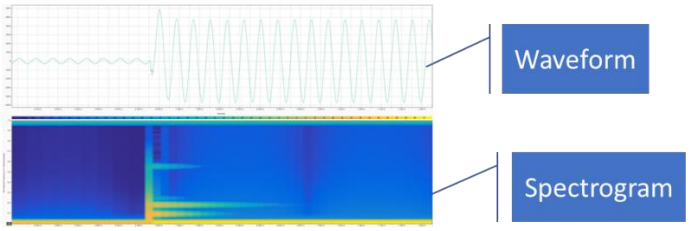


Figure 1 Spectrogram of a Waveform

The array of values can be interpreted as an image where at or near the bottom of the image are the low frequency values and the top the image are the high-frequency values for each time step.

This image of the Short Time Fourier transform can be passed into a CNN. In the case of power system data some additional modification of the data is required before passing it to a CNN for training. No two circuits will produce the exact same waveform for a given fault type, but the NN can be trained to classify the fault type regardless of the circuit or location on the circuit.

1.1 POWER SYSTEM FILTER-BANK (CHANNELIZER)

The FFT maps to a wide spectrum of frequencies. However, in power systems analysis it has been long established that harmonics are important to the understanding of what is taking place on the power system.

Given that the STFT produces results in a wide frequency spectrum and power system analysis is based on harmonics of the fundamental frequency there is a need for a way to convert FFT results into harmonic components. A filter-bank is used to do this conversion. The filter-bank is a function designed to extract the harmonic frequencies from the spectrogram image thereby reducing the number of rows in the spectrogram and focusing on the most relevant power system information.

Each row of the STFT image represents a particular frequency, the row is multiplied by the corresponding frequency column of the filter-bank. Rows of the spectrogram that do not correspond with the center

frequency of a filter get multiplied by value less than one and in many cases by value of zero. Frequency rows of the STFT image that correspond to a center frequency in the filter-bank get multiplied by a value of one. The results get added together to create a new row for spectrogram that corresponds to each filter in the filter-bank. This results in a spectrogram with fewer rows but highlights the most relevant information. To better illustrate the point in the function of the filter bank see Figure 50. This approach is advantageous when processing the spectrograms using a Convolutional Neural Network (CNN).

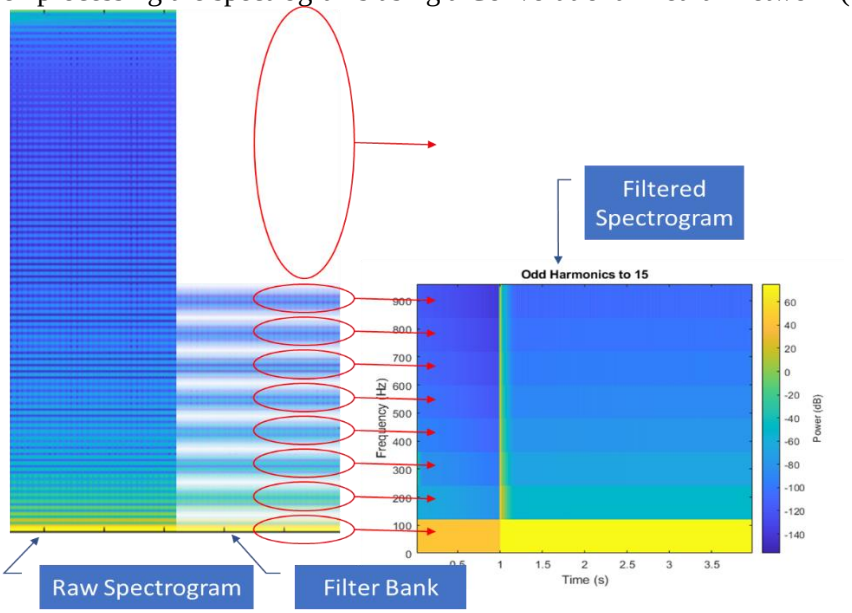


Figure 2 The Behavior of the Filter-Bank

1.2 SIGNAL DATA NORMALIZATION

When performing power systems analysis on waveforms there are two types of signals used, namely voltage and current signals. Both voltage and current have three phases. The amplitude of the voltage signals have an order of magnitude of tens of thousands of volts (10^4 volts) or hundreds of thousands of volts (10^5 volts) depending on whether the voltage is for distribution or transmission. The amplitude of the current signal may be on the order of tens of amps (10s amps) for load currents to two-digit thousands of amps (10^4 amps) for fault currents.

There is enough difference between voltage magnitude, load current magnitude and fault current magnitude that necessitates some compensation so that the load current signal is of a comparable magnitude to the voltage signal. Therefore, the waveform data needs to be scaled, or normalized, so as to make the voltage and current waveforms have comparable scales.

1.3 MERGING THE VOLTAGE AND CURRENT SPECTROGRAMS

The Voltage spectrograms were combined with the corresponding phase of the Flipped Current spectrograms into single spectrogram images where voltage and current are separated by a column of zeros. The current was put on the left and the voltage was put on the right as illustrated in Figure 61. The output was saved to a 3-dimensional array with spectrograms for each phase along the third dimension.

In the convolution process, the new pixel of the convolved image is the dot product of the CNN kernel with the part of the original image that is of the same size as the kernel. Depending on how the kernel and/or the CNN is set up the edges of the image include the dot product with in some cases padding

values (zeros) that surround the image. Therefore, the edges of the image include convolution with zero padding which washes out information contained at the edge of the image. Combining the voltage and current into single images in this way keeps the most important information in the center of the image thereby reducing or eliminating the washout effect. Moving the important information to the center of the image ultimately improves the results of the CNN.

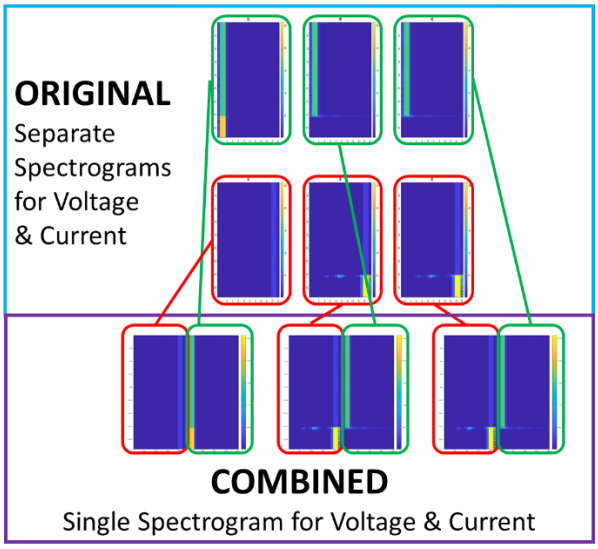


Figure 3 Combined Spectrograms of Voltage and Flipped Current

1.4 LOGARITHM

The spectrograms were processed by applying a logarithm. Taking a log of small numbers can lead to roundoff error. Therefore, the features were scaled using a small offset then taking the log. This provides the data with a smoother distribution and accentuates the most important information.

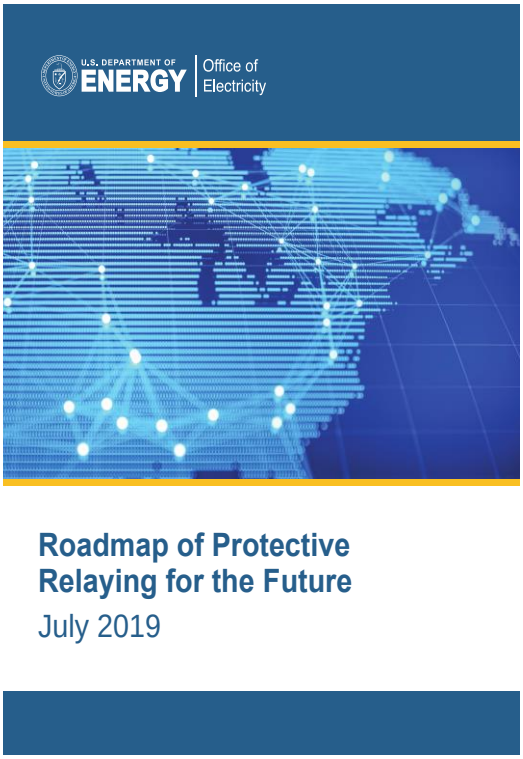
1.5 CONFIGURE AND TRAIN THE CONVOLUTIONAL NEURAL NETWORK (CNN)

At this point the spectrogram images were passed through the CNN for training. The structure of the neural network is illustrated in Figure 64.

Figure 4 Neural Network Structure

The training options for the neural network include the Adam optimizer, with 25 max Epochs, a Mini-batch size of 128, and an Initial Learn Rate of 3e-4.

2. BACKGROUND AND CONTEXT



ORNL led the development of the DOE Roadmap of Protective Relaying for the Future which started with a White Paper "Future State of Protecting Relaying" which later went through a peer-review comprised of 70 utility and industry participants that identified protection issues that need to be addressed in the electric utility industry. The white paper and peer-review culminated in a protective relaying Roadmap that included timeline and strategies for developing tools and equipment to address protection needs for the future. The roadmap included Protective Relaying Challenges including:

Transmission and Generation

- Risk of settings, configuration errors
- Adequate post-fault analysis
- Aging protection infrastructure
- Workforce development
- Understanding the many settings possible in a protective relay
- Communications performance, bandwidth and speed
- Accurate fault detection and localization
- Advanced sensor integration
- Lack of negative sequence components with DER adversely impacts directional relaying

Distribution

- Adaptive protection for different modes of operation, flexible configuration
- High impedance fault detection
- Changing communications infrastructure
- Model coordination, data management
- DER, bidirectional power flow

Microgrid

- Coordination of distributed and central intelligence
- Interoperability, standards for legacy and new inverters
- Protection schemes for dynamic systems

Some work in the industry has investigated phase identification [1] [2] [3]. There was not a lot found in the literature where voltage and current waveform data is being processed outside of phasor-measurement-unit (PMU) data. This may be partially due to the relative lack of readily available data for research. Some techniques have been developed that use contextual information [4] [5] such as weather, affected phase(s), season, event time, and interrupting device. Other techniques use extracted features from waveform data [6] [7] such as the derivative of current and voltage signal, energy, amplitude, correlation coefficient, etc. Self-recoverability, zero current time, degree of distortion, transition time and waveform randomness, are extracted from the recorded waveforms in [8], and used as the input to a fuzzy inference.

PMU data does not provide sufficient detail of the waveforms the low 60Hz sample rate (1 sample per cycle). Using waveforms as a direct source to analyze system events requires higher resolution than PMU data provides. High-resolution event recordings are highlighted by [9] and has been implemented in some power systems [10]. Point-on-wave (POW) data, features a sample rate of 1 kHz or higher. POW data provides more detail in the waveforms.

2.1 AI/ML FOR PROTECTION AND THE ADAPTIVE RELAY

The broad value of the work was in waveform processing by allowing for the advancement of protection for the power grid. Ultimately, plug and play protection and self-healing on the power system was the goal. The work aimed to eliminate engineering studies required to find protection settings that protect the power system equipment and the public.

Voltage and current waveform data were fed into NNs. These NNs reduce the data to actionable information. The illustration and brief definitions below are intended to provide some high-level clarity into the terminology.

- **Artificial Intelligence (AI):** A characteristic of a machine or software application that allows for mimicry of human behavior and displays decision-making capabilities.
- **Machine Learning (ML):** A subset of AI, ML is the process by which AI is incorporated into a machine or software package. This process specifically involves teaching how to learn from previously-encountered data-sets.
- **Neural Network (NN):** An extension of ML involving learning highly complex, non-linear functions to perform regression and/or classification tasks
- **Artificial Neural Network (ANN):** Often used for regression and classification problems[1]
- **Convolutional Neural Network (CNN):** Often used for image processing, CNNs employ the idea of "convolution" from signal processing to extract relevant features from localized regions in the input data set.
- **Recurrent Neural Network (RNN):** RNNs use "hidden nodes and layers" to learn temporal features in a data set and are typically best suited for time-series learning.

Today, protection engineers choose protection settings that define protection curves on a Time-Current Curve (TCC) (see
Figure 5). This approach works for radial circuits with strong utility sources. Long circuits may exhibit low available fault current at the end of the line.[2] Additionally, modern distribution systems with Distributed Energy Resources (DER) such as wind, solar and batteries are becoming more prevalant. In these and other contexts, the conventional approaches to protection are facing new challenges. The motivation for using machine learning in protection was to investigate possible alternatives to conventional protection curves that are used today.

[1] In this context ANNs are referring to fully connected neural networks which are distinct from CNNs, RNNs and other structures.

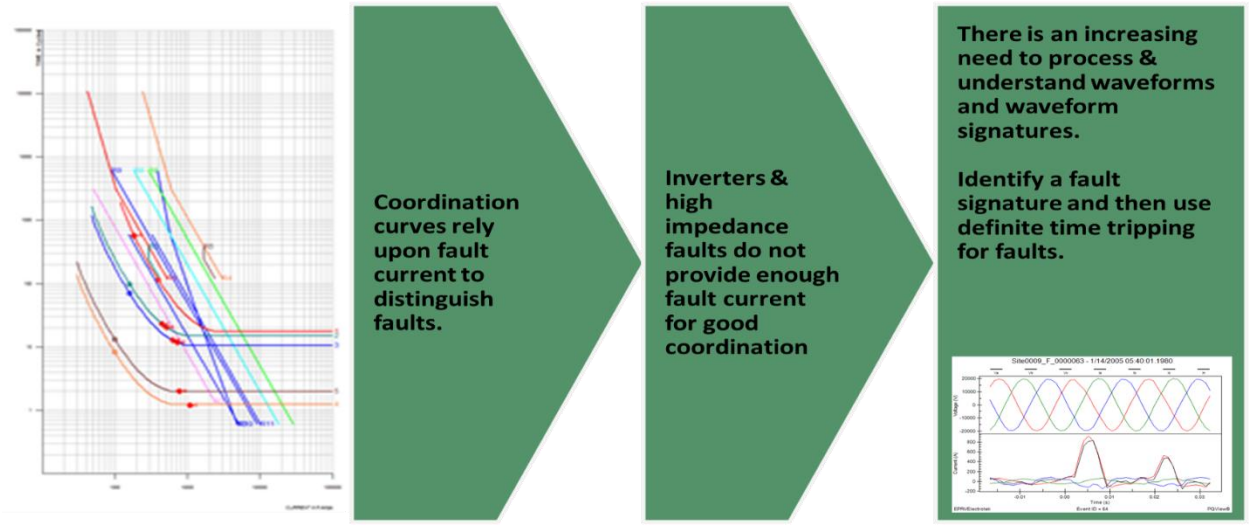


Figure 5 Evolution of Protection with Waveforms and Waveform Signatures

The machine learning work started under the Microgrid[3] project funding pertains to the concept of an Adaptive Relay (AR). The work investigated the potential of incorporating machine learning and artificial intelligence for functions that would be helpful in the AR. ORNL was in partnership with Sandia National Lab.

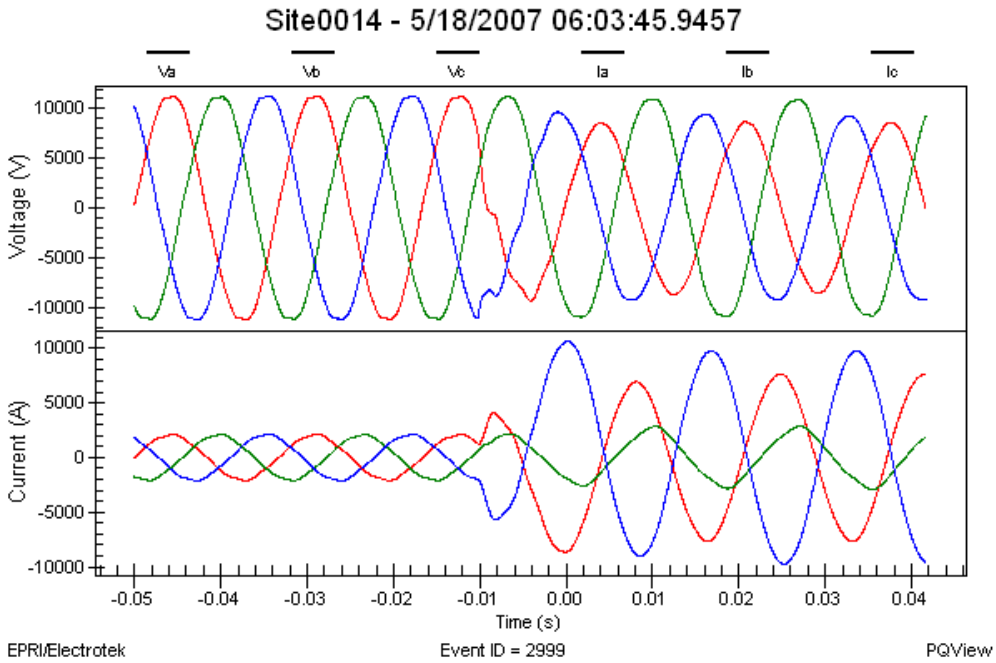


Figure 6 Power System Waveform Event Capture

[3] Travis Smith was the PI for the overall Microgrid Adaptive Relay work

Power system waveform data such as that in Figure 6 was to be processed using ML. The voltage and current waveform data are used for input into a NN. The NN reduces the input data to actionable information (see Figure 7). Actionable information may include fault detection/classification, microgrid protection, etc.

Information that can be derived from waveform data includes (but is not limited to):

- Power factor
- Direction of power/current
- Unfiltered/filter the waveforms
- Instantaneous V, I, Z, P, Q, S
- RMS V, I, Z, P, Q, S
- Symmetrical components
- Clark transforms
- Park transforms
- Harmonics THD V, I
- Harmonics individual V, I
- SAG
- Swell
- LG fault
- LL fault
- LLG fault
- LLL (three phase) fault
- Capacitor ringing
- Switching (closes/opens)
- Min, Max, Avg
- Voltage present
- Ride through
- Electrical distance to a fault

The above is not an exhaustive list but serves as a sample of things that waveform data can be used for.

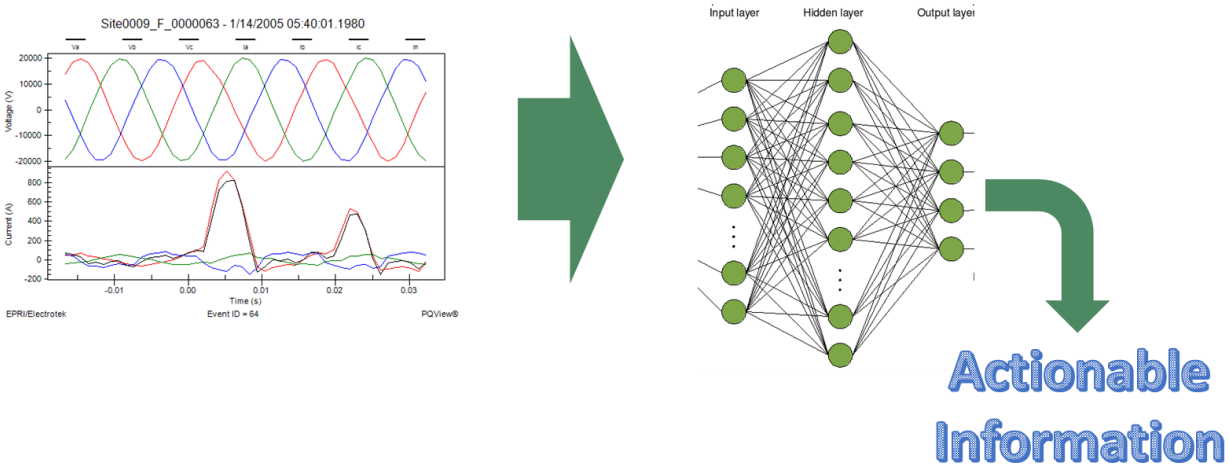


Figure 7 Waveform to Neural Network to Actionable Information

The training of the neural networks can take hours or days on a conventional computer for each network structure investigated. An NVIDIA DGX machine specifically designed and built for artificial intelligence training has been set up at ORNL. The machine is configured for optimal use in training the algorithms being investigated.

One of the tasks was to investigate if machine learning could be used to identify which circuit breaker operated (opened/closed) based on the non-simultaneity signature of the circuit breaker. The term "non-simultaneity" refers to the slight timing difference between when each
pole of the 3-phase circuit breaker opening (see Figure 8).

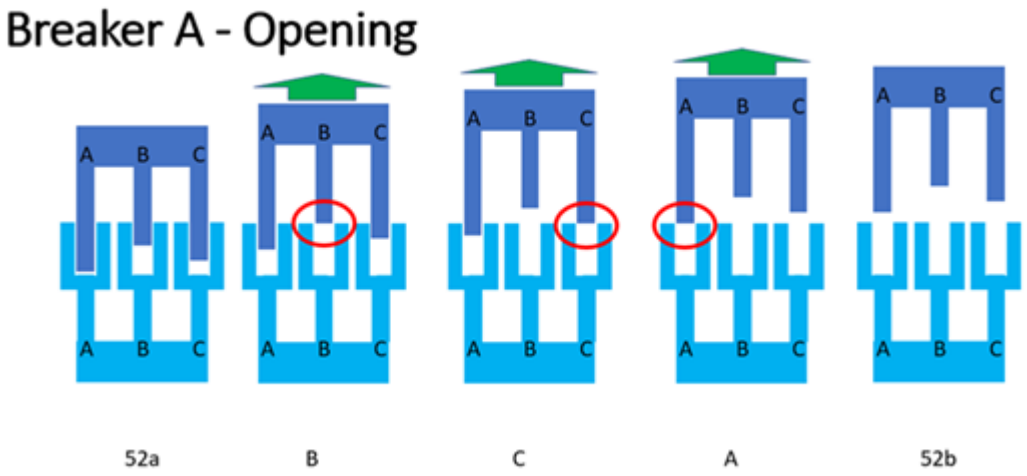


Figure 8 Breaker Non-Simultaneity

To analyze non-simultaneity, the simpler task of fault classification was used as a starting point. In fault classification, a dataset of simulated faults was created for the project (AG, BG, CG, AB, BC, AC, ABC, ABG, BCG, ACG and ABC where G is "Ground" and A, B, C are phase labels). The initial results were around 14% accuracy (see Figure 9). With some modifications to the structure of the NN (see section 5 below), the accuracy increased to >99% with the simulated data (see Figure 10).

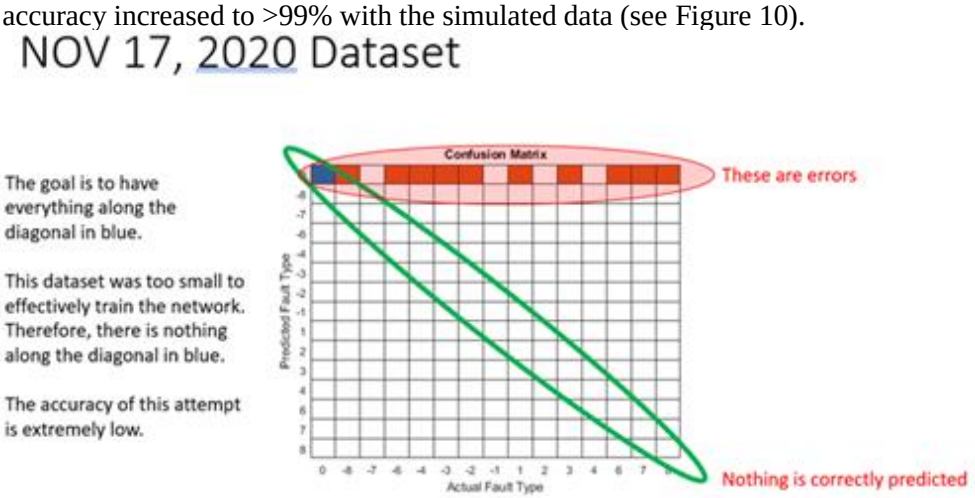


Figure 9 Early Results of Neural Net. for Waveform Classification

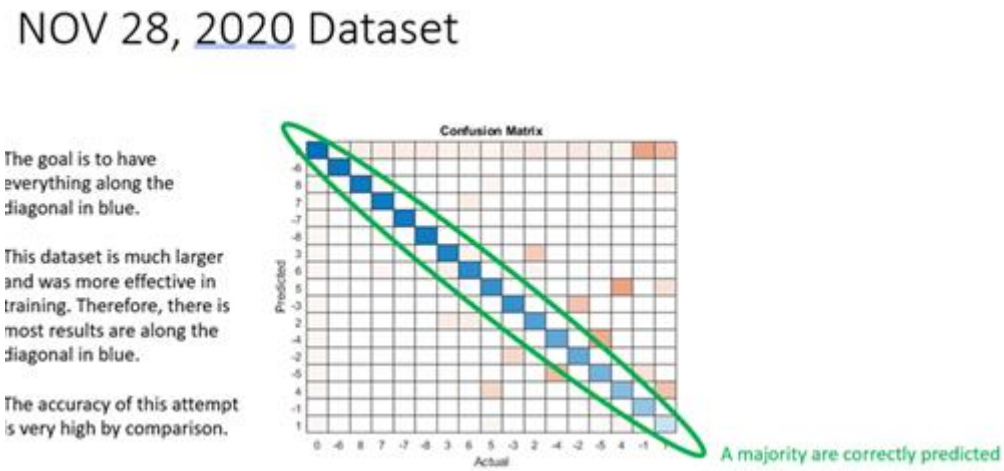


Figure 10 Improved Results of Neural Net. for Waveform Classification

The work progressed to the non-simultaneity task. A different NN structure was developed for the task which achieved accuracy of >85% if looking at the waveform data from the recloser of interest. The investigation exposed how looking at the waveform data from the substation rather than waveform data from the recloser of interest resulted in lower accuracy. This is because the waveforms, obtained from the view of the substation, get washed out the further the waveform travels which was not initially considered. This is due to the filtering effects of the resistive, inductive, and capacitive effects of the lines.

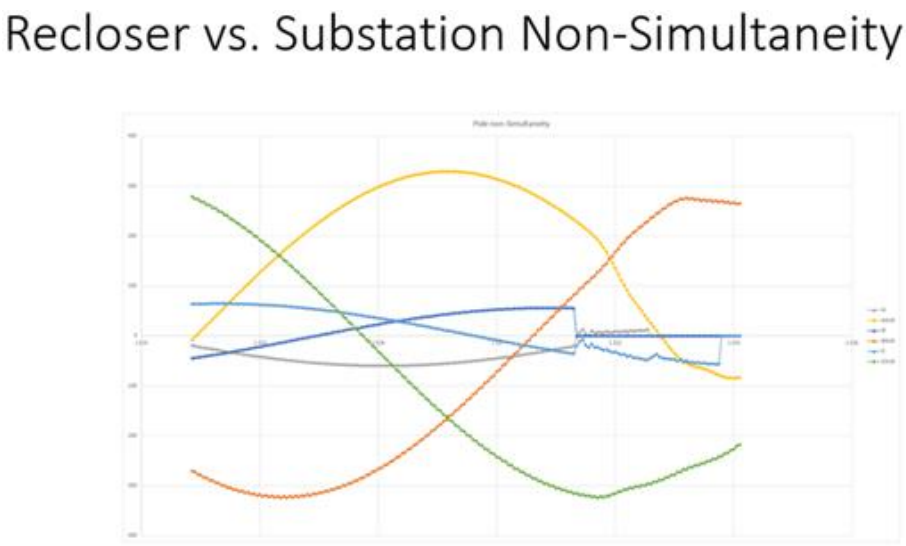


Figure 11 Switching Waveform at Station vs at Recloser (x-axis: time, y-axis: signal amplitude)

The washout effect can be seen in Figure 11. The grey, blue and light blue curves are the waveforms at the recloser and the green, yellow and orange waveforms are the same event at the substation, respectively. At the recloser the "step" shapes in the waveforms are sharp and clear. At the substation, the waveforms are washed out. One reason for the difference is the amplitude of the load at different points of the circuit, but this does not explain the loss of "step" shape in the waveform.

2.2 HOW IS CLASSIFICATION PERFORMED TODAY?

EPB of Chattanooga is a partner utility with ORNL. EPB has an extensive deployment of S&C Intellirupters that are recording event captures that consist of voltage and current waveforms. They have been developing tools to help investigate the event capture waveforms. The concept that EPB is working on is something they call "speed of thought". The idea is to narrow down what the event might be sufficiently for an engineer or operator to quickly determine for themselves how to classify the event.

Today, individual tests are run to identify various system conditions/events. Each test gives a result of pass, fail, yes, no, true or false. There is no conclusion other than it may be one system condition/event or another based on a test having a true, yes or pass value. Figure 12 shows a set of tests and results for a single event to illustrate the point.

TaskID	Device	DateTime	Group	GBBID	Bucket	Bin	TestResult
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group1	GBB101	WaveformPresent	FiveCycles	pass
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group1	GBB102	VoltagePresent	TwoDirections	pass
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group1	GBB105	VoltagePresent	OneDirection	pass
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group2	GBB223	NoBucket	NoBin	yes
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group3	GBB301	PowerQuality	persist	no
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group3	GBB302	PowerQuality	case1SAG	fail
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group3	GBB303	PowerQuality	case2SAG	fail
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group3	GBB304	PowerQuality	case3SAG	fail
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group3	GBB329	Fault	Short	yes
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group3	GBB333	PowerQuality	case4SAG	TRUE
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group5	GBB507	Harmonics	current	yes
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group5	GBB508	Harmonics	voltage	yes
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group6	GBB601	MinMaxAvg	General	yes
20410002	EBR202-Z91334	2018-08-01 03:17:35.7526884	Group7	GBB700	IEEE1668RideThrough	NoSag	NoPlot

Figure 12 Tests and Results for a Single Event Waveform

The tests are a tremendous improvement over what was done in the past (often nothing). In the past there were rarely waveforms to analyze and thus engineers did not rely on waveforms to try to identify events. The utility often did not worry about things unless they received a complaint from a customer. Once a complaint was registered, then one of the more savvy and technical engineers would place a recording power quality meter to try to capture some events (waveforms) to further analyze. These were typically one-off investigations.

Many modern reclosers and circuit breakers have microprocessor relays that record waveforms of system events. In some cases, utilities may record a half-a-dozen event captures for every event. This is thousands of events per year. So, the algorithms illustrated in Figure 12 are a large leap forward from where the industry was 10-20 years ago. With that said, the industry will need faster, more automated, more conclusive, and easier to use systems going forward that can process massive amounts of event recordings without extensive input/support from power system engineers.

Several techniques for fault classification have been developed such as [11] [12]:

- Wavelet approach
- Artificial neural network approach
- Fuzzy logic approach
- Neuro-fuzzy technique
- Wavelet and ANN technique
- Wavelet and fuzzy-logic technique
- Wavelet and neuro-fuzzy technique
- Support vector machine
- Genetic algorithm
- DWT-ELM approach
- Theory and FPGA-based implementation
- GSM technique
- PMU-based protection scheme
- Decision tree-based method
- Multi-information measurements
- Fast estimation of phasor components
- PCA based framework
- Pilot scheme
- Functional analysis and computational intelligence
- Euclidean distance-based function
- Pattern recognition approach
- Comparison and conclusion

Most of these techniques were developed with protection in mind. Many of them have been deployed in commercial software and/or equipment, but few (if any) of these techniques fit for general waveform classification such as identifying the cause of the fault etc. Spectrograms have been used as well as CNN as in [13]. The notable contributions of this work include:

- Use of spectrograms on combined voltage and current waveforms for three phases
- Use of a power system filter-bank to concentrate on power system harmonic information

- Waveform signal normalization for the voltage and current waveforms
- Processing the combined spectrograms by applying a logarithm

2.3 DARKNET MACHINE LEARNING OF POWER SYSTEM WAVEFORM SIGNATURES

The processing of power system waveforms using machine learning moved to the Darknet project. Under Darknet the focus was expanded beyond protection to include any system events or anomalies captured in waveforms.

The previous protection neural network investigations confirmed that unique neural networks could be trained to identify faults and other system protection events. However, if time-series waveform data combined with Recurrent Neural Networks (RNN) were used, then it would necessitate a unique NN for each event type.

The lesson learned from the previous tests for waveform processing and the protection neural network investigations was that unique code to identify individual system conditions would quickly become cumbersome and infeasible in the long term with respect to commercially deployable solutions. Therefore, a solution that could be adapted to a wide variety of system events is desirable. For example:

- Switching,
- Reach,
- Reclosing,
- Harmonics,
- Motor starting,
- Phase unbalance,
- Capacitor switching,
- LTC/Regulator tap changes,
- Cold transfers (drop and pick),
- Hot transfers,
- Inrush/transformer energization,
- etc

From the perspective of commercially viable solutions, it is important to find a universal NN structure (deep learning algorithm) that works for a wide variety of system event types.

2.4 POWER SYSTEM WAVEFORM PROCESSING

The fields of audio engineering, speech and audio processing use spectrograms fed into CNNs to perform tasks such as word recognition, speaker recognition, and cleaning noise from the signal of interest. These are all examples that have direct equivalents in power system waveforms and waveform “signatures”.

Audio signals are very similar to power system signals other than the fact that the frequency of audio signals tends to be quite high where is the frequency of power system waveforms tend to be quite low. Additionally, audio waveforms are often mono signals which result in a single waveform, whereas power system waveform data exists in three phases of current data and three phases of voltage data, yielding six waveforms altogether.

In audio signal processing, the audio waveform data is converted to an image using spectrograms [14], [15]. Spectrograms are plots of short time Fourier transforms. Short time Fourier transforms are time sampled Fourier transforms of waveform data. In other words, a Fourier transform is performed on a snippet of waveform data (a time step). The next snippet of waveform data is also converted to a Fourier transform and so forth and so on until the entire waveform is converted to samples of Fourier transforms. Each Fourier transform results in a harmonic spectrum of intensities for each harmonic frequency that exists within that snippet.

Each of the snippets has a harmonic spectrum associated with it that describes all the frequency components of that snippet. The frequency spectrum of each snippet is horizontally stacked next to each other for all snippets of the overall waveform. This creates a two-dimensional array of data where each column represents a snippet of time within the waveform and each row represents a harmonic component (see Figure 13).

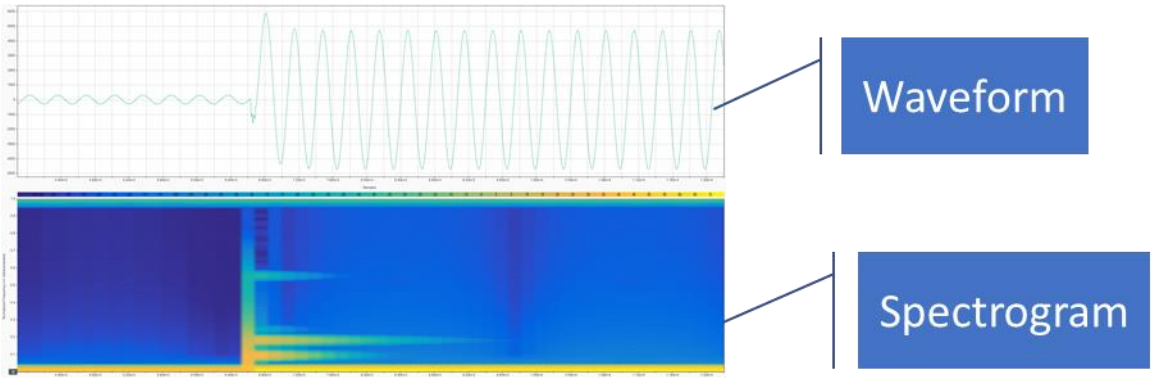
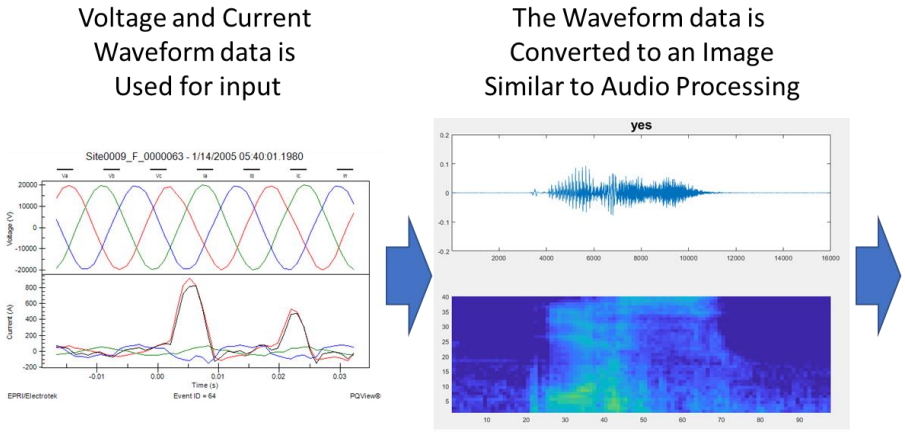


Figure 13 Spectrogram of a Waveform

The array of values can be interpreted as an image where at or near the bottom of the image are the low frequency values and the of the top the image are the high-frequency values of the harmonic components for each time step (see Figure 14).

This image of the Short Time Fourier transform can be passed into a CNN. In the case of power system data some additional modification of the data is required before passing it to a CNN for training (see Figure 15).



Waveform data is converted to an image by way of spectrogram. The spectrogram image is then processed using a Convolutional Neural Network to reduce to actionable information. Actionable information may include: Fault Detection/Classification, Microgrid Protection, etc.

Figure 14 Convert Power System Waveform to a Spectrogram Image Similar to Audio Processing

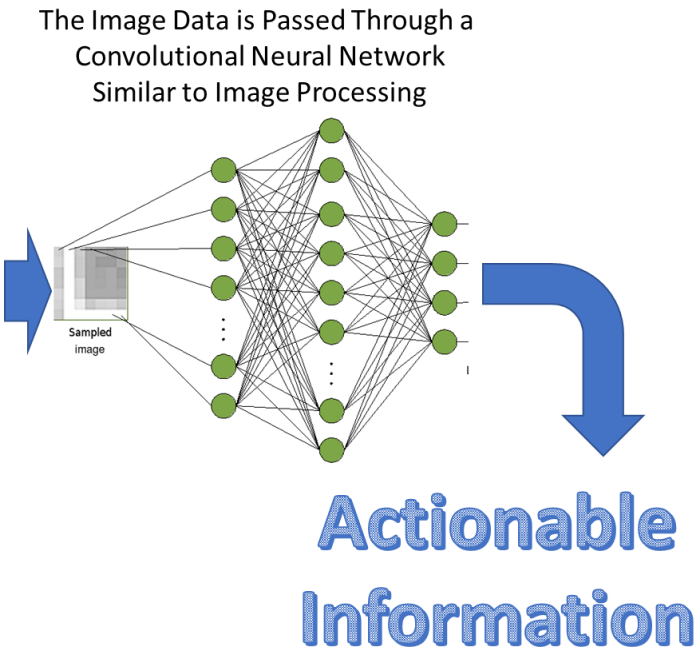


Figure 15 Pass the Spectrogram Image Through a Convolutional Neural Network to Extract Actionable Information

A waveform signature could be treated as a spoken word like "YES" or "UP". No two people produce the exact same waveform when speaking each of these words, but the algorithm can still distinguish the word regardless of the speaker. No two circuits will produce the exact same waveform for a given fault type (AG for example), but the NN can be trained to classify the fault type regardless of the circuit or location on the circuit.

3. EVENT WAVEFORM DATA

A Signature Library is in development to house various system events for use in training NN algorithms conceptually similar to ImageNet [16], but for power system event waveform files. The data currently includes simulated events, motor starts and breaker pole non-simultaneity events. Field data is being collected and used from distribution events recorded by EPB of Chattanooga [10] and from the EPRI/DOE [17] event library.

The DOE/EPRI National Database Repository of Power System Events (https://pqmon.epri.com) contains approximately 300 event recordings with event type classifications such as tree, weather, equipment, etc. (Figure 16). Each event recording includes event files that include (Figure 17): images of waveforms and raw datafiles in various formats. Each event also includes event properties with more detail (Figure 18)

DOE/EPRI National Database Repository of Power System Events

EPRI ELECTRIC POWER RESEARCH INSTITUTE

Disturbance Library Home

Click the EventId to see event waveforms and download data.

EventId	SiteName	FeederId	Phase	First Event	Last Event	Cause	Weather	Isolation Equipment Code	Failed Equipment Code	Details
0001	Site0009	F_0000065	2	2005-01-01 10:10:34.487	2005-01-01 10:10:39.870	Tree	Clear Weather	Recloser	LINE CUT	Fault caused line recloser lockout. Tree Outside Right of Way (Fall/Lean On Primary)
0004	Site0009	F_0000065	2	2005-01-01 10:10:34.487	2005-01-01 10:10:39.870	Tree	Clear Weather	Recloser	LINE CUT	Fault caused line recloser lockout. Tree Outside Right of Way (Fall/Lean On Primary)
0005	Site0009	F_0000065	2	2005-01-01 10:10:34.487	2005-01-01 10:10:39.870	Tree	Clear Weather	Recloser	LINE CUT	Fault caused line recloser lockout. Tree Outside Right of Way (Fall/Lean On Primary)
0007	Site0009	F_0000065	2	2005-01-01 10:10:34.487	2005-01-01 10:10:39.870	Tree	Clear Weather	Recloser	LINE CUT	Fault caused line recloser lockout. Tree Outside Right of Way (Fall/Lean On Primary)
3042	Site0006	F_0000041	4	2005-01-04 03:39:06.000	2005-01-04 03:39:06.000	Equipment	Unknown	Fuse	Other	Equipment, Device UG, Damaged.
0021	Site0009	F_0000061	1	2005-01-07 19:20:10.599	2005-01-07 19:20:10.613	Equipment	Clear Weather	CO15	FUSE	Overhead Insulator Failure. BROKEN INSULATOR
0022	Site0009	F_0000061	1	2005-01-07 19:20:10.599	2005-01-07 19:20:10.613	Equipment	Clear Weather	Fuse CO15	FUSE	Overhead Insulator Failure. BROKEN INSULATOR
0062	Site0009	F_0000065	4	2005-01-14 05:39:59.364	2005-01-14 05:40:01.198	Undetermined	Raining	CO15LB	FUSE	storm
0064	Site0009	F_0000065	4	2005-01-14 05:39:59.364	2005-01-14 05:40:01.198	Undetermined	Raining	CO15LB	FUSE	storm
0067	Site0009	F_0000065	4	2005-01-14 07:22:52.495	2005-01-14 07:22:52.828	Tree	Thunderstorm	CO15LB	FUSE	Tree/Limb Growth
0065	Site0009	F_0000065	4	2005-01-14 07:22:52.495	2005-01-14 07:22:52.828	Tree	Thunderstorm	CO15LB	FUSE	Tree/Limb Growth
0068	Site0009	F_0000065	2	2005-01-14 08:07:18.573	2005-01-14 08:07:18.573	Tree	Clear Weather		CUSTOMER	VINES ON TRANSFORMER
2760	Site0010		1	2005-01-18 05:41:21.000	2005-01-18 05:41:21.000	Unknown	Unknown	Unknown	Unknown	Short duration variation. No outage information found.
3048	Site0012	F_0000037	3	2005-01-25 03:29:36.000	2005-01-25 03:29:51.000	Equipment	Unknown	Breaker	Capacitor	Equipment, Capacitor Station, Damaged.
2761	Site0010	F_0000032	4	2005-01-30 16:02:34.000	2005-01-30 16:02:34.000	Equipment		Breaker		A conductor off the pin on the mainline of F_0000032 out of Site0010 Sub caused the circuit breaker to trip.
2762	Site0010	F_0000029	2	2005-02-02 15:10:14.000	2005-02-02 15:10:14.000	Equipment		Breaker		A cable fault on the underground portion of F_0000029 out of Site0010 Sub. caused the circuit breaker to trip.

Figure 16 DOE/EPRI National Database Repository of Power System Events [17]

Event Files

Portable Network Grapics Image File	0001.png
JPEG Image File	0001.jpg
Windows Bitmap File	0001.bmp
Windows Metafile	0001.wmf
Enhanced Metafile	0001.emf
Comma Separated Variable File	0001.csv
IEEE Std 1159.3 PQDIF File	0001.pqd
IEEE Std 37.111 COMTRADE Files	0001.cfg | 0001.dat
Circuit Model	Site0009_F_0000065.txt

Figure 17 DOE/EPRI Event Files [17]

Event Properties

Event ID	1
Event Name	Site0009_F_0000065 - 1/1/2005 10:10:34.4870
Event Type	
Cause	Tree
Failed Equipment Code	LINE CUT
Isolation Equipment Code	Recloser
Weather	Clear Weather
Details	Fault caused line recloser lockout. Tree Outside Right of Way (Fall/Lean On Primary)

Figure 18 DOE/EPRI Event Properties [17]

3.1 SIMULATED EVENT FILES

EPB Chattanooga provided a circuit model of one of their distribution circuits in CYMEdist format (Figure 19). ORNL[4] converted the EPB circuit model from CYMEdist format into a MATLAB/Simulink model[5]. The Simulink model was augmented to simulate motor starts of various sizes and types and save waveform files of the motor starting events[6]. The augmented model was further modified[7] to simulate several system events such as faults (phase-to-ground, phase-to-phase, phase-to-phase-to-ground, and three-phase).

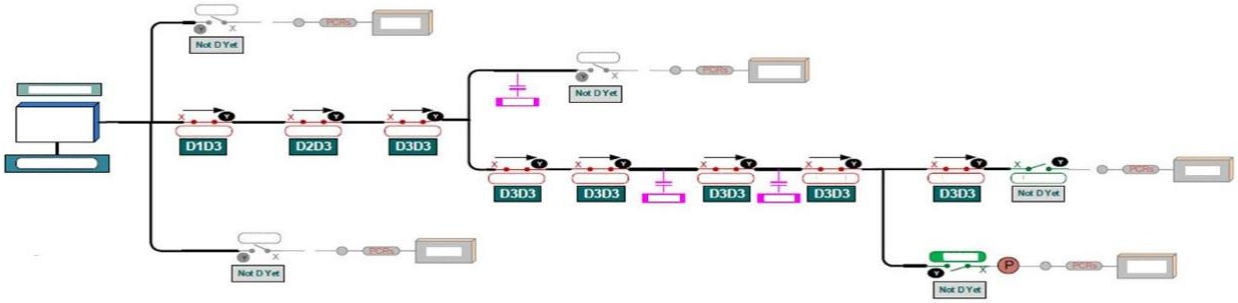


Figure 19 Distribution Circuit - Basis for Simulated System Events

Several event files were created over 2019, 2020 and 2021 including motor start, bolted fault events, breaker non-simultaneity, and current transformer saturation. The files were labeled in different ways depending on the project that the files were intended for. Some files were labeled at a file level while others included relevant labeling at each timestep in the event recording. Many of the simulated event files were prepared to be loaded into the Signature Library that is under development at ORNL.

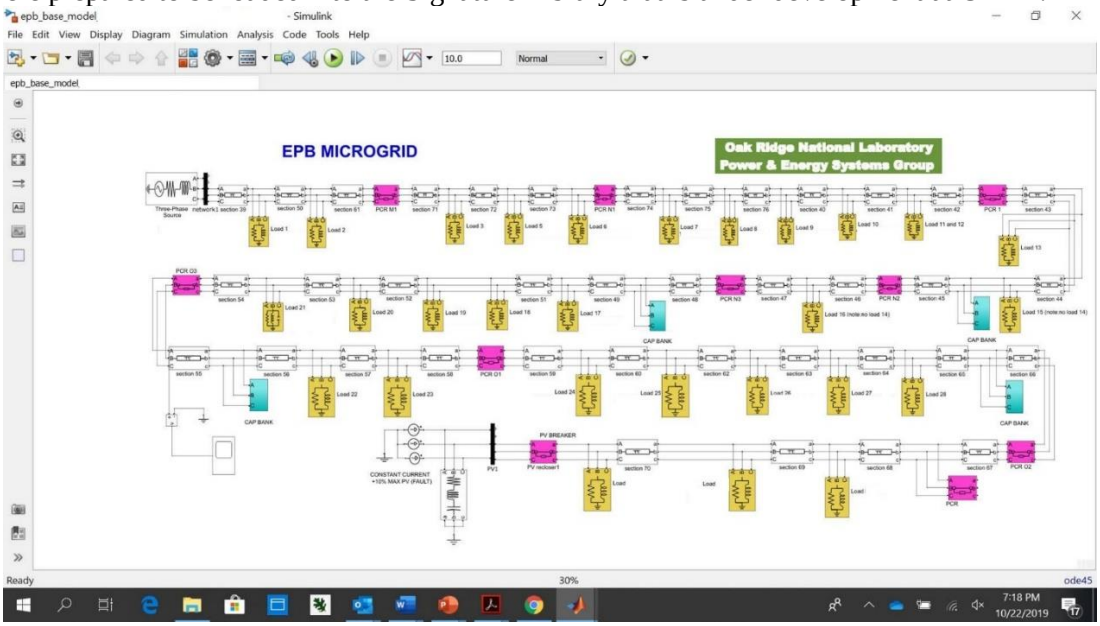


Figure 20 MATLAB/Simulink Model of Distribution Circuit

[4] Oak Ridge National Laboratory (ORNL)

[5] Mike Marshall and Neil Sheppard, of ORNL, converted the original CYMEdist model to MATLAB/Simulink model

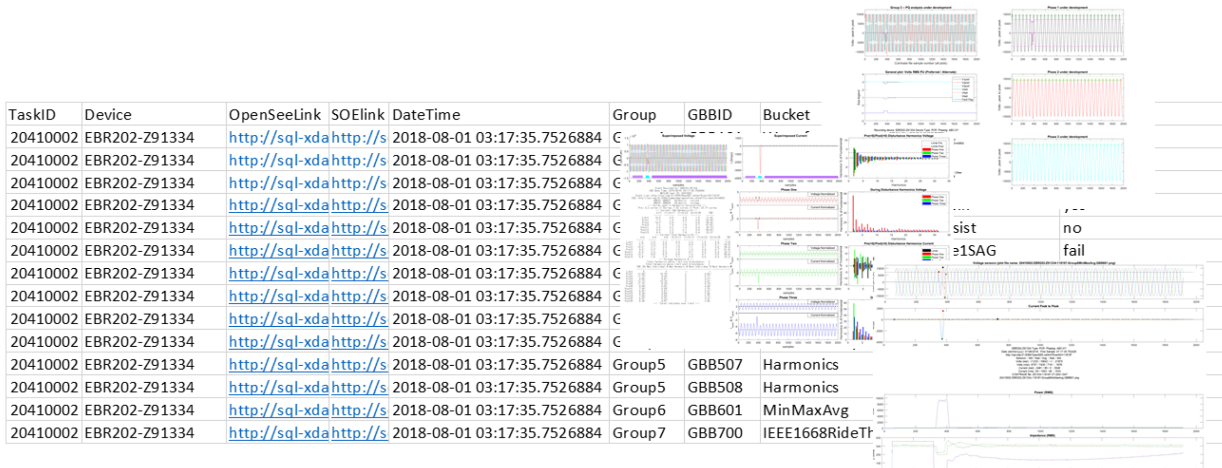
[6] Emilio Piesciorovsky, of ORNL, augmented the MATLAB/Simulink model with motors.

[7] Isabelle Snyder, of ORNL, augmented the MATLAB/Simulink model to simulate a variety of system events.

3.2 EPB EVENT DATA

Under a Non-Disclosure Agreement EPB Chattanooga provided nine months of event capture waveforms from across their system. As part of a federal grant EPB installed many S&C Electric Company IntelliRupter® reclosing devices across their system with 1000 foot spacing between the reclosing devices on distribution circuits. All the IntelliRupter® reclosing devices communicate back to the control center over fiber-optic communication system. Each of the reclosing devices has a microprocessor-based relay that is configured to capture and record waveforms for system events (see Figure 21).

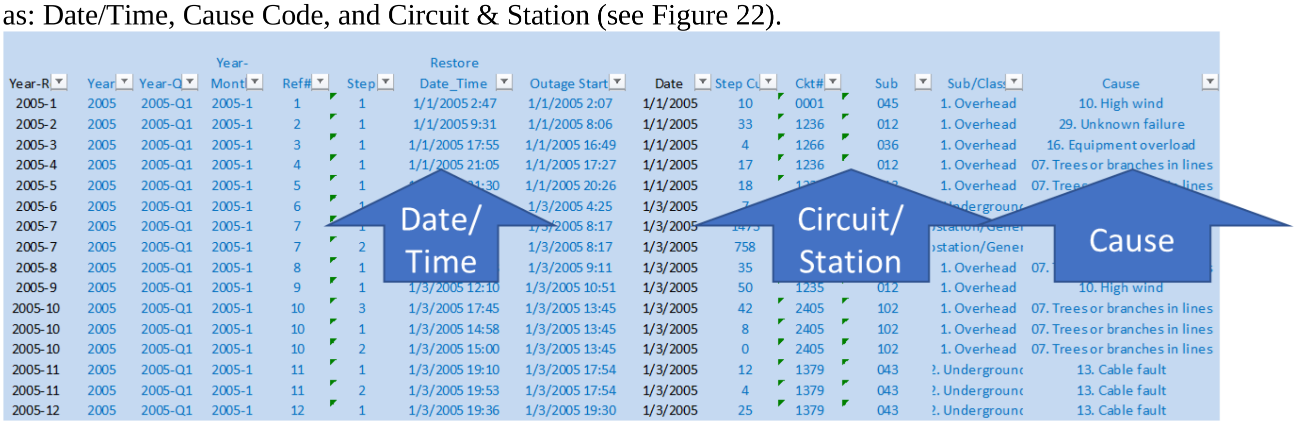
- Each event includes waveform capture data in the COMTRADE format
- Each event waveform data is passed through a series of tests to attempt to classify the event.



TaskID	Device	OpenSeeLink	SOElink	DateTime	Group	GBBID	Bucket		
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	G	[illegible]	[illegible]	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	G	[illegible]	[illegible]	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	G	[illegible]	[illegible]	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	G	[illegible]	[illegible]	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	G	[illegible]	[illegible]	sist	no
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	G	[illegible]	[illegible]	e1SAG	fail
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	G	[illegible]	[illegible]	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	G	[illegible]	[illegible]	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	G	[illegible]	[illegible]	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	G	[illegible]	[illegible]	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	Group5	GBB507	Harmonics	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	Group5	GBB508	Harmonics	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	Group6	GBB601	MinMaxAvg	[illegible]	[illegible]
20410002	EBR202-Z91334	http://sql-xda	http://s	2018-08-01 03:17:35.7526884	Group7	GBB700	IEEE1668RideTh	[illegible]	[illegible]

Figure 21 Sample Listing of Waveform Event Captures

Utilities maintain outage records in order to calculate reliability metrics and report to the state public utility commission. The outage records are often maintained in Outage Management Systems (OMS). The outage records include information that is useful in cross-referencing with waveform event captures such as: Date/Time, Cause Code, and Circuit & Station (see Figure 22).



Year-R	Year	Year-Q	Year-Mont	Ref#	Step	Restore Date_Time	Outage Start	Date	Step Cu	Ckt#	Sub	Sub/Class	Cause
2005-1	2005	2005-Q1	2005-1	1	1	1/1/2005 2:47	1/1/2005 2:07	1/1/2005	10	0001	045	1. Overhead	10. High wind
2005-2	2005	2005-Q1	2005-1	2	1	1/1/2005 9:31	1/1/2005 8:06	1/1/2005	33	1236	012	1. Overhead	29. Unknown failure
2005-3	2005	2005-Q1	2005-1	3	1	1/1/2005 17:55	1/1/2005 16:49	1/1/2005	4	1266	036	1. Overhead	16. Equipment overload
2005-4	2005	2005-Q1	2005-1	4	1	1/1/2005 21:05	1/1/2005 17:27	1/1/2005	17	1236	012	1. Overhead	07. Trees or branches in lines
2005-5	2005	2005-Q1	2005-1	5	1	[illegible]	1/1/2005 20:26	1/1/2005	18	[illegible]	[illegible]	1. Overhead	[illegible]
2005-6	2005	2005-Q1	2005-1	6	[illegible]	[illegible]	1/3/2005 4:25	1/3/2005	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2005-7	2005	2005-Q1	2005-1	7	[illegible]	[illegible]	[illegible]	1/3/2005	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2005-7	2005	2005-Q1	2005-1	7	2	[illegible]	1/3/2005 8:17	1/3/2005	758	[illegible]	[illegible]	[illegible]station/Gener	[illegible]
2005-8	2005	2005-Q1	2005-1	8	1	[illegible]	1/3/2005 9:11	1/3/2005	35	[illegible]	[illegible]	1. Overhead	[illegible]
2005-9	2005	2005-Q1	2005-1	9	1	1/3/2005 12:10	1/3/2005 10:51	1/3/2005	50	1235	012	1. Overhead	10. High wind
2005-10	2005	2005-Q1	2005-1	10	3	1/3/2005 17:45	1/3/2005 13:45	1/3/2005	42	2405	102	1. Overhead	07. Trees or branches in lines
2005-10	2005	2005-Q1	2005-1	10	1	1/3/2005 14:58	1/3/2005 13:45	1/3/2005	8	2405	102	1. Overhead	07. Trees or branches in lines
2005-10	2005	2005-Q1	2005-1	10	2	1/3/2005 15:00	1/3/2005 13:45	1/3/2005	0	2405	102	1. Overhead	07. Trees or branches in lines
2005-11	2005	2005-Q1	2005-1	11	1	1/3/2005 19:10	1/3/2005 17:54	1/3/2005	12	1379	043	2. Undergroun	13. Cable fault
2005-11	2005	2005-Q1	2005-1	11	2	1/3/2005 19:53	1/3/2005 17:54	1/3/2005	4	1379	043	2. Undergroun	13. Cable fault
2005-12	2005	2005-Q1	2005-1	12	1	1/3/2005 19:36	1/3/2005 19:30	1/3/2005	25	1379	043	2. Undergroun	13. Cable fault

Figure 22 Sample Listing of Outage Management System Data

SQL Queries were constructed to cross-reference the OMS data with the descriptive metadata of the waveform captures provided by EPB. The queries resulted in selecting the waveforms that have corresponding event cause codes in the OMS records. This narrowed down the total collection of event waveforms to those with verifiable cause codes. This set of labeled data is used for developing, training, and testing of the various machine learning algorithms.

4. PREPROCESSING WAVEFORM DATA

Several techniques exist for preprocessing waveform or RMS data. The techniques investigated for the previous protection work include symmetrical components, Clark/Park transform, cycle to cycle difference, RMS magnitude, Fourier transform. These techniques among others were considered and compared as part of the initial investigation into preprocessing the waveform data before sending it into a neural network. The solutions were programmed into the MATLAB/Simulink model illustrated in Figure 23.

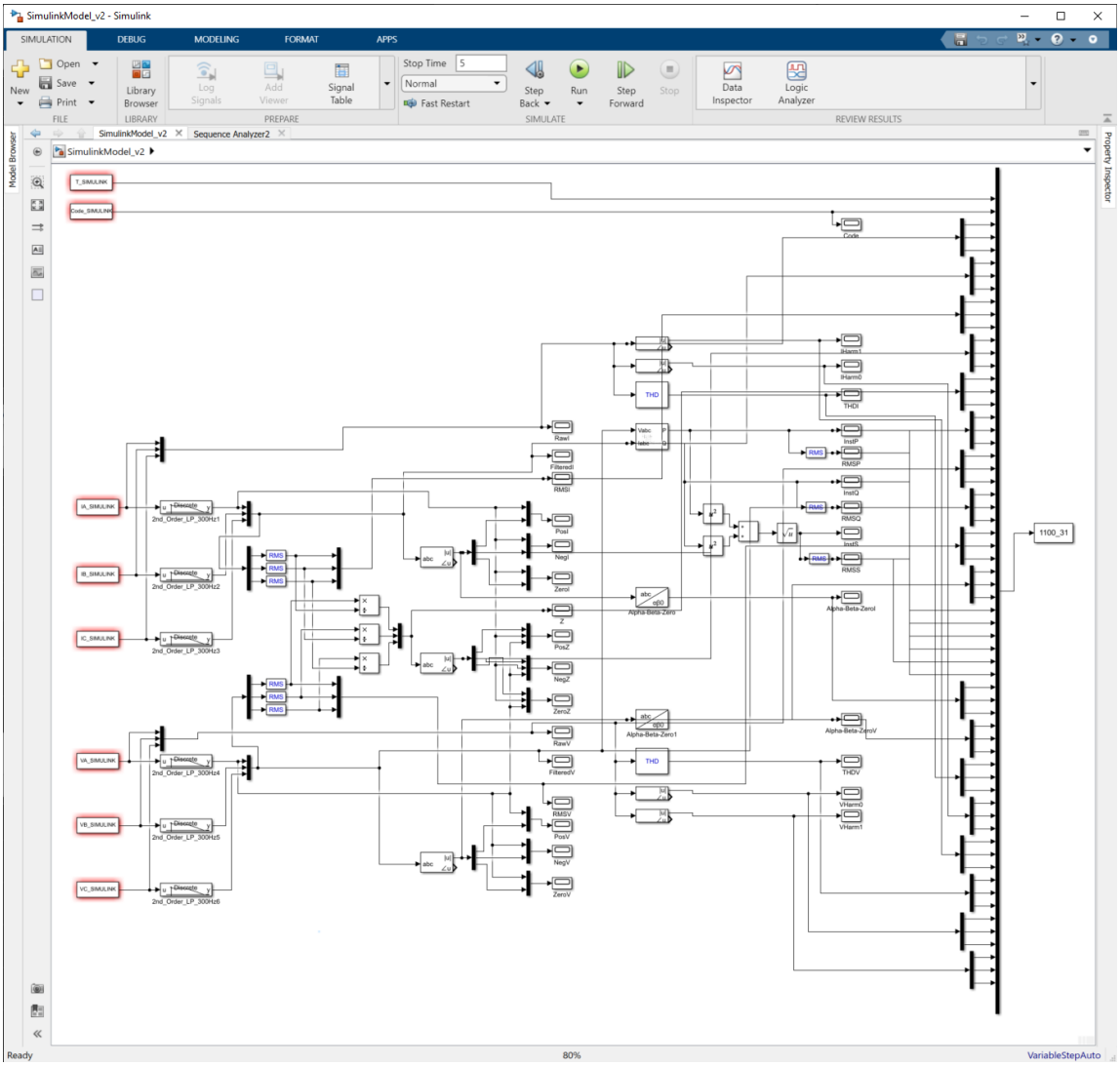


Figure 23 MATLAB/Simulink Model for Preprocessing Waveform Data

Many of these techniques such as symmetrical components are frequently used in power system protection and analysis. The conclusion that was drawn from these techniques is that there is a time cost in the form of a calculation delay (see Figure 24). The time delay is not related to the computation time but rather in waiting for the data of all three phases that is required. For example, if doing a cycle-to-cycle magnitude difference calculation then a full cycle must pass in order to perform the calculation. In the case of symmetrical components two thirds of a cycle must pass in order to perform asymmetrical components calculation.

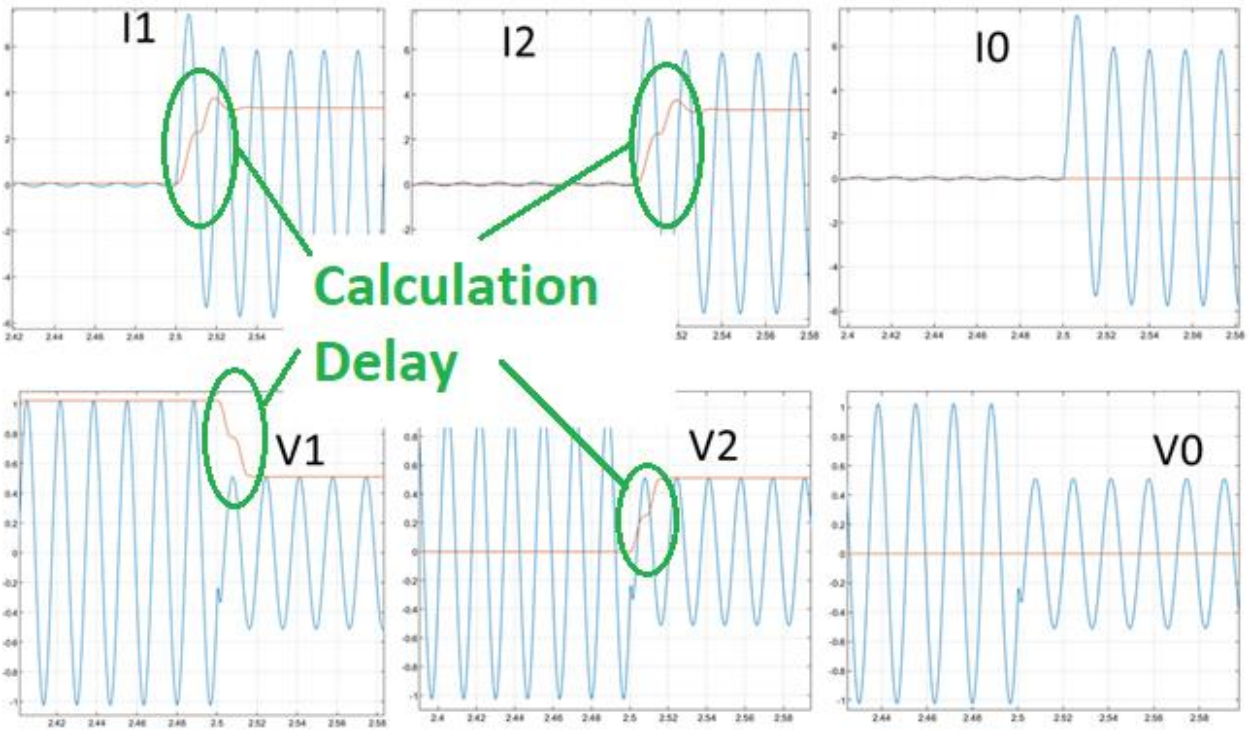


Figure 24 Preprocessing of Waveform (blue) with Symmetrical Components Output (red) (x-axis: time, y-axis: signal amplitude)

One conclusion reached in the earlier protection work is that it is possible to get good results from a Long Short-Term Memory (LSTM) NN in classifying the event without preprocessing the waveform data. Another conclusion reached in the protection work is that a different LSTM neural network structure was required for each system condition that was of interest. One neural network structure was required for fault classification and a different neural network structure was required for CT saturation. In learning that different neural network structures are required for different problems it became apparent that a LSTM neural network approach could ultimately require several different neural networks for all the different system conditions of interest in power systems. Therefore, a generalized neural network structure that could be trained on a wide variety of power system conditions such as faults, motor starts and harmonics, etc. is desirable for commercially deployable solutions.

5. SEQUENCE-TO-SEQUENCE NEURAL NETWORKS

Sequence-to-sequence (S2S) neural networks were investigated as part of the initial development of neural networks for power system waveform data. Sequence to sequence refers to a determination of the state of the circuit at each time step of the data file. This is in contrast to sequence classification where there is a single determination for the entire data file. The investigation started with a MATLAB example problem that shows how to classify each time step of the sequence using a LSTM network on a data set of human body movement [18].

5.1 LSTM

LSTM was the technique chosen for the sequence-to-sequence neural network. Several other structures were attempted such as temporal convolutional neural networks (TCN), 1-Dimensional Convolutional Neural Network (1-D CNN) but the best results were achieved using LSTM.

5.2 TWO-LAYER LSTM

To get greater depth in the memory of the LSTM, two layers of LSTM were stacked in the network architecture (see Figure 25). This structure provided the highest accuracy and classifying the fault type. The two-layer LSTM was also used for breaker identification using pole non-simultaneity.

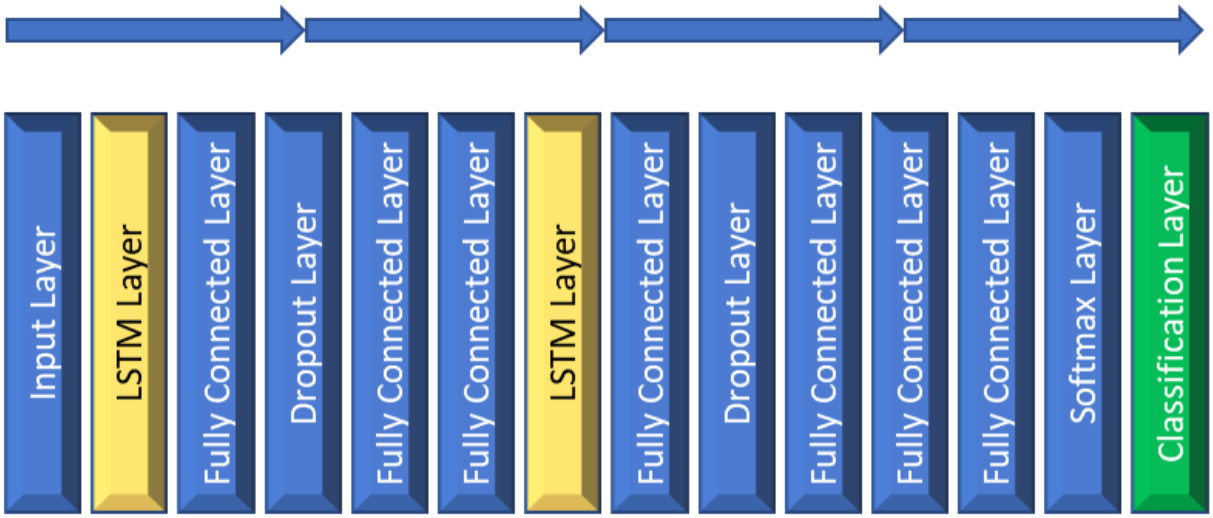


Figure 25 Two-Layer LSTM

6. DELAY ERROR

In protection and other time sensitive problems, it is imperative to rapidly recognize a fault condition regardless of the technique used. If machine learning is to be used for protection purposes, then the nature of fault events requires correct identification of the event but correct identification is just as important as how long the correct identification takes. machine learning often uses a Confusion Matrix (CM) to graphically display the accuracy of a machine learning algorithm as shown in Figure 26.

The CM is a common tool used to identify accuracy of classification systems. Counts of accurate and inaccurate classification of each label are identified in a matrix so as to highlight the accuracy of each label. However, the CM does not provide any information concerning the amount of time on average the correct classification took. If a Machine Learning Algorithm (MLA) takes longer to identify a fault than the time it takes for the equipment that it is protecting to be damaged, then a correct fault identification by the MLA is of little use.

Therefore, a new metric to quantify the accuracy and speed of each algorithm had to be developed called Delay-Error (DE). DE is a technique that is helpful in presenting how effective an MLA is in identifying protection related events and how quickly the algorithm correctly categorizes the events (see Figure 27).

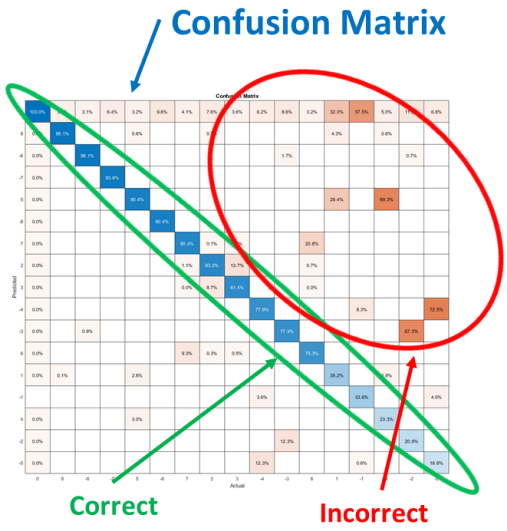
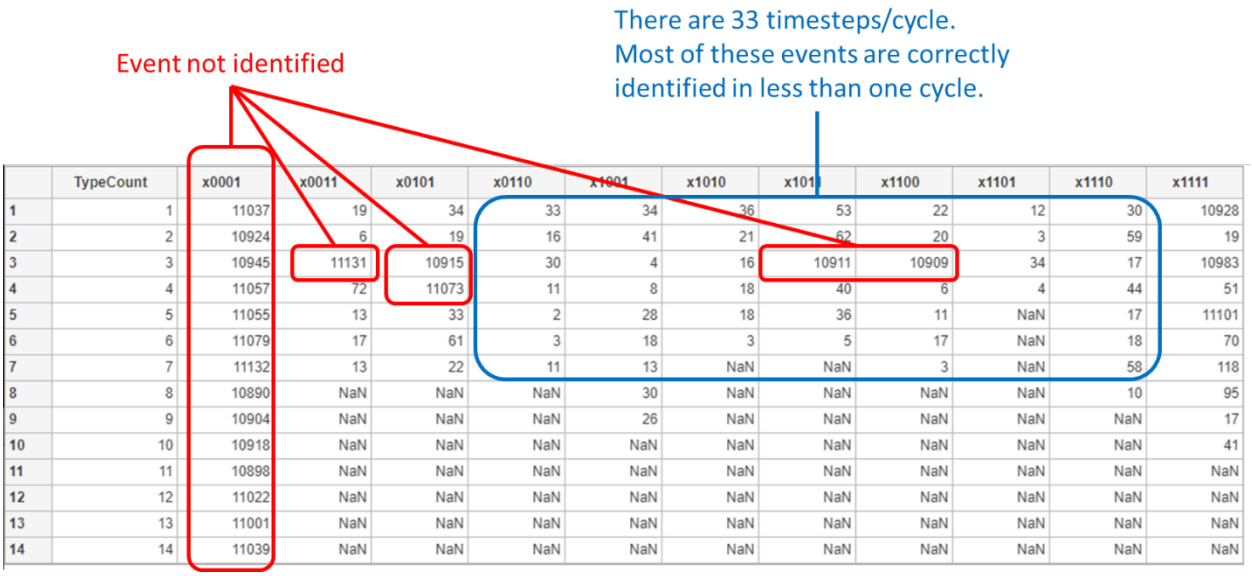


Figure 26 Confusion Matrix of Machine Learning Algorithm

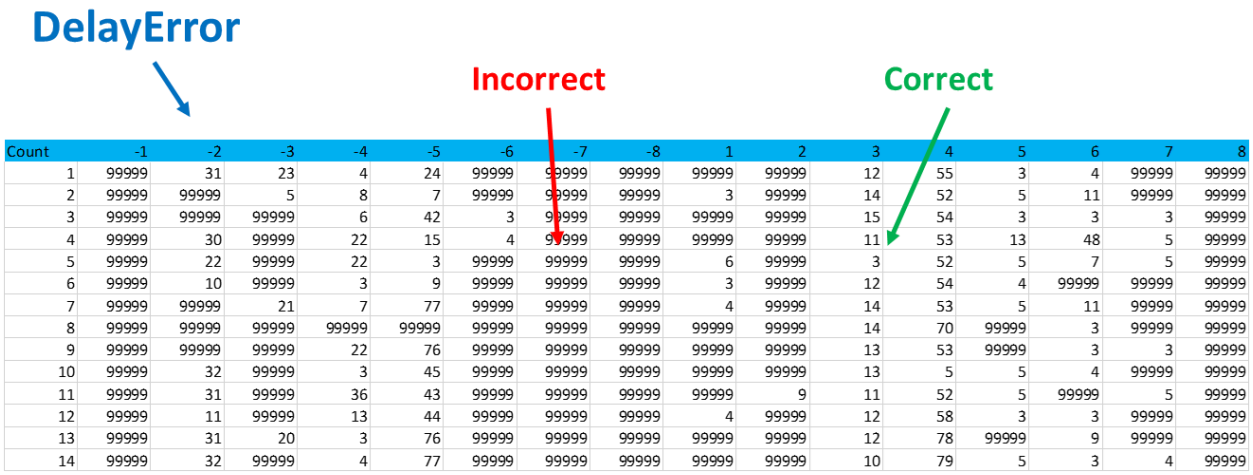


	TypeCount	x0001	x0011	x0101	x0110	x1001	x1010	x1011	x1100	x1101	x1110	x1111
1	1	11037	19	34	33	34	36	53	22	12	30	10928
2	2	10924	6	19	16	41	21	62	20	3	59	19
3	3	10945	11131	10915	30	4	16	10911	10909	34	17	10983
4	4	11057	72	11073	11	8	18	40	6	4	44	51
5	5	11055	13	33	2	28	18	36	11	NaN	17	11101
6	6	11079	17	61	3	18	3	5	17	NaN	18	70
7	7	11132	13	22	11	13	NaN	NaN	3	NaN	58	118
8	8	10890	NaN	NaN	NaN	30	NaN	NaN	NaN	NaN	10	95
9	9	10904	NaN	NaN	NaN	26	NaN	NaN	NaN	NaN	NaN	17
10	10	10918	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	41
11	11	10898	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
12	12	11022	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	13	11001	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14	14	11039	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure 27 Delay-Error Table (each column represents a different classification label)

The DE calculates the number of time steps between the start of the event of interest and when the MLA recognizes that the event has started. It does this for each event file and every category of label within the test data set. To accomplish this, a S2S neural network is used to produce labeled output for each timestep of the output. The labels of each timestep of the input test data is compared to the labels of the output of the neural network. The number of timesteps between the start of the event in the labels of input test data to the start of the event in the labels of output test data are counted. The count of the difference is the DE for the test file.

The results are put into a table that shows columns for each label type or event type (see Figure 28). Each row of each column represents a unique test file. The cell value on each row displays the DE, the number of time steps required for the MLA to identify event. If the MLA fails to identify the event entirely the total number of timestamps in the event file is displayed. Figure 28 illustrates that several events (in the red boxes) were not correctly classified.



Count	-1	-2	-3	-4	-5	-6	-7	-8	1	2	3	4	5	6	7	8
1	99999	31	23	4	24	99999	99999	99999	99999	99999	12	55	3	4	99999	99999
2	99999	99999	5	8	7	99999	99999	99999	3	99999	14	52	5	11	99999	99999
3	99999	99999	99999	6	42	3	99999	99999	99999	99999	15	54	3	3	3	99999
4	99999	30	99999	22	15	4	99999	99999	99999	99999	11	53	13	48	5	99999
5	99999	22	99999	22	3	99999	99999	99999	6	99999	3	52	5	7	5	99999
6	99999	10	99999	3	9	99999	99999	99999	3	99999	12	54	4	99999	99999	99999
7	99999	99999	21	7	77	99999	99999	99999	4	99999	14	53	5	11	99999	99999
8	99999	99999	99999	99999	99999	99999	99999	99999	99999	99999	14	70	99999	3	99999	99999
9	99999	99999	99999	22	76	99999	99999	99999	99999	99999	13	53	99999	3	3	99999
10	99999	32	99999	3	45	99999	99999	99999	99999	99999	13	5	5	4	99999	99999
11	99999	31	99999	36	43	99999	99999	99999	99999	9	11	52	5	99999	5	99999
12	99999	11	99999	13	44	99999	99999	99999	4	99999	12	58	3	3	99999	99999
13	99999	31	20	3	76	99999	99999	99999	99999	99999	12	78	99999	9	99999	99999
14	99999	32	99999	4	77	99999	99999	99999	99999	99999	10	79	5	3	4	99999

Figure 28 Correct vs. Incorrect Event Identification in Delay-Error Table

The number of rows in the table represents the maximum number of files for any particular event type (columns). If there are fewer event files for a particular event type than the maximum number of files for any particular event type, then the cells for the higher count rows are occupied by a "NaN" to indicate that there are no more files for that particular event type (column).

The total time that for correct classification for each file can be calculated by dividing the number of timesteps in a particular cell by the sample rate in the data. Figure 28 illustrates that many of the events (in the blue box) were correctly identified in less than one cycle (33 timesteps/cycle).

6.1 SUMMARY STATISTICS

Another script ingests the data from DE to calculate summary statistics on the DE results. Statistics include things like the average accuracy, average time for correct classification (see Figure 29), average overall time (accurate and error). Statistics are presented by event type (column) as in Figure 30. amount of time for each event type and the average time overall which provides measure of accuracy in time.

avg				A	B	C	D
	samp/sec	samp/cyc	128	256	512	1024	2048
1	20000	333.3	0.06	0.06	0.08	0.08	0.13
2	10000	166.7	0.05	0.05	0.07	0.07	0.12
3	6667	111.1	0.04	0.05	0.03	0.08	0.06
4	5000	83.3	0.04	0.04	0.03	0.06	0.07
5	4000	66.7	0.04	0.05	0.05	0.07	0.07

Figure 29 Delay-Error Summary Statistics (Data Sample Rate vs NN Structure)

The statistics are useful for troubleshooting and analysis. Reporting statistics in different ways helps to highlight different aspects of the test data and results in order to help identify strengths and weaknesses of different neural network structures and effectiveness on different categories. The summary statistics sometimes help identify errors in the neural network structures and sometimes errors in the test or train data.

	Accuracy																
	-1	-2	-3	-4	-5	-6	-7	-8	1	2	3	4	5	6	7	8	Overall
DS1_128	0%	73%	29%	78%	75%	20%	0%	0%	29%	2%	88%	98%	59%	61%	39%	7%	41%
DS2_128	8%	25%	81%	63%	47%	5%	0%	0%	20%	25%	86%	98%	46%	14%	66%	14%	38%
DS3_128	51%	61%	61%	36%	90%	19%	19%	12%	53%	68%	85%	98%	36%	41%	64%	29%	51%
DS4_128	47%	53%	83%	25%	86%	47%	37%	37%	59%	53%	88%	49%	69%	86%	51%	29%	56%
DS5_128	53%	64%	80%	80%	68%	47%	42%	27%	49%	41%	88%	98%	54%	98%	56%	29%	61%
DS1_256A	22%	81%	5%	44%	63%	83%	0%	0%	36%	3%	86%	53%	49%	22%	71%	17%	40%
DS2_256A	20%	56%	46%	5%	90%	0%	0%	0%	10%	39%	69%	98%	59%	20%	46%	12%	36%
DS3_256A	61%	42%	86%	80%	51%	19%	19%	14%	59%	51%	88%	98%	34%	59%	69%	31%	54%
DS4_256A	54%	76%	56%	51%	80%	42%	37%	39%	42%	51%	78%	98%	37%	97%	39%	29%	57%
DS5_256A	71%	78%	58%	37%	85%	53%	42%	25%	56%	44%	88%	98%	54%	95%	54%	37%	61%
DS1_512B	76%	90%	83%	0%	92%	0%	3%	0%	47%	36%	49%	85%	42%	97%	71%	54%	52%
DS2_512B	80%	90%	0%	0%	92%	2%	0%	0%	24%	15%	73%	78%	24%	75%	71%	2%	39%
DS3_512B	27%	41%	88%	31%	88%	19%	19%	12%	66%	58%	80%	59%	49%	75%	64%	29%	50%
DS4_512B	47%	90%	31%	44%	78%	47%	39%	41%	49%	58%	88%	49%	69%	81%	63%	34%	57%
DS5_512B	42%	76%	47%	19%	71%	54%	41%	25%	71%	41%	88%	97%	53%	98%	68%	31%	58%
DS1_1024C	76%	90%	83%	0%	92%	0%	3%	0%	47%	36%	49%	85%	42%	97%	71%	54%	52%
DS2_1024C	90%	90%	29%	0%	83%	24%	0%	0%	78%	66%	53%	95%	17%	56%	71%	14%	48%
DS3_1024C	78%	86%	61%	34%	90%	32%	19%	8%	66%	56%	51%	98%	31%	98%	64%	25%	56%
DS4_1024C	83%	90%	29%	22%	92%	64%	36%	36%	73%	61%	56%	95%	37%	85%	69%	27%	60%
DS5_1024C	93%	90%	46%	36%	92%	66%	42%	29%	75%	64%	68%	95%	54%	85%	51%	32%	64%
DS1_2048D	80%	86%	88%	10%	64%	81%	78%	75%	76%	59%	14%	76%	24%	76%	58%	66%	63%
DS2_2048D	66%	86%	88%	49%	61%	8%	42%	2%	75%	47%	10%	49%	27%	42%	47%	7%	44%
DS3_2048D	93%	90%	49%	39%	78%	22%	17%	7%	68%	64%	37%	42%	47%	98%	69%	25%	53%
DS4_2048D	88%	90%	69%	44%	64%	59%	8%	37%	80%	71%	85%	97%	41%	97%	37%	22%	62%
DS5_2048D	97%	90%	63%	53%	71%	47%	32%	32%	75%	69%	64%	76%	69%	98%	44%	53%	65%
	Med	High	Med	Low	High	Low	Low	Low	Med	Med	High	High	Low	High	Med	Low	
								-8								8	

Figure 30 Delay-Error Summary Statistics (Event Type vs. NN Structure)

6.2 DELAY ERROR VERSUS CONFUSION MATRIX

Both CM and DE are useful for understanding the accuracy of an MLA in identifying events. CM is very useful in providing a matrix that is easy to see which events are classifying accurately in which events are not. The CM is very visual and its display of that accuracy along with counts of how many events are misclassified.

The DE is useful in understanding the amount of time required to correctly identify each event. The time is very important in the context of protection systems and other time sensitive operations. Protection systems rely on correctly identifying a fault or other event in a short period of time in order to clear the fault from the system and protect system equipment and/or personnel. The faster an event can be correctly identified the less time there will be for the fault to continue to burn. This is particularly important in the context of high-voltage transmission systems where the total clearing time is often on the order of single digit cycles.

7. SPECTROGRAM BASED CNN FOR POWER SYSTEM WAVEFORMS

Audio Signals tend to be in the kilohertz frequency range whereas power systems typically have 60 Hz signals. Audio signals are often (not always) a single data stream. Power systems signals are typically six data streams (three voltage and three current). Otherwise, power systems signals (Figure 31 Single-Phase Power System Current Waveform) are very similar to audio signals (Figure 32 Audio Waveform of the Word YES).

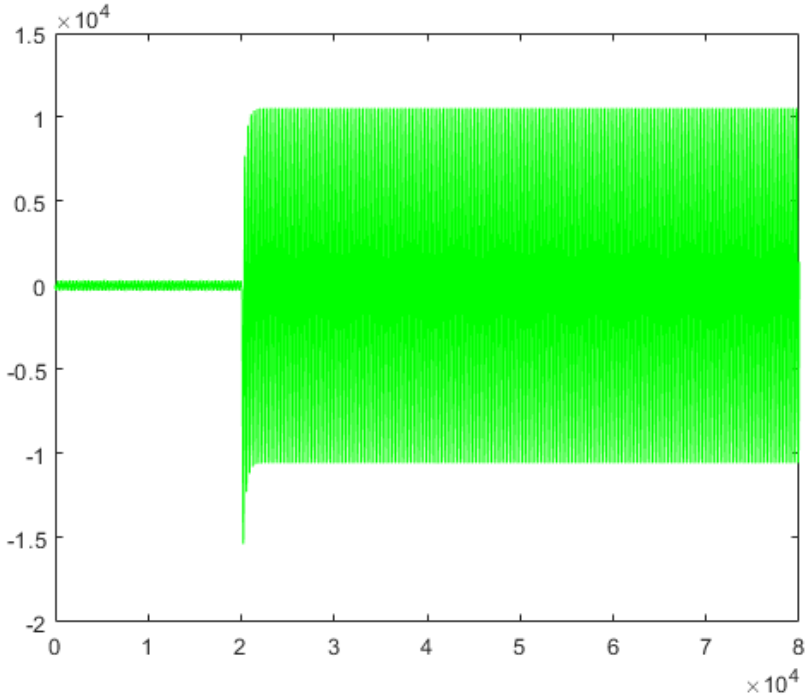


Figure 31 Single-Phase Power System Current Waveform (horizontal axis is time, vertical axis is current amplitude)

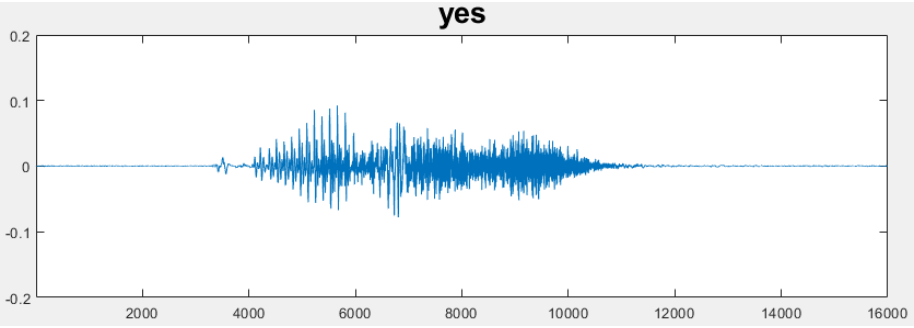


Figure 32 Audio Waveform of the Word YES (x-axis is time, y-axis is audio signal amplitude)

To address the need for a commercially viable solution that can classify waveform data, energies were directed to develop a universal NN structure (deep learning algorithm) that works for a wide variety of system event types. The structure that showed most promise was one that included the use of spectrograms. The technique has shown positive results in audio engineering, particularly with respect to speech recognition. A MATLAB example problem illustrates some code in theory of how to use spectrograms in speech command recognition [19]. A YouTube course titled “Audio Signal Processing for Machine Learning”, was referenced to gain greater insight into how spectrograms in conjunction with CNN are used to process audio signals [14]. Further depth of understanding was gained from the book titled “Fundamentals of Music Processing”, by Meinard Müller [15].

7.1 BASIC SPECTROGRAM

A basic spectrogram function is available within MATLAB.

spectrogram(C80k,window,noverlap, freqloc)

When the spectrogram function[8] is performed on the fault waveform in Figure 31 a spectrogram image is produced as illustrated in Figure 33. Time is on the x-axis and harmonic frequency is on the y-axis. Notice at the time of fault initiation (2 on the x-axis) a vertical line appears. This vertical line illustrates how a step function includes a wide variety of harmonics including high harmonics. It is also important to note the very bottom edge along the x-axis is bright yellow. This is illustrating the low frequency components of the waveform including DC and 60 Hz. With respect to 60 Hz, note how the bottom edge becomes brighter yellow past the two-index on the x-axis. This is illustrating how the magnitude of the fundamental frequency increases because of the fault current. These characteristics will be leveraged in the final version of the power system spectrograms described below.

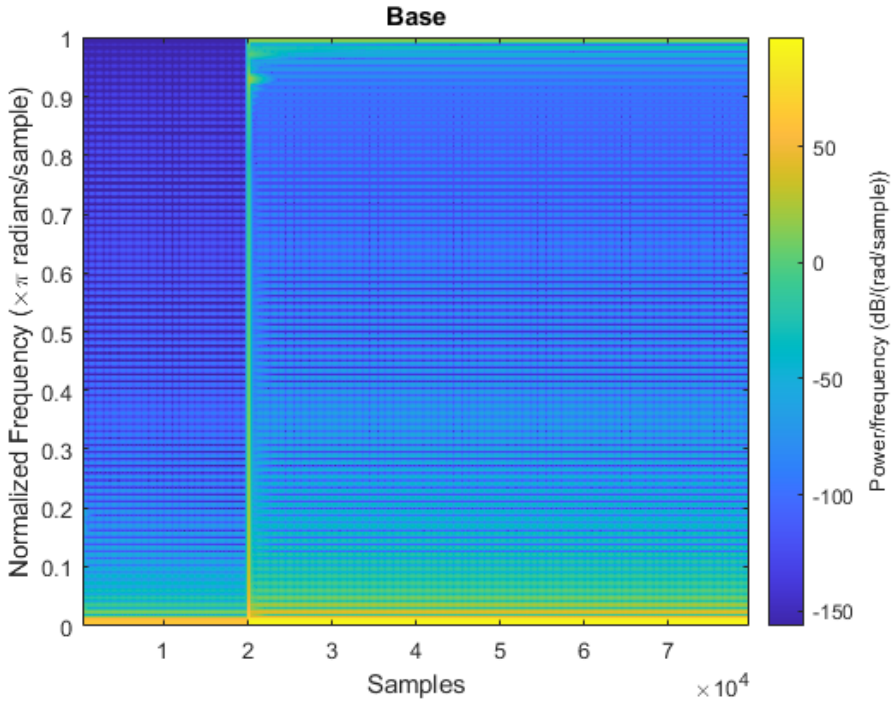


Figure 33 Spectrogram of Fault Current Waveform

The MATLAB spectrogram function takes x, window, noverlap and nfft as arguments, where:

- X: is the input signal
  - C80k

[8] Spectrogram: https://www.mathworks.com/help/signal/ref/spectrogram.html

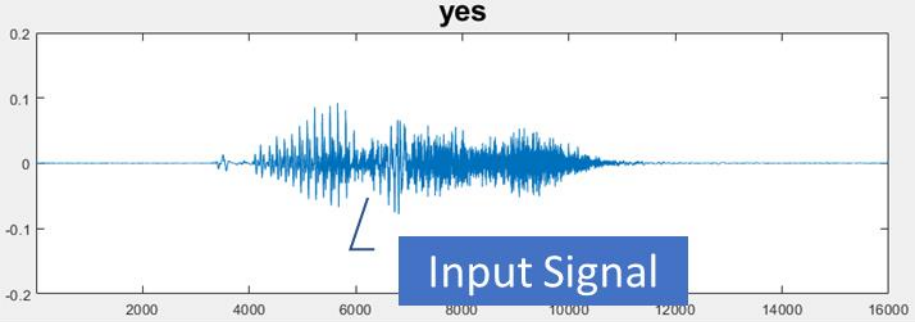


Figure 34 Input Audio Signal (x-axis: time, y-axis: signal amplitude)

- Window: is used to divide the signal into segments and perform windowing
    - window

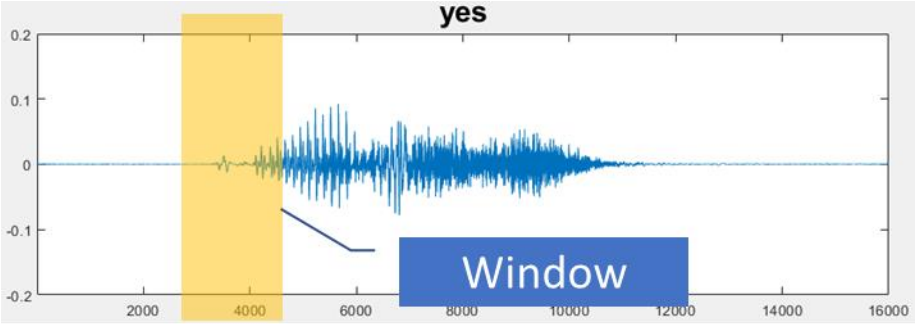


Figure 35 Window for the Short-Term Fourier Transform (STFT) (x-axis: time, y-axis: signal amplitude)

- Noverlap: is the number of samples of overlap between adjoining segments
    - noverlap

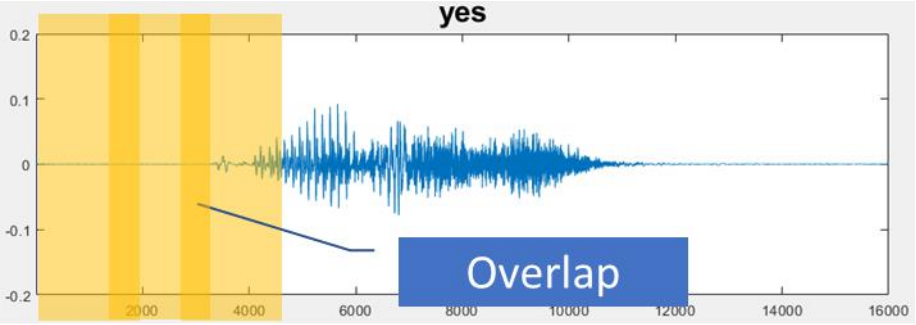


Figure 36 Overlapping Windows for the STFT (x-axis: time, y-axis: signal amplitude)

- Nfft: is the number of sampling points to calculate the discrete Fourier transform
    - freqloc
    - This is a combination of the window width and the sample rate of the data. It is the number of samples within the window.

A Short-Time Fourier Transform (STFT[9]) is a Fast Fourier Transform (FFT[10]) of a window (see Figure 35) of the signal data. After performing the FFT the window is shifted in time. The new window and the

[9] MATLAB function stft: https://www.mathworks.com/help/signal/ref/stft.html
[10] MATLAB function fft: https://www.mathworks.com/help/matlab/ref/fft.html

previous window may overlap as illustrated in Figure 36. Another FFT is performed on the next window. This is repeated across the entire length of the signal. The STFT (see Figure 37) technique is the basis for the Spectrogram.

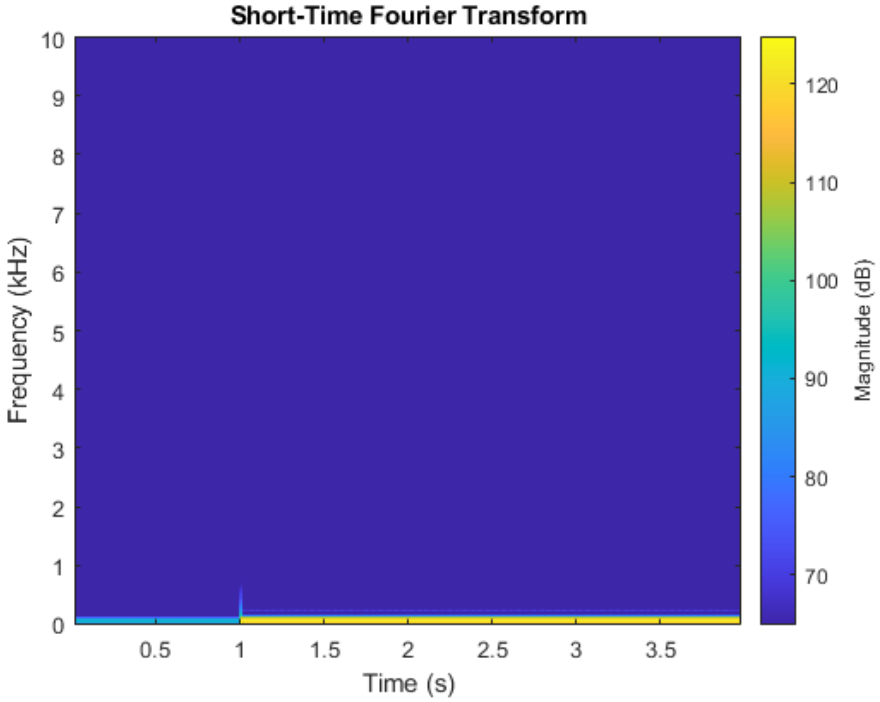


Figure 37 Short-Time Fourier Transform (STFT)

7.2 WINDOW

The window length for the PSNN is two cycles. A two-cycle window is long enough to capture the beginning of faults. Further, it is long enough to capture the beginning of a motor start which can last anywhere from a fraction of a second to thirty seconds. By keeping the window width narrow (2-cycles) it minimizes the number of samples that go into the Fast Fourier Transform (FFT) and thereby the number of computations required to calculate the FFT. A two-cycle window width should be appropriate for a wide variety of power system events of interest. In future work it may be worthwhile to investigate different window widths for different types of power system events.

With respect to the window, recall how the FFT of the spectrogram generates a wide spectrum of frequencies at the edge of a step function. This is noticeable in the fault waveform in Figure 31 creating a spike in the spectrogram image illustrated in Figure 33. The same thing happens at the edge of a STFT Window.

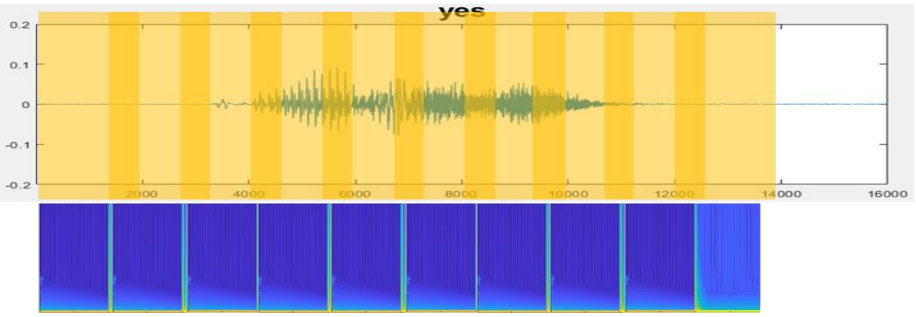


Figure 38 Uncompensated Windowing (x-axis: time, y-axis: signal amplitude)

If not compensated for, the spectrogram generated using STFT would show a spike at the edge of every window used to break up the signal data as illustrated in Figure 38. The spikes would be misleading when trying to interpret the resulting spectrogram. Therefore, the edges of each window must be mathematically compensated for. To compensate for the window edge effect a windowing function us used as illustrated in Figure 39. The signal data is multiplied by the window function which is zero outside the window and approaches zero toward the edges of the window. This eliminates the sharp step function at the edge of the window.

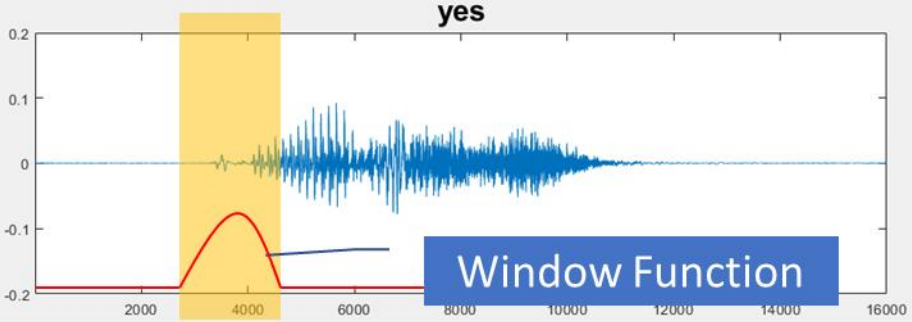


Figure 39 Window Function (x-axis: time, y-axis: signal amplitude)

If the windows join edge to edge, then information could be lost at the edge of each window. Therefore, the windows are overlapped as illustrated in Figure 40. This preserves the information at the edges of the windows.

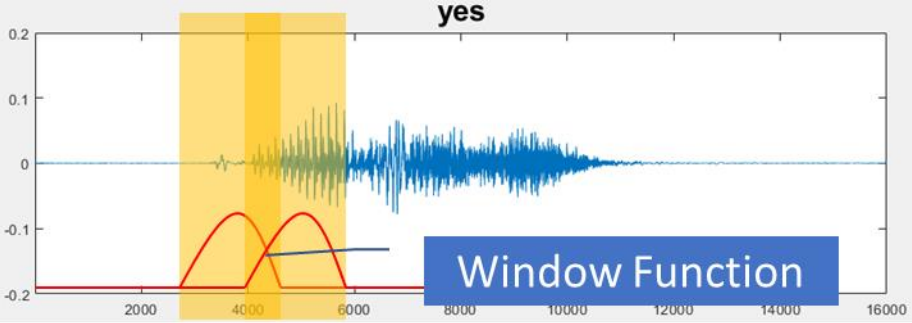


Figure 40 Window Function Overlap (x-axis: time, y-axis: signal amplitude)

There are many different types and shapes of window functions available for use and many of them are preprogrammed into MATLAB for use on signal data including:

- Hann (Hanning) window
- Hamming window
- Kaiser window
- Triangular window
- Tukey (tapered cosine) window
- Parzen (de la Vallée Poussin) window
- Nuttall-defined minimum 4-term Blackman-Harris window
- Gaussian window
- Flat top weighted window
- Chebyshev window
- Bohman window
- Minimum four-term Blackman-Harris window
- Blackman window

- Bartlett window
- Modified Bartlett-Hann window

The default windowing function in the MATLAB spectrogram function is the Hamming window. Each windowing function was checked and qualitatively compared to find the one most suited for the power system waveform data. The Parzen windowing function (see Figure 41) was found to give optimal results which is consistent with the recommendation made in [20].

Similar in concept but separate from window width is datafile length. Datafile length is the amount of time in a particular event file. In the case of this study, the datafile length is four seconds. While data files may include more than four seconds of data, the most important portion of the fault events are sufficiently represented in four seconds. In the case of large motors starting the entire event may be more than 30 seconds, but a significant portion of the important and descriptive information will mostly be in the first four seconds. Therefore, four seconds was chosen for the PSNN described here.

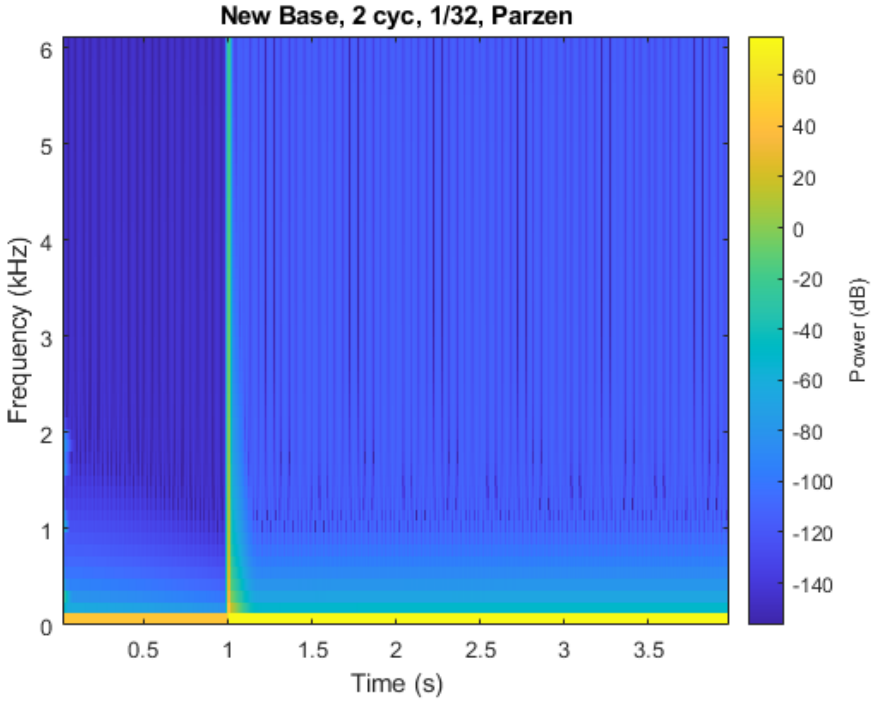


Figure 41 Spectrogram with Parzen Windowing Function

A function was created to calculate window width in cycles based on sample frequency and power system frequency (typically 60 Hz in the USA and 50 Hz in Europe). The name of the function is titled "Fcn_PSysSampCycle.mlx" in the functions folder. The function is called in the main body of the code and returns a structure variable named "sampCycle" (see page C-14).

7.3 HOP & OVERLAP

The overlap between adjacent windows is the amount by which a window overlaps the previous window when creating the spectrogram. A closely related concept is that of Hop. Hop is the amount by which the window is shifted in time.

$$\text{Hop} = \text{Window} - \text{Overlap} \qquad (1)$$

Hop and overlap are closely related, one or the other may be used in functions to create spectrograms. In the case of the built-in MATLAB spectrogram function, overlap is used to identify shift between subsequent windows.

It is important to ensure that a hop is chosen that minimizes calculation but provides sufficient resolution in the time step is appropriate for a wide variety of system conditions of interest. Several different hop sizes were qualitatively compared in order to ensure image quality in the resulting spectrograms (Figure 42, Figure 43, Figure 44, Figure 45).

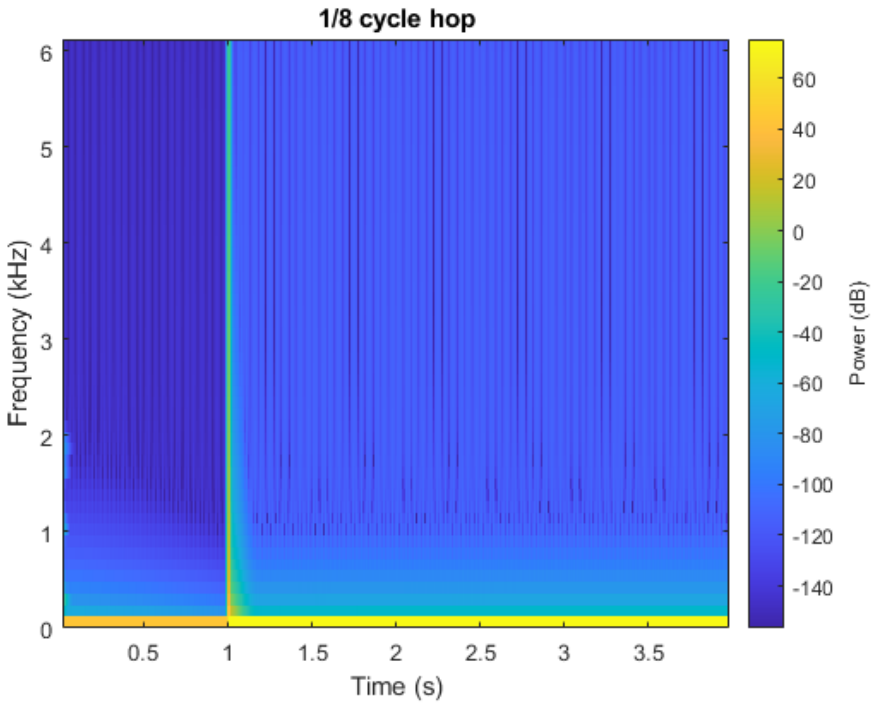


Figure 42 1/8th Cycle Hop

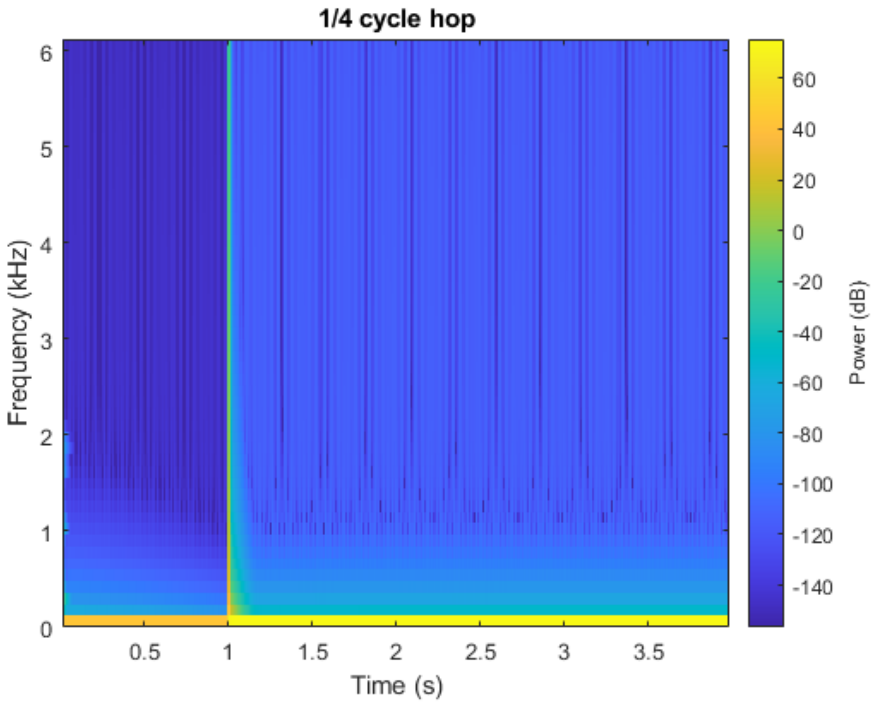


Figure 43 1/4th Cycle Hop

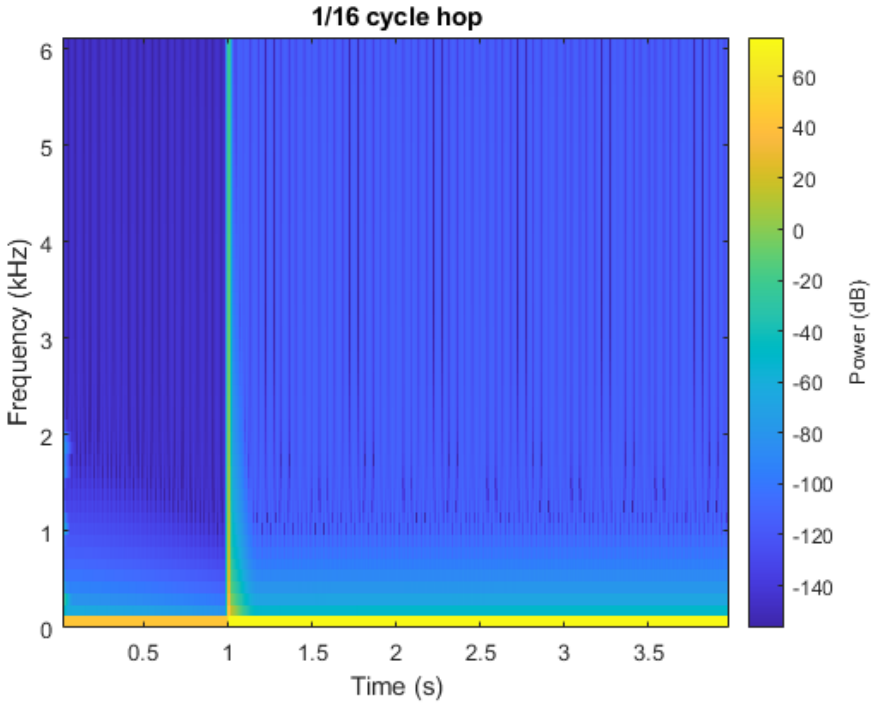


Figure 44 1/16th Cycle Hop

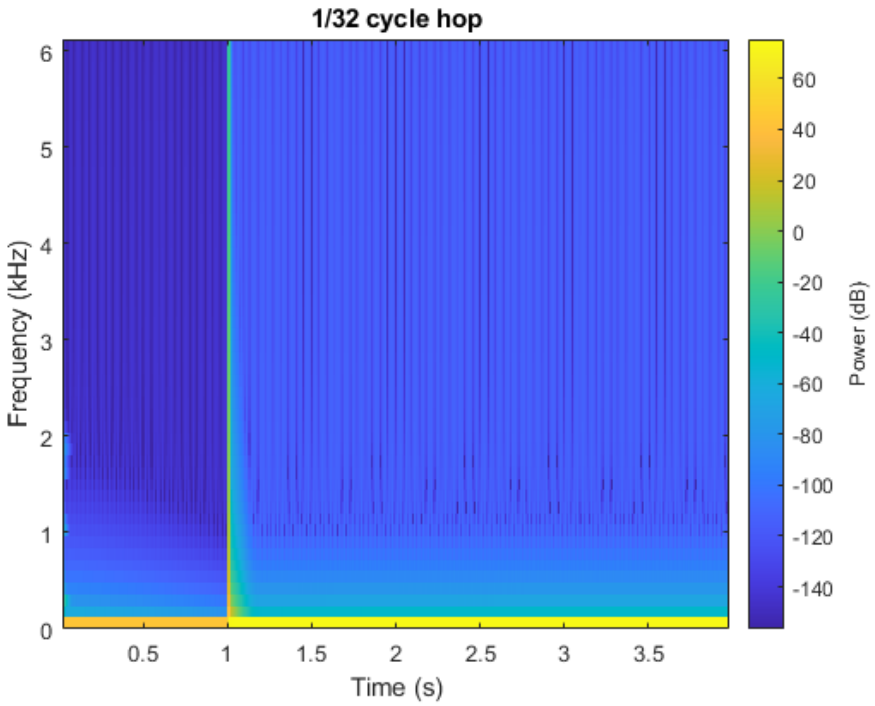


Figure 45 1/32nd Cycle Hop

In the case of PSNN 1/32nd of a cycle was chosen as the hop. A hop of 1/32nd of a cycle provides sufficient resolution in the data to allow for the use of PSNN for protection purposes. In protection it is important to identify a fault event as quickly as possible. Speed is particularly important in the context of transmission protection where total clearing time may be 3 cycles which includes relay time, communication time and mechanical operation of the breaker. When working with the LSTM neural networks it was found that the NN's could, in many cases, identify a fault event in approximately 3/32nd of a cycle. Therefore, a hop of 1/32nd of a cycle was chosen for PSNN in order to achieve similar performance. A shorter hop would result in more calculation time with limited additional benefit.

7.4 DFT POINTS (NFFT)

A Discrete Fourier Transform (DFT[11]) of a digital signal in the time domain samples a signal multiple times and maps the signal to a range of discrete frequencies in the frequency domain. In power systems analysis this is related to the concept of harmonics where continuous-time power system signal is transformed (mapped) to integer multiples of the fundamental frequency[12] in the frequency domain such as: 60 Hz = 1st harmonic, 120 Hz = 2nd harmonic, 180 Hz = 3rd harmonic, 240 Hz = 4th harmonic, and so on.

Typically, power system signals are represented in the harmonic components. Signals processed with DFT are transformed to a range of frequencies which are not necessarily integer multiples of a power system fundamental frequency (60 Hz). In the case of DFT, the user selects an integer number of frequencies to map to, which is called NFFT in the case of the spectrogram function in MATLAB is

[11] The efficient version of the DFT is the FFT. The MATLAB function is fft: https://www.mathworks.com/help/matlab/ref/fft.html

[12] 60 Hz in the USA and 50 Hz in other parts of the world such as Europe

called NFFT[13] [14] [15] for Number of DFT points (frequencies). Several values for NFFT were tested and qualitatively compared in Figure 46, Figure 47, and Figure 48. There was no observable difference between the different selections at this stage. The higher the number of DFT points the greater the required computation. Therefore, lower values for DFT points are desirable when possible. However, if the number of DFT points is less than the window width then the data is padded with zeros which will tend to skew the results.

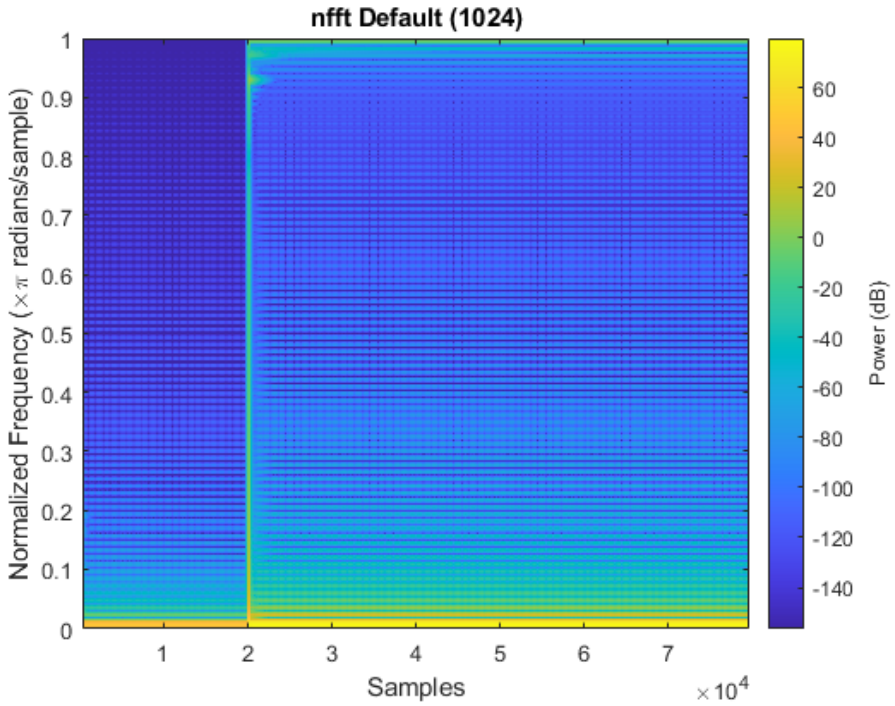


Figure 46 Default NFFT = 1024

Balance needs to be found between:

- window width,
  - wide window increases potential to characterize important parts of the event signal
  - wide window increases calculation time and requirements
- DFT points,
  - more DFT points increases resolution and precision
  - more DFT points increases calculation time and requirements
- required precision of the final calculations
  - slow events may not require high precision (i.e. wind & solar variability)
  - fast events will require high precision (transients and protection events)

The default number of DFT points in the PSNN code is 1024. The value is held in the FFT parameters function titled "Fcn_PSysFFTparams" in the variable name is FFTparams.nfft. When the main code runs the Fcn_PSysFFTparams function is called from which point FFTparams.nfft is made available to the rest

[13] In the MATLAB function spectrogram: **nfft** — Number of DFT points
https://www.mathworks.com/help/signal/ref/spectrogram.html
[14] In the MATLAB function stft: **FFTLength** — Number of DFT points
https://www.mathworks.com/help/signal/ref/stft.html
[15] In the MATLAB function fft: **n** — Transform length https://www.mathworks.com/help/matlab/ref/fft.html

of the code. The value can be permanently changed in the Fcn_PSysFFTparams function or can be temporarily updated later in the code by updating the value for the FFTparams.nfft variable.

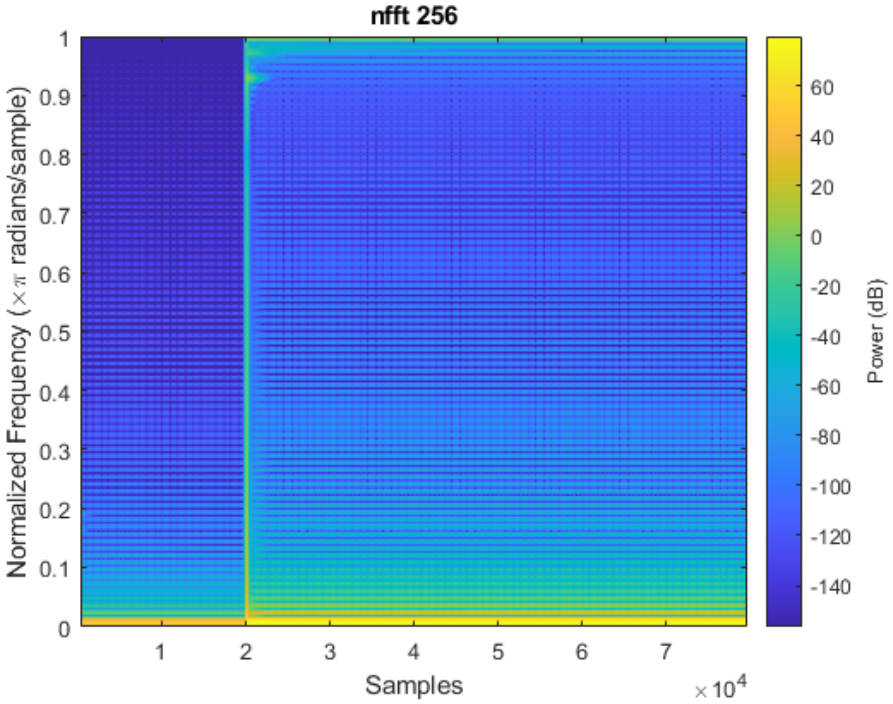


Figure 47 NFFT = 256

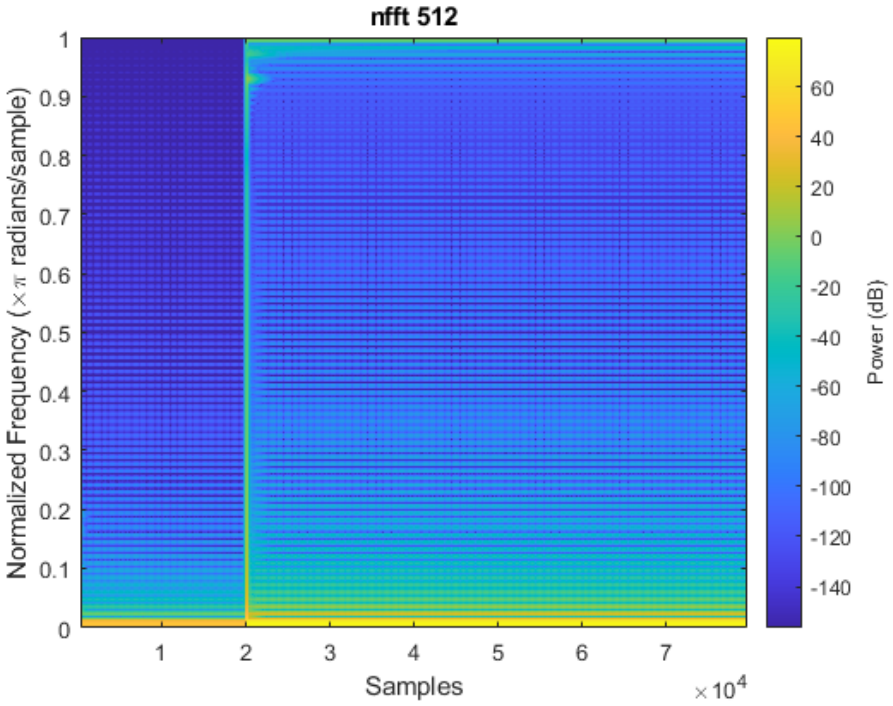


Figure 48 NFFT = 512

7.5 POWER SYSTEM FILTER-BANK (CHANNELIZER)

Understanding DFT points is important in understanding the Filter-Bank (Characterizer). As explained in section 7.4, the FFT maps to a wide spectrum of frequencies as defined by DFT points. However, in power systems analysis it has been long established that harmonics are important to the understanding of what is taking place on the power system.

Odd harmonics are most relevant in understanding power system events[16]. Another, pseudo-harmonic that is of great importance is that of the Direct Current (DC) component. The DC component represents zero-frequency or direct current and is a common occurrence at the beginning of a fault, when motors are starting and other key times. The DC component is usually short lived (on the order of single-digit cycles), but is very informative when present. At other times Triplen harmonics are most appropriate for understanding some system conditions.

Given that the FFT produces results in a wide frequency spectrum and power system analysis is based on harmonics of the fundamental frequency there is a need for a way to convert FFT results into harmonic components. A filter-bank is used to do this conversion. The filter-bank is a matrix of rows equal to DFT points and columns equal to the harmonic frequencies of interest. The filter-bank matrix is used as a filter to the spectrogram or STFT matrix and reduce the number of rows in the matrix/image while focusing on the most important information for power systems analysis.

7.5.1 The Filter-Bank

The filter-bank is a function designed to extract the harmonic frequencies from the spectrogram image thereby reducing the number of rows in the spectrogram and focusing on the most relevant power system information. Graphically the filter–bank is illustrated in Figure 49. The y-axis is the magnitude at which the rows of the spectrogram is multiplied when passed into the filter-bank algorithm. The x-axis represents the range of frequencies. Each triangular line represents a separate filter. Each triangular filter is centered on a harmonic frequency of interest. The triangle drops from a magnitude of one at the center frequency to a value of zero at the center frequency of the next filter triangle.

Each row of the STFT image represents a particular frequency, the row is multiplied by the corresponding frequency column of the filter-bank. Rows of the spectrogram that do not correspond with the center frequency of a filter get multiplied by value less than one and in many cases by value of zero. Frequency rows of the STFT image that correspond to a center frequency in the filter-bank get multiplied by a value of one. The results get added together to create a new row for spectrogram that corresponds to each filter in the filter-bank. This results in a spectrogram with fewer rows but highlights the most relevant information.

To better illustrate the point in the function of the filter bank see Figure 50. The raw spectrogram image is depicted on the left side of Figure 50. An illustrative depiction of the filter-bank is shown near the middle of Figure 50. The final result, a filtered spectrogram, is depicted on the right side of Figure 50. Take note of how higher order frequencies may be eliminated from the data set. Also take note how lower order frequencies are concentrated into a reduced number of rows in the spectrogram image. This approach is advantageous when processing the spectrograms using a CNN. In audio engineering a Mel Filter-Bank is used but is inappropriate for power systems as it is configured for audio signals as opposed to power system signals as this Power System Filter-Bank is.

[16] note that odd harmonics are the most informative in power system analysis and the even harmonics are typically ignored

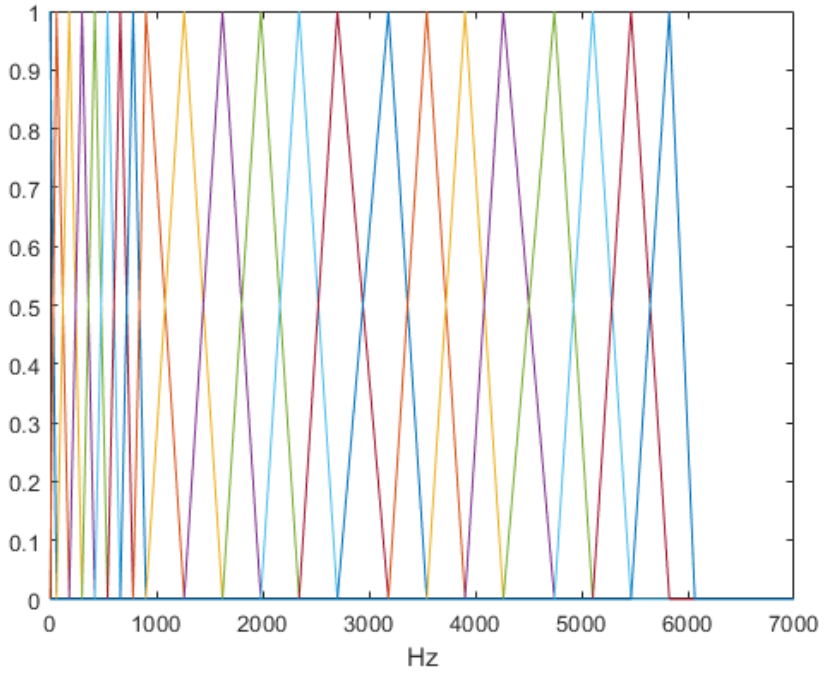


Figure 49 Power System Filter-Bank

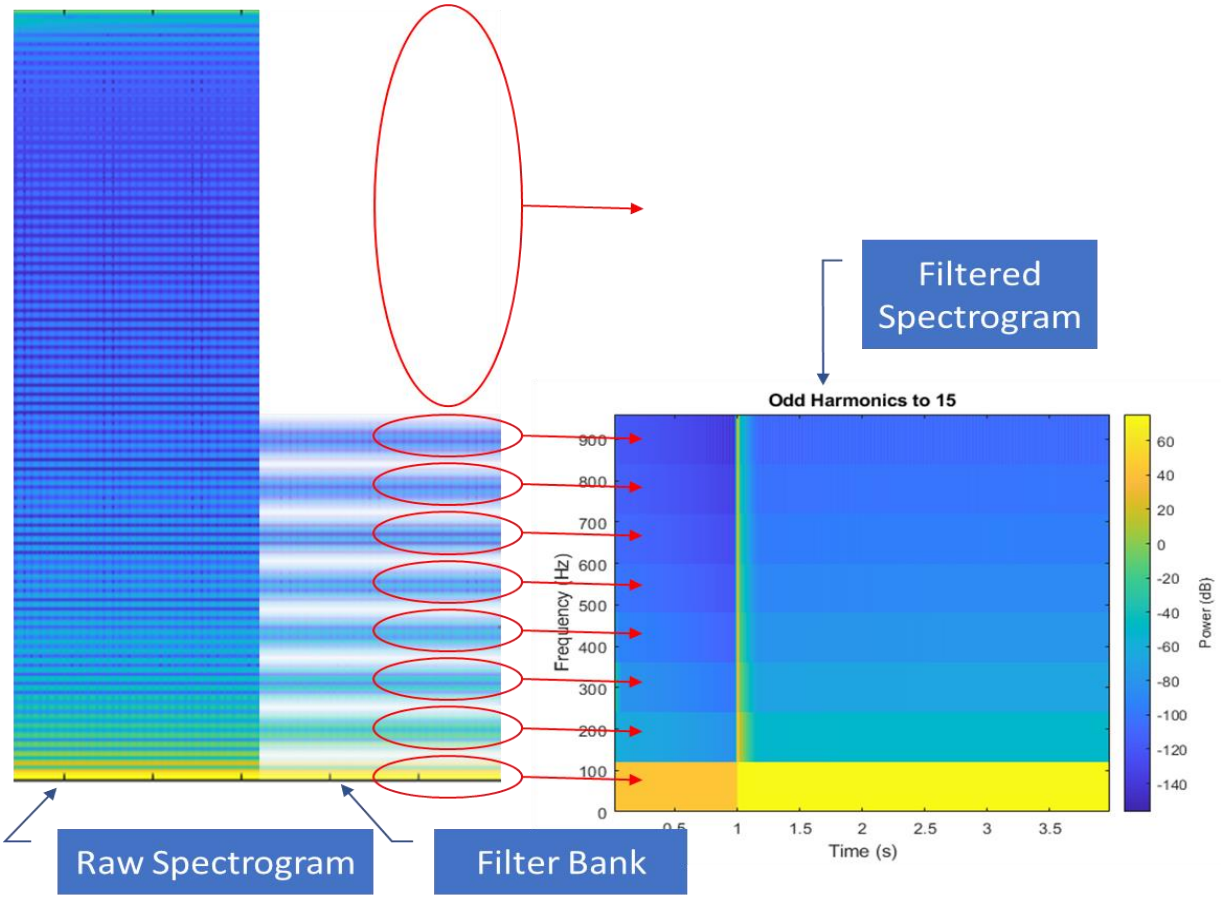


Figure 50 The Behavior of the Filter-Bank

7.5.2 Harmonics and Sequence Components

In order to gain a better understanding of harmonics it is worth wile to tie them back to sequence components. Foundational understanding is that of the relationship between harmonic order and sequence. The three-phase positive sequence components contain three sinusoids 120º from one another. Positive-sequence (A-B-C) phase rotation is [0, 120, 240] = [0, 120, -120]. Negative-sequence (A-C-B) phase rotation is [0, 240, 120] = [0, −120, 120]. Zero-sequence are in phase with each other (e.g., 0, 0, 0).

Harmonic phase sequence can be determined by multiplying the harmonic number h with the positive sequence phase rotation [0, 120, 240]. For example:

- first-harmonic (the fundamental),
    - h = 1, produces 1 x [0, 120, -120] = [0, 120, -120] or [0, -120, 120]
    - which is positive sequence
- second-harmonic,
    - h = 2, produces 2 x [0, 120, -120] = [0, 240, -240] or [0, −120, 120]
    - which is the negative sequence
- third-harmonic,
    - h = 3, produces 3 x [0, 120, -120] = [0, —360, 360] or [0, 0, 0°]
    - which is the zero sequence

Phase sequence for all other harmonic orders can be determined in the same fashion.

To that end the relationship of sequence to lower level harmonic order are :

- Harmonic Order = 1, 7, 13, … are POSITIVE SEQUENCE
- Harmonic Order = 5, 11, 17, … are NEGATIVE SEQUENCE
- Harmonic Order = 3, 9, 15, 21 … are ZERO SEQUENCE
    - These are the Triplen Harmonics

7.5.3 Harmonics Chosen for the Power System Filter-Bank

Part of the objective of the power system filter–bank is to reduce the number of rows in the spectrogram image while still emphasizing the most relevant information concerning power system events. As a starting point the filter bank eliminates even numbered harmonics as they are not very helpful in analyzing power system data. All odd harmonics are included up to the 15th harmonic. Above the 15th harmonic, triplen harmonics are included. Additionally, low-frequencies below 30 Hz are gathered together as a DC component. Spectrograms passed through the filter-bank with all harmonics, odd harmonics to the 9th, odd harmonics to the 15th, odd harmonics to the 51th, odd harmonics to the 101th are qualitatively compared in Figure 51, Figure 52, Figure 53, Figure 54, Figure 55, and Figure 56.

The filter–bank function is designed to take as an input the top harmonic of interest which may be the 15th (see), the 51st (see) or the 101st harmonic. For many applications the 15th harmonic should be sufficient as that is what is typically the cut off for many meters and analysis tools today.

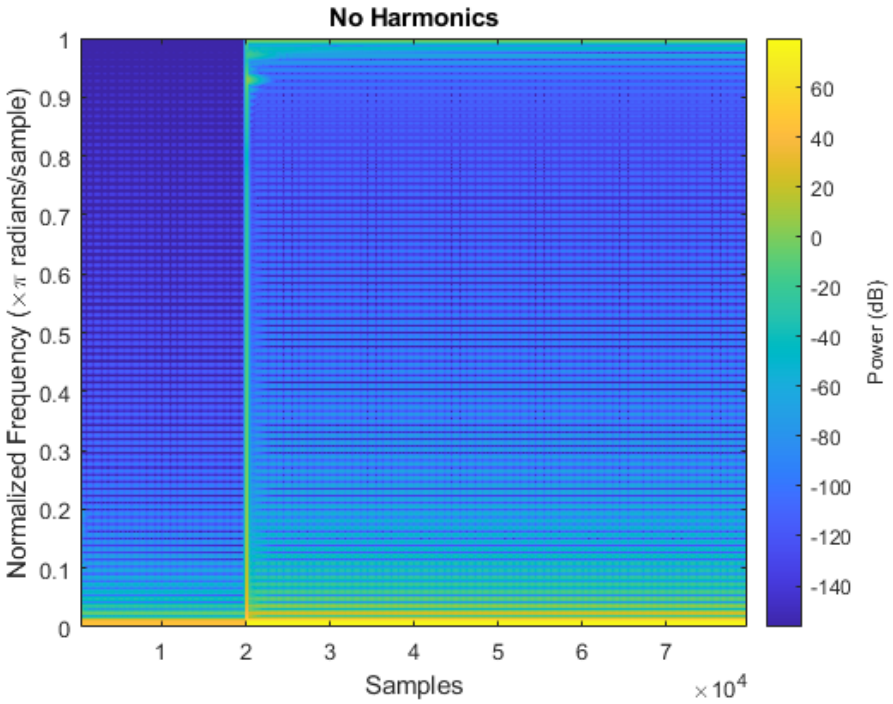


Figure 51 Raw (Unfiltered) Spectrogram

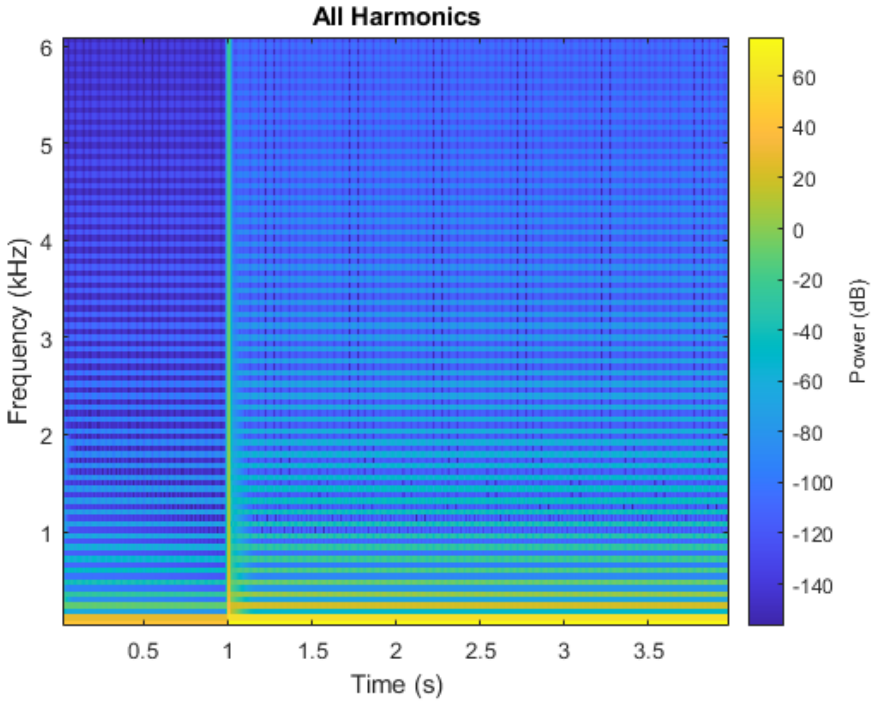


Figure 52 All Harmonics

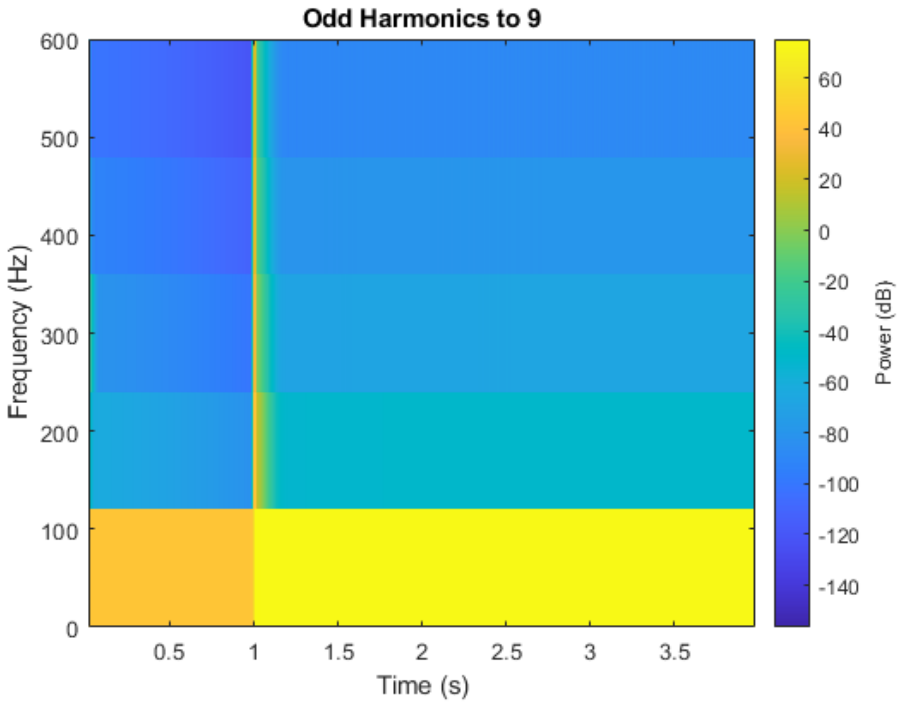


Figure 53 Odd Harmonics to the 9th

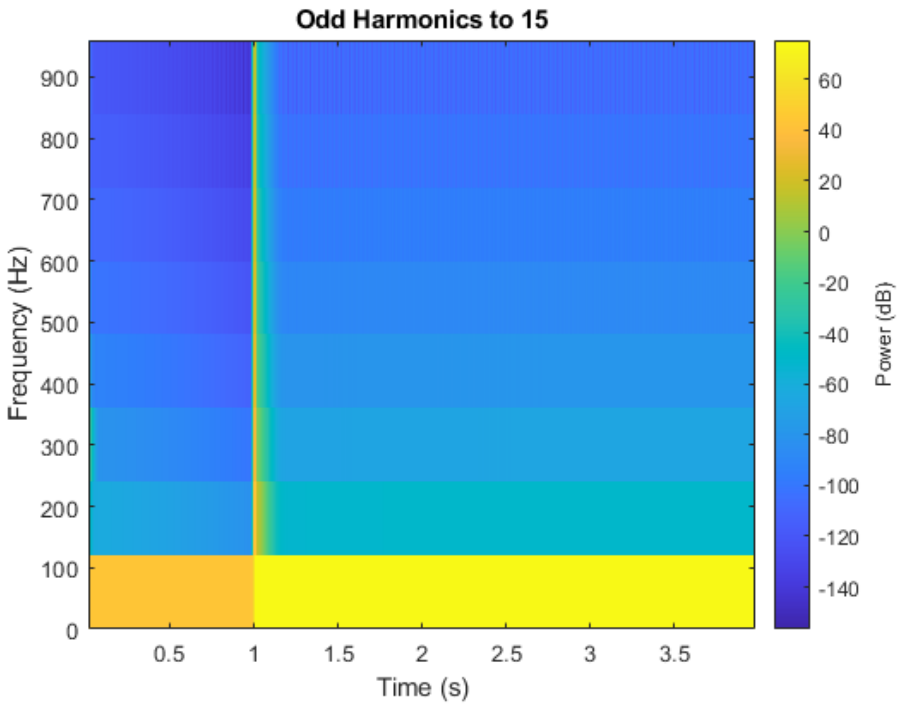


Figure 54 Odd Harmonics to the 15th

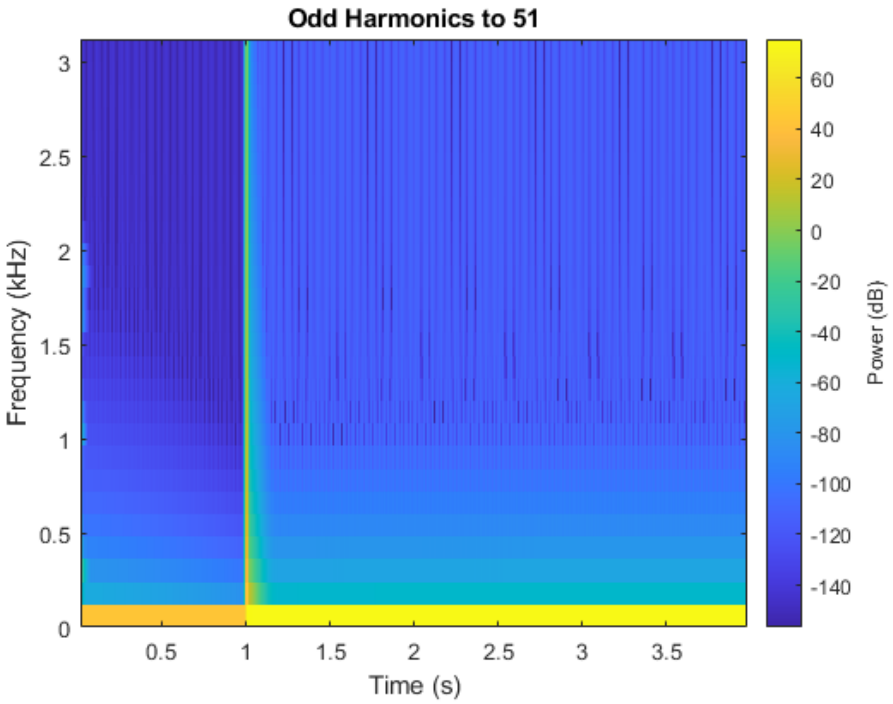


Figure 55 Odd Harmonics to the 51st

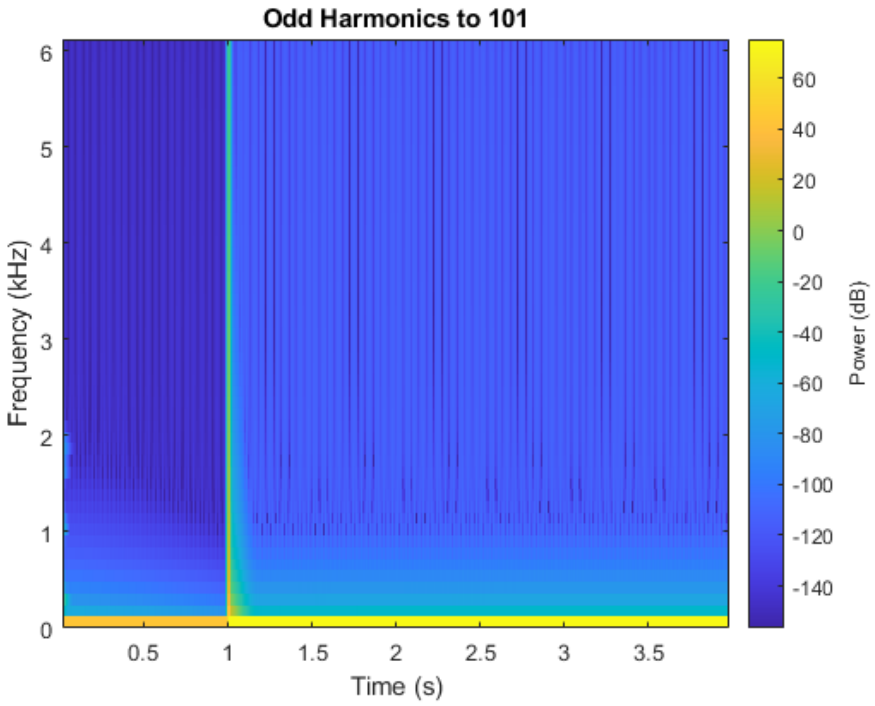


Figure 56 Odd Harmonics to the 101st

7.6 SIGNAL DATA NORMALIZATION

When performing power systems analysis on waveforms there are two types of signals used, namely voltage and current signals. Both voltage and current have three phases. The amplitude of the voltage signals has an order of magnitude of tens of thousands of volts (10^4 volts) or hundreds of thousands of volts (10^5 volts) depending on whether the voltage is for distribution or transmission. The amplitude of the current signal may be on the order of tens of amps (10s amps) for load currents to single-digit thousands of amps (1^3 amps) for fault currents.

There is enough difference between voltage magnitude, load current magnitude and fault current magnitude that necessitates some compensation so that the load current signal is of a comparable magnitude to the voltage signal. When plotted on the same plot it can be difficult to distinguish voltage from current. Figure 57 illustrates how the current signal appears to be zero when plotted against the voltage signal. While there are nonzero values for the current signal, they are much smaller than the voltage signal values.

The use of spectrograms and the filter-bank is building up to use Convolutional Neural Networks (CNN) to process the waveform data. Just as in the visual illustration below the CNN will have difficulty making use of the voltage and current signals together when there is significant difference in the order of magnitude between them. Therefore, the waveform data needs to be scaled, or normalized, so as to make the voltage and current waveforms have comparable scales.

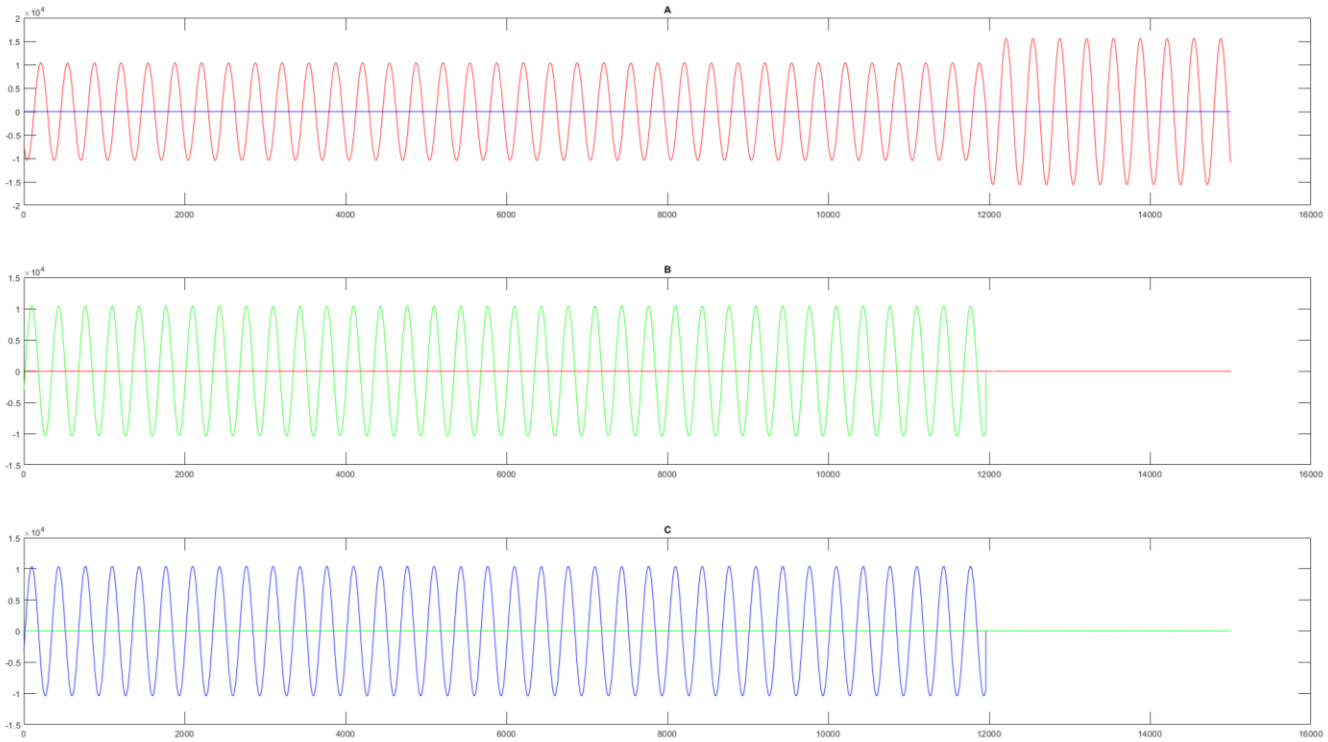


Figure 57 Uncompensated (Non-normalized) Voltage and Current Signals (x-axis: time, y-axis: signal amplitude)

To normalize the voltage and current waveforms, it is important to note that there are two different scales of magnitude in the current waveforms, that of load current and that of fault current. In both power flow analysis software and protection analysis a technique is used to normalize voltage and current data, often referred to as “Per-Unit” or simply “p.u.”. While the technique employed in the PSNN is inspired by per-unit, it does not strictly adhere to the per-unit system of normalization. Doing so will sometimes exaggerate the scale problem illustrated in Figure 57.

The normalization used for PSNN leaves the current unchanged but scales the voltage by:

$$Voltage\_Out = Voltage\_In \,/\, (BaseVoltage * 1e3) \,/\, sqrt(2/3) \quad (2)$$

Where the BaseVoltage is the line voltage in kV RMS and Voltage_In is the voltage signal data. Figure 58 illustrates how the voltage and current waveforms are both visible in the same plots. While both voltage and current are notably different in magnitude and all plots of Figure 58, they are similar enough to be distinguishable whether it is the load current portion of the signal or the fault current portion of signal.

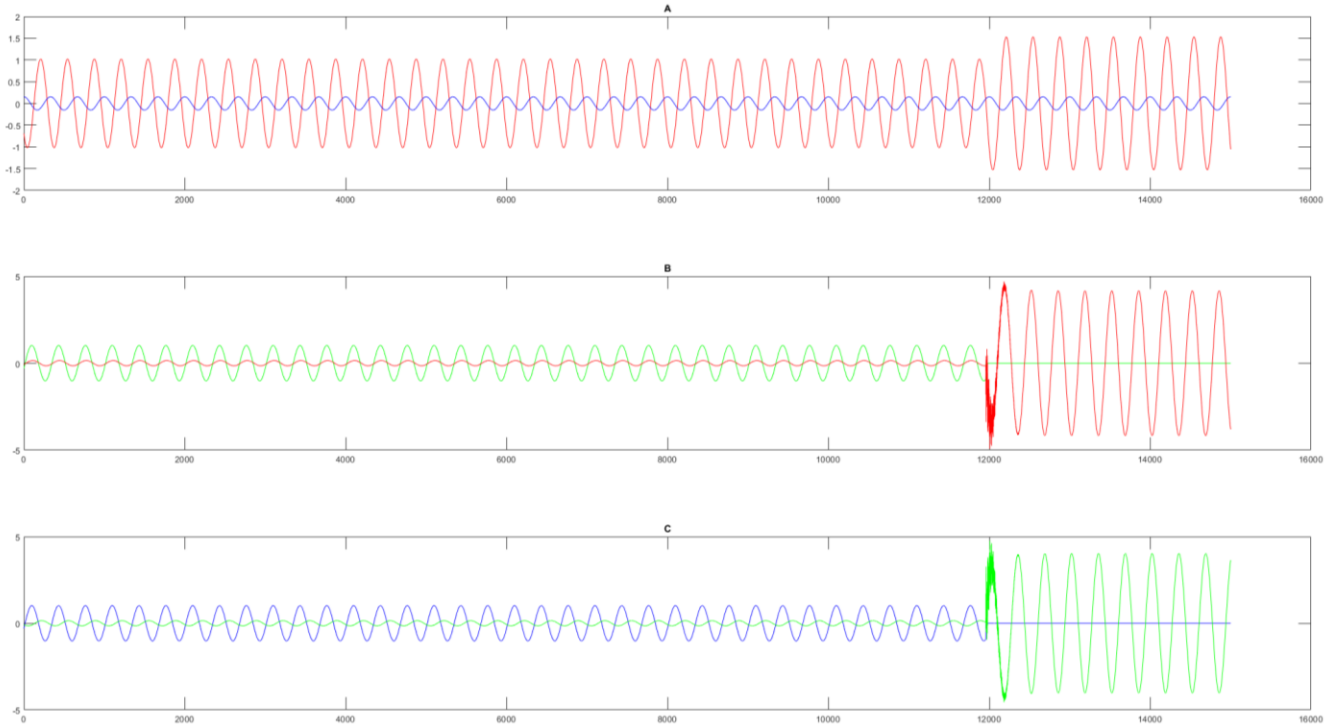


Figure 58 Compensated (Normalized) Voltage and Current Signals (x-axis: time, y-axis: signal amplitude)

7.7 MERGING THE VOLTAGE AND CURRENT SPECTROGRAMS

The process to this point includes:

1) Normalize the voltage waveform data using the pseudo-per-unit calculations
2) Compute Power Spectrograms of the normalized waveform data using the Fcn_PSysSpectIV function which leverages PSysFilterBank function (see Figure 59)
   a) For many applications set:
      i) NFFT = 1024 (unless the window width is smaller)
      ii) Top harmonic = 15 (unless higher order harmonics are of interest)

In the next step the coincident phase voltage spectrograms were merged to the current spectrograms. This pulls the voltage and current into a single image for each phase. This is beneficial as the voltage generally responds at the same time as the current in the case of faults and other system events. By combining the voltage and current into a common spectrogram image the CNN can get the benefit of seeing a full picture of what is going on and it reduces the inputs into the CNN which saves on calculation.

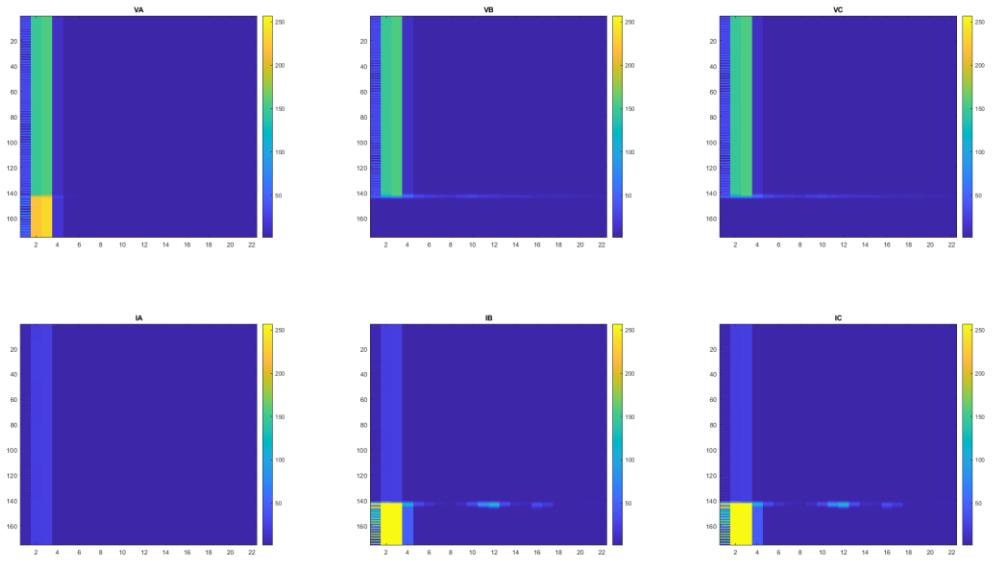


Figure 59 Spectrograms of Voltage (top row) & Current (bottom row)

The next step is to flip the current signal spectrograms from left to right (see Figure 60).

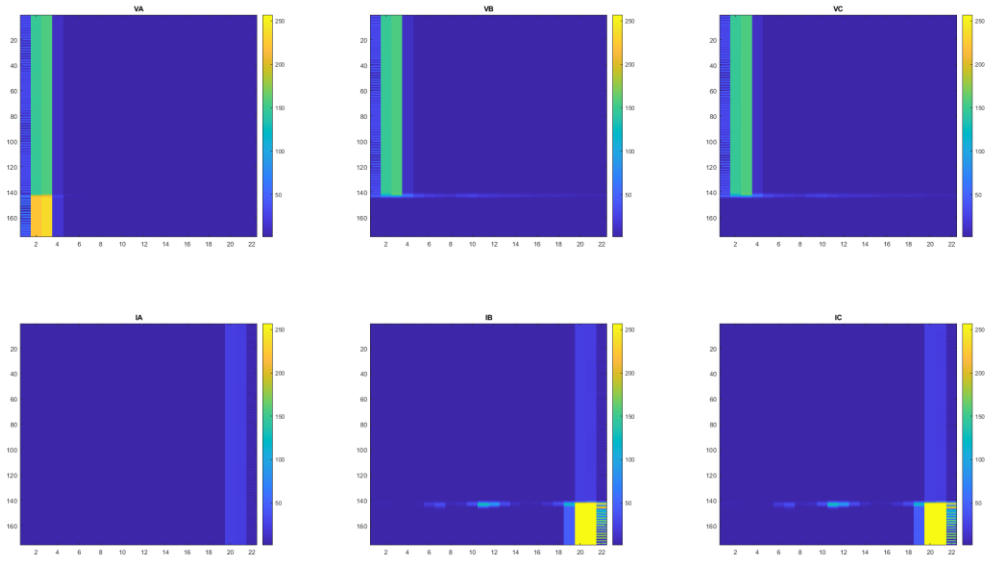


Figure 60 Spectrograms of Voltage (top row) & Current (bottom row) where Current is Flipped from Left to Right

Finally, the Voltage spectrograms were combined with the corresponding phase of the Flipped Current spectrograms into single spectrogram images where voltage and current are separated by a column of zeros. The current was put on the left and the voltage was put on the right as illustrated in Figure 61. The output was saved to a 3-dimensional array with spectrograms for each phase along the third dimension.

In the convolution process, the new pixel of the convolved image is the dot product of the CNN kernel with the part of the original image that is of the same size as the kernel. Depending on how the kernel and/or the CNN is set up the edges of the image include the dot product within some cases padding values

(zeros) that surround the image. Therefore, the edges of the image include convolution with zero padding which washes out information contained at the edge of the image. Combining the voltage and current into single images in this way keeps the most important information in the center of the image thereby reducing or eliminating the washout effect. Moving the important information to the center of the image ultimately improves the results of the CNN.

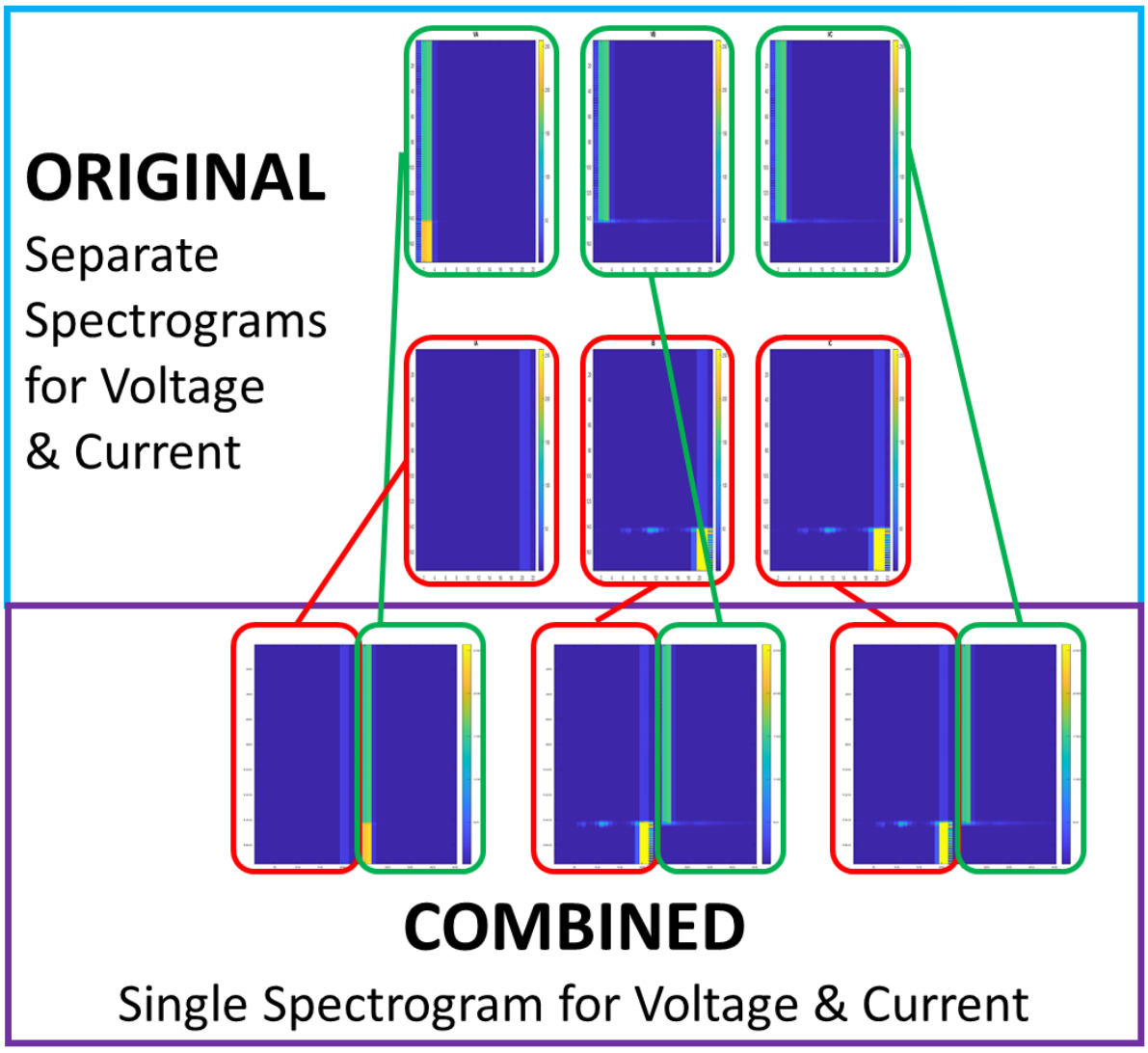


Figure 61 Combined Spectrograms of Voltage and Flipped Current

7.8 LOGARITHM

The spectrograms were finally processed by applying a logarithm. Taking a log of small numbers can lead to roundoff error. Therefore, the features were scaled using a small offset (epsil) then taking the log. This provides the data with a smoother distribution and accentuates the most important information to improve the performance of the CNN (see Figure 62).

$$epsil = 1e-6 \qquad (3)$$

$$X = log10(X + epsil) \qquad (4)$$

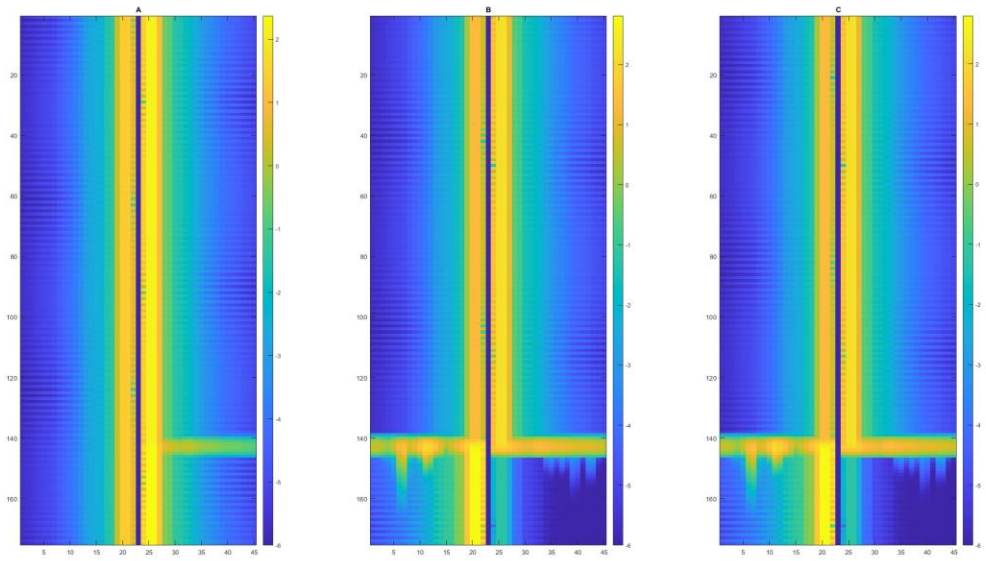


Figure 62 Results After Application of the Logarithm

A comparison of the impact of the Logarithm can be seen in Figure 63.

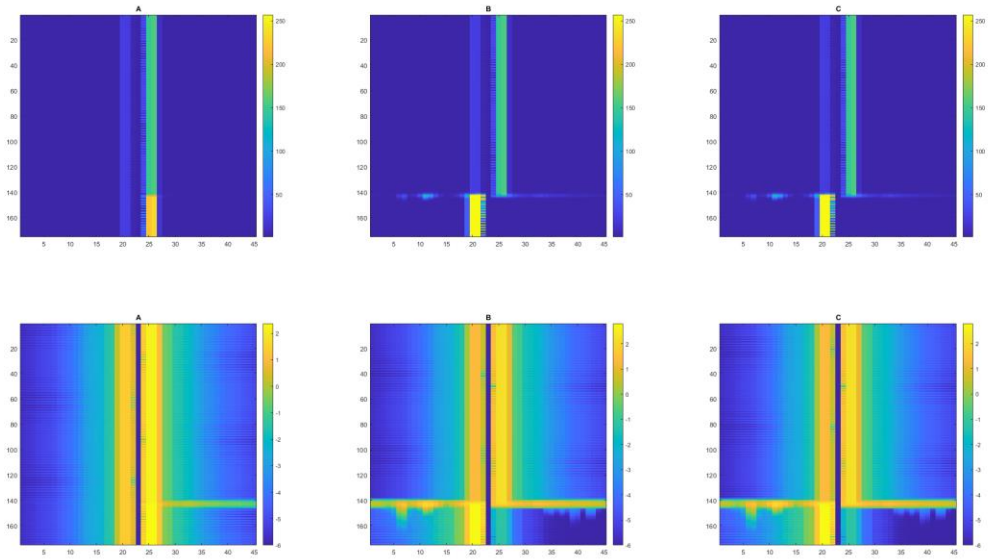


Figure 63 Comparison of Before and After the Application of the Logarithm

7.9 CONFIGURE AND TRAIN THE CONVOLUTIONAL NEURAL NETWORK (CNN)

The process to this point includes:

1) Normalize the voltage waveform data using the pseudo-Per-Unit calculations
2) Compute Power Spectrograms of the normalized waveform data using the Fcn_PSysSpectIV function which leverages PSysFilterBank function (see Figure 59)
   a) For many applications set:

i) NFFT = 1024 (unless the window width is smaller)[17]
ii) Top harmonic = 15 (unless higher order harmonics are of interest)

3) Flip the current spectrograms (see Figure 60)
4) Combine the voltage spectrograms with the flipped current spectrograms (see Figure 61)
5) Apply the Logarithm function to the combined spectrograms (see (3)

At this point the spectrogram images were passed through the CNN for training. The structure of the neural network is illustrated in Figure 64 (code on page C-8).

Figure 64 Neural Network Structure

The training options for the neural network include the Adam optimizer, with 25 max Epochs, a Mini-batch size of 128, and an Initial Learn Rate of 3e-4. The training options are outlined in MATLAB code in section C-13.

Once training is started, the error drops to near zero and the accuracy goes to near 100% in just a few epochs. Additional training beyond the first few epochs pushes the validation accuracy to 100% by the time training is complete (see Figure 65). Testing is performed with fresh data that the network has not seen. The results of the test show 2 errors out of approximately 1400 test samples equal to 99.86% (see Figure 66).

[17] The sample rate of the dataset is 20,000 samples/sec

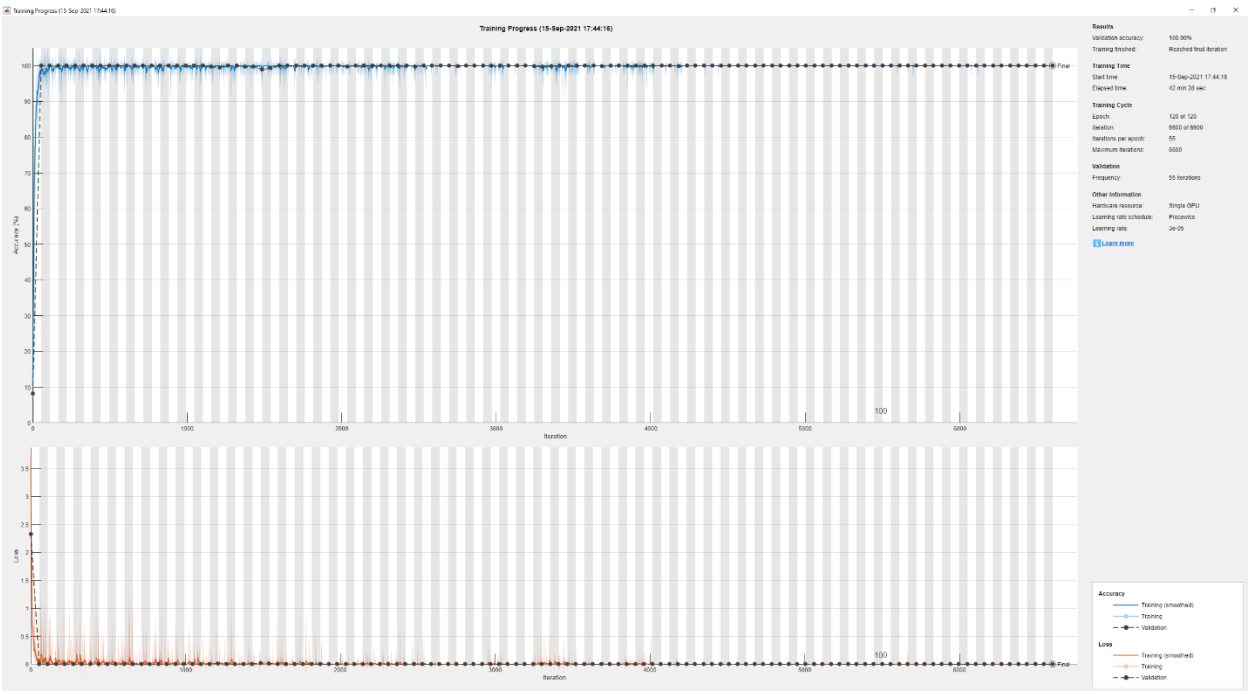


Figure 65 Training Plot with Accuracy (top) and Error (bottom)

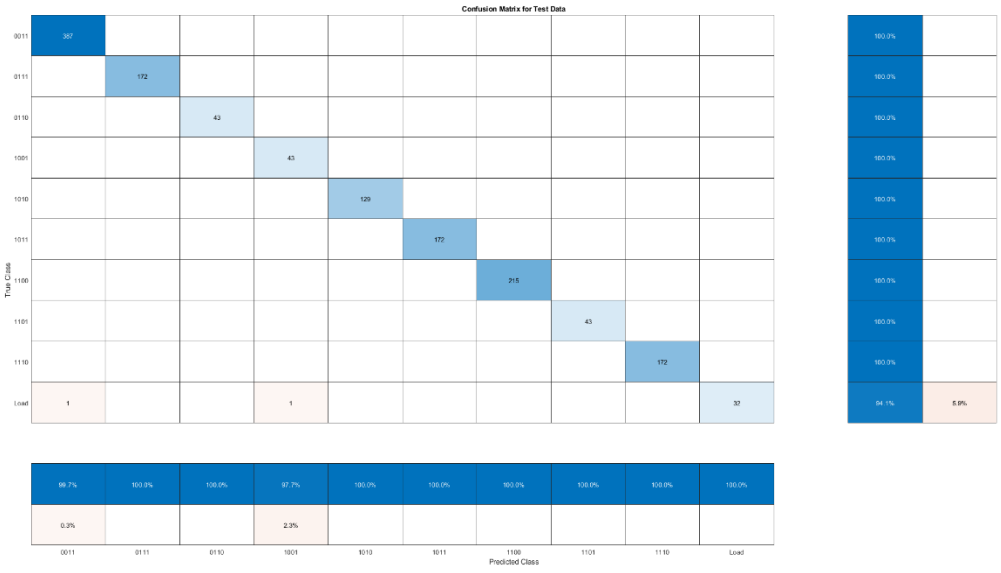


Figure 66 Confusion Matrix of Test Results After Training

8. WAVEFORM DATA FOR NEURAL NETWORK TRAINING

In order to fully represent the waveform, including intermediate and higher order harmonics, high sample rates are required. The waveform data captured by power quality meters and other modern instrumentation typically has sample rates of 3000 to 20,000 samples per second. The PSNN that leverages spectrograms and CNN is designed for these high sample rates.

Waveform data is becoming increasingly available throughout the industry from power quality meters, microprocessor-based relays, event recorders and other intelligent electronic devices (IED). However, at this point researchers do not yet have an organized and labeled power system waveform data set to use for neural network training. Comparable examples for other industries include:

- ImageNet: https://image-net.org/
- WordNet: https://wordnet.princeton.edu/
- FERET: https://www.nist.gov/programs-projects/face-recognition-technology-feret
- MNIST: http://yann.lecun.com/exdb/mnist/
- Among many others.

The most comprehensive data set for power systems analysis is the PMU data set for FOA 1861 (https://www.naspi.org/). The FOA 1861 data set is not publicly available yet but is in the process of being curated and posted on the web through a public resource. The PMU data is at a sample rate of 30 to 60 samples per second which is insufficient for detailed waveform representation particularly harmonic characteristics.

Given that large and comprehensive publicly available waveform data sets are not yet available to researchers, other approaches will need to fill the gap until the data is available. Some data has been acquired from EPB of Chattanooga and has been cross referenced with Outage Management System (OMS) records which provides for some labeling of the dataset. Other waveforms from EPB are being investigated and labeled using scripts developed in-house by EPB to assist in the labeling of other events not covered by OMS records. The EPB data set provides a starting point for researchers at ORNL, but is proprietary information and cannot be shared with the public.

Another project underway at ORNL is the signature library. The signature library is being developed to warehouse collections of power system data such as waveforms and PMU data. The signature library project is working with utilities such as EPB and others willing to share sets of sanitized power system data.

Other event types not covered by the EPB data set or other available data sets are not yet available such as events like:

- cold load pickup
- capacitor switching
- switching transients
- switching
- etc.

While the event waveform captures are being recorded for a wide variety of events throughout the industry, at this point many of the events are unlabeled. In other cases, the event waveform captures do not exist yet. Until the industry gets to a point where event waveform captures from the field can be used for neural network training simulated data will need to fill the gap in the interim.

Simulated data often consists of idealized examples of events and may not include characteristics from field conditions. Therefore, simulated data can be used as a starting point, however simulation tools will need to be improved as time progresses in order to generate waveforms that more closely reflect field conditions. To that end, we propose here a process by which waveforms are simulated and refined. As the simulated data is being used field examples are collected. As these field samples are collected, they are compared against the simulated waveforms. With understanding gained from this comparative analysis, the simulation models are adjusted to create more realistic waveforms. This cycle repeats, allowing for

new simulated data to be created with the improved simulation tools. The cycle repeats until a point is reached at which there is sufficient field data to successfully train the neural networks and simulated data can be shelved or used for more rudimentary training purposes (see Figure 67 Data Cycle).

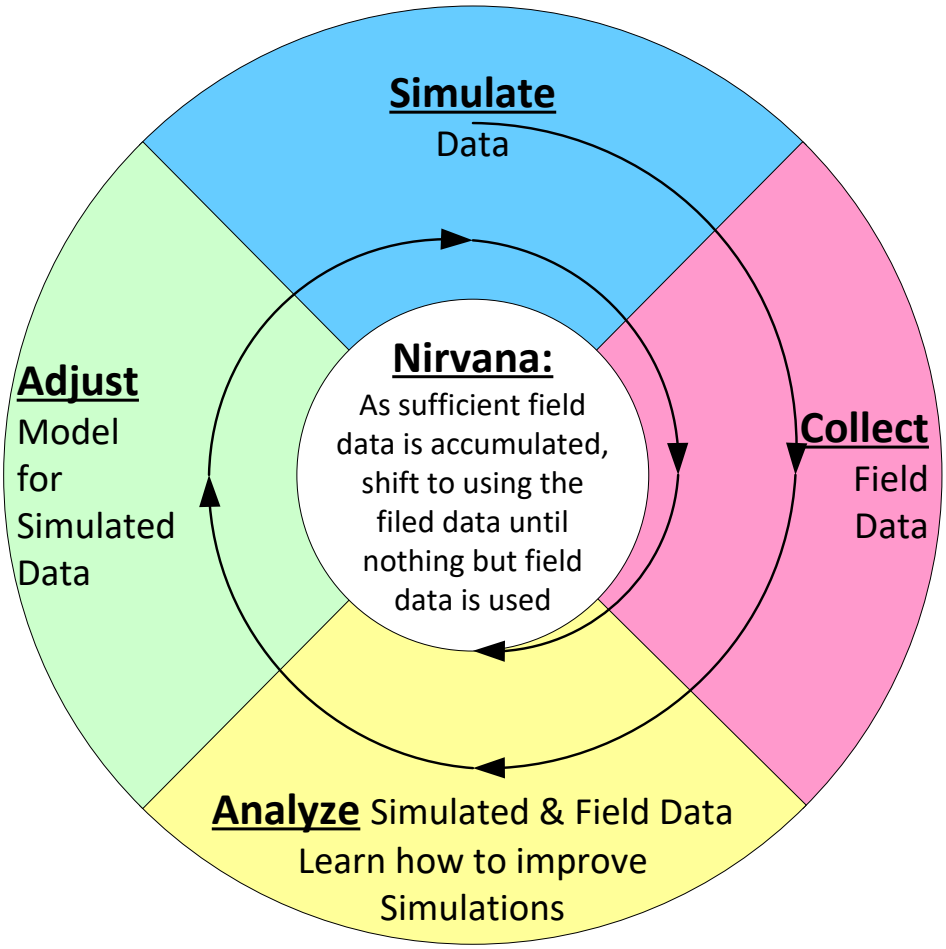


Figure 67 Data Cycle

9. FINDINGS AND CONCLUSIONS

A Power System Neural Network (PSNN) has been developed to use a Convolutional Neural Network (CNN) to classify events within waveform data for power system functions such as power system protection, power system planning, and power system operations.

The process follows the following procedure:

1) Normalize the voltage waveform data using the pseudo-Per-Unit calculations
2) Compute Power Spectrograms of the normalized waveform data using the Fcn_PSysSpectIV function which leverages the Power System Filter-Bank in the PSysFilterBank function (see Figure 59)
    a) For many applications set:
        i) NFFT = 1024 (unless the window width is smaller)
        ii) Top harmonic = 15 (unless higher order harmonics are of interest)
3) Flip the current spectrograms (see Figure 60)

4) Combine the voltage spectrograms with the flipped current spectrograms into single spectrogram images (see Figure 61)
5) Apply the Logarithm function to the combined spectrograms (see (3)
6) Use the spectrogram images as input to a multi-layer CNN (see Figure 64)

The test results on independent simulated test and validation datasets show greater than 99% accuracy.

Field data needs to be collected, labeled, organized, and curated for further training and testing of the PSNN. A major effort in the use of the field data involves developing scripts to isolate the actual fault event from the other surrounding nonevent data. Further, work with the industry will require new code and considerable effort to manually and programmatically label the field data. Additionally, data for new events such as capacitor switching, among others, will need to be simulated and collected from the field.

The next step for this algorithm is to modify it into a sequence-to-sequence classification system. This will allow the technique to be used on streaming data which will allow the use of this technique on live streaming data such as for protective relay applications and in distribution management systems.

While the results thus far are based on simulated data, the performance of the PSNN is very promising and should work for a wide variety of power system conditions of interest. Ultimately, much of the custom code and tools used today and much of the manual effort expended today may be automated using this PSNN.

10. BIBLIOGRAPHY

[1] M.-F. N.-C. Y. a. W.-F. C. Guo, "Deep-Learning-Based Fault Classification Using Hilbert-Huang Transform and Convolutional Neural Network in Power Distribution Systems," *IEEE Sensors Journal,* vol. 19, no. 16, p. 6905–6913, 2019.

[2] K. a. P. W. P. Zhu, "Fault Classification of Power Distribution Cables by Detecting Decaying DC Components with Magnetic Sensing," *IEEE Transactions on Instrumentation and Measurement,* vol. 69, no. 5, p. 2016–2027, 2019.

[3] V. a. A. Y. Ashok, "Fault Diagnosis Scheme for Cross-Country Faults in Dual-Circuit Line with Emphasis on High-Impedance Fault Syndrome," *IEEE Systems Journal ,* 2020.

[4] X. B. S. a. S. M. Jiang, "Automated Distribution Network Fault Cause Identification with Advanced Similarity Metrics," *Transactions on Power Delivery,* vol. 36, no. 2, pp. 785-793, 2020.

[5] M. S. A. S. a. M. R. Bashkari, "Outage cause detection in power distribution systems based on data mining," *IEEE Transactions on Industrial Informatics,* vol. 17, no. 1, pp. 640-649, 2020.

[6] M. A. H. S. a. T. G. Jarrahi, "Novel change detection and fault classification scheme for ac microgrids," *IEEE Systems Journal,* vol. 14, no. 3, pp. 3987-3998, 2020.

[7] A. E. F. E.-S. a. M. S.-P. Hooshyar, "Fault type classification in microgrids including photovoltaic DGs," *IEEE Transactions on Smart Grid,* vol. 7, no. 5, pp. 2218-2229, 2015.

[8] Y.-L. e. a. Liang, "Typical fault cause recognition of single-phase-to-ground fault for overhead lines in nonsolidly earthed distribution networks," *IEEE Transactions on Industry Applications,* vol. 56, no. 6, pp. 6298-6306, 2020.

[9] A. a. J. F. Silverstein, "High-resolution, time-synchronized grid monitoring devices," NASPI (North American Synchrophasor Initiative), 2020.

[10] EPB Chattanooga, *System Wide S&C IntelliRupter® Waveform Captures, Shared under NDA.,* 2020.

[11] E. J. B. a. K. R. Avagaddi Prasad, "A review on fault classification methodologies in power transmission systems: Part—I," *Journal of electrical systems and information technology,* vol. 5, no. 1, pp. 48-60, 2018.

[12] E. J. B. a. K. R. Avagaddi Prasad, "A review on fault classification methodologies in power transmission systems: Part-II," *Journal of Electrical Systems and Information Technology,* vol. 5, no. 1, pp. 61-67, 2018.

[13] H. Xue and et al., "A Novel Deep Convolution Neural Network and Spectrogram Based Microgrid Power Quality Disturbances Classification Method," in *2020 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2020.

[14] V. Velardo, "Audio Signal Processing for Machine Learning," YouTube, 2020.

[15] M. Müller, Fundamentals of Music Processing, Springer International Publishing, 2015.

[16] Stanford Vision Lab, *ImageNet,* Stanford University, Princeton University, 2020.

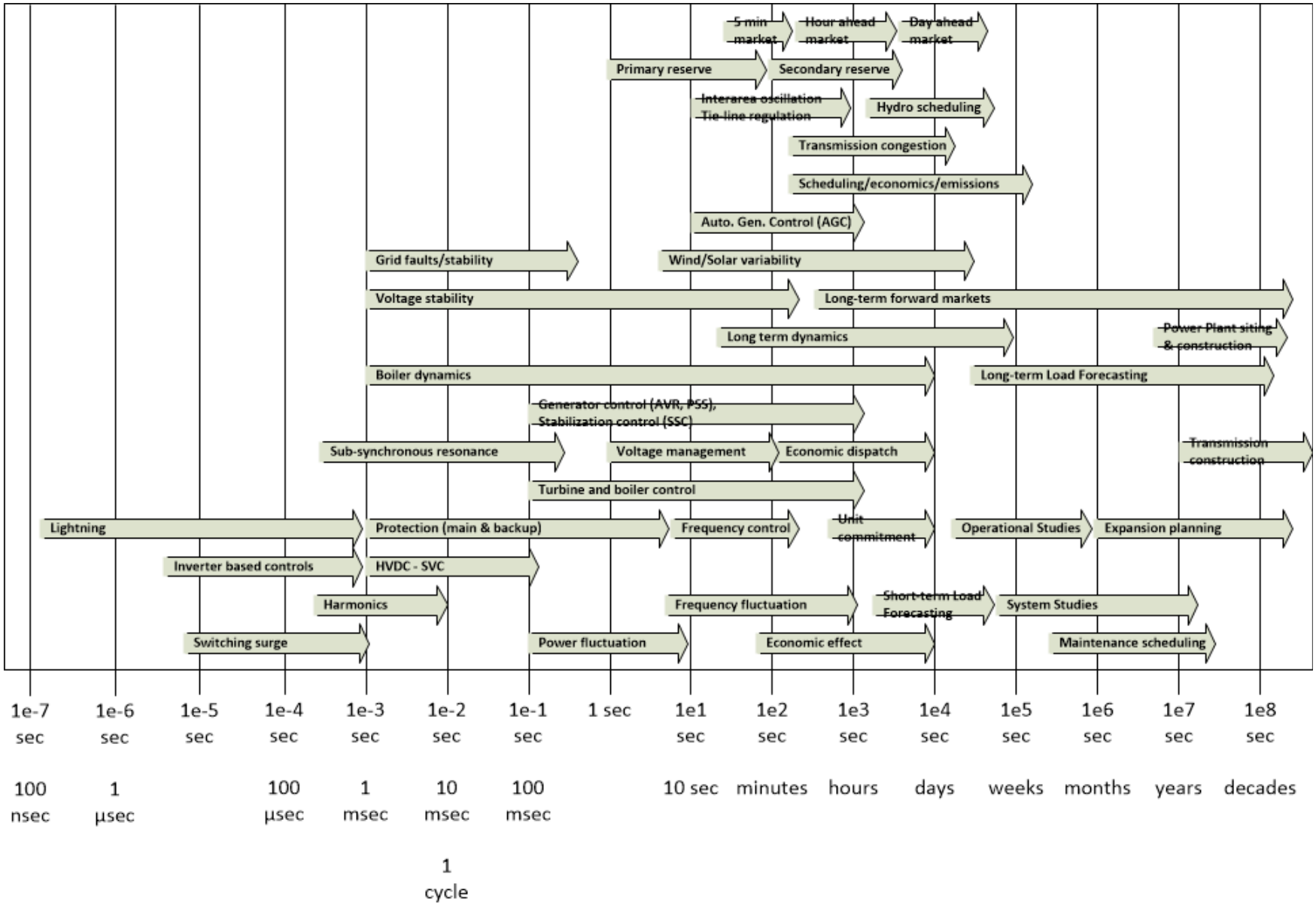
[17] DOE/EPRI, *DOE/EPRI National Database Repository of Power System Events.*

[18] The MathWorks, Inc., "Sequence-to-Sequence Classification Using Deep Learning".

[19] The MathWorks, Inc., "Speech Command Recognition Using Deep Learning".

[20] C. J. P. COMON, Handbook of Blind Source Separation, Independent Component Analysis and Applications, Amsterdam: Academic Press, Elsevier, 2010.

APPENDIX A. Event Timing


5 min market
Hour ahead market
Day ahead market
Primary reserve
Secondary reserve
Interarea oscillation
Tie-line regulation
Hydro scheduling
Transmission congestion
Scheduling/economics/emissions
Auto. Gen. Control (AGC)
Grid faults/stability
Wind/Solar variability
Voltage stability
Long-term forward markets
Long term dynamics
Power Plant siting & construction
Boiler dynamics
Long-term Load Forecasting
Generator control (AVR, PSS), Stabilization control (SSC)
Sub-synchronous resonance
Voltage management
Economic dispatch
Transmission construction
Turbine and boiler control
Lightning
Protection (main & backup)
Frequency control
Unit commitment
Operational Studies
Expansion planning
Inverter based controls
HVDC - SVC
Harmonics
Frequency fluctuation
Short-term Load Forecasting
System Studies
Switching surge
Power fluctuation
Economic effect
Maintenance scheduling
1e-7 sec
1e-6 sec
1e-5 sec
1e-4 sec
1e-3 sec
1e-2 sec
1e-1 sec
1 sec
1e1 sec
1e2 sec
1e3 sec
1e4 sec
1e5 sec
1e6 sec
1e7 sec
1e8 sec
100 nsec
1 µsec
100 µsec
1 msec
10 msec
100 msec
10 sec
minutes
hours
days
weeks
months
years
decades
1 cycle

APPENDIX B. Useful Reference Material for Machine Learning

10.1 BOOKS

Deep Learning, by Ian Goodfellow and Yoshua Bengio and Aaron Courville, An MIT Press book
https://www.deeplearningbook.org/

Fundamentals of Music Processing, by Meinard Müller, Springer International Publishing Switzerland 2015

10.2 YOUTUBE LECTURES & CLASSES

Neural Networks Demystified
https://youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
@stephencwelch
Welch Labs
7 videos
Last updated on Oct 2, 2015

Data Science: Machine Learning
https://learning.edx.org/course/course-v1:HarvardX+PH125.8x+2T2019/home
PH125.8x: Data Science: Machine Learning - Course Syllabus
Course Instructor: Rafael Irizarry
8th course of nine in the HarvardX Data Science Professional Certificate

Deep Learning
https://youtube.com/playlist?list=PLyqSpQzTE6M9gCgajvQbc68Hk_JKGBAYT
Prof. Mitesh M. Khapra
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
NPTEL-NOC IITM
118 videos
Last updated on Oct 25, 2018

Audio Signal Processing for Machine Learning
https://youtube.com/playlist?list=PL-wATfeyAMNqIee7cH3q1bh4QJFAaeNv0
Valerio Velardo - The Sound of AI
23 videos
Last updated on Oct 19, 2020

Deep Learning (for Audio) with Python
https://youtube.com/playlist?list=PL-wATfeyAMNrtbkCNsLcpoAyBBRJZVlnf
Valerio Velardo - The Sound of AI
19 videos
Last updated on Mar 9, 2020

10.3 MATLAB HELP

Help
https://www.mathworks.com/help/index.html

10.4 MATLAB EXAMPLES

Speech Command Recognition Using Deep Learning
https://www.mathworks.com/help/deeplearning/ug/deep-learning-speech-recognition.html

Sequence Classification Using Deep Learning
https://www.mathworks.com/help/deeplearning/ug/classify-sequence-data-using-lstm-networks.html

Sequence-to-Sequence Classification Using Deep Learning
https://www.mathworks.com/help/deeplearning/ug/sequence-to-sequence-classification-using-deep-learning.html

Sequence Classification Using 1-D Convolutions
https://www.mathworks.com/help/deeplearning/ug/sequence-classification-using-1-d-convolutions.html

Sequence-to-Sequence Classification Using 1-D Convolutions
https://www.mathworks.com/help/deeplearning/ug/sequence-to-sequence-classification-using-1-d-convolutions.html

10.5 OTHER HELPFUL YOUTUBE RESOURCES

Machine Learning & Deep Learning Fundamentals
https://youtube.com/playlist?list=PLZbbT5o_s2xq7LwI2y8_QtvuXZedL6tQU

3Blue1Brown
https://www.youtube.com/channel/UCYO_jab_esuFRV4b17AJtAw

10.6 MIT

Linear Algebra (Prof. Gilbert Strang)
https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/

Matrix Methods in Data Analysis, Signal Processing, and Machine Learning (Prof. Gilbert Strang)
https://ocw.mit.edu/courses/mathematics/18-065-matrix-methods-in-data-analysis-signal-processing-and-machine-learning-spring-2018/

MIT Deep Learning and Artificial Intelligence Lectures (Lex Fridman)
https://deeplearning.mit.edu/

APPENDIX C. MATLAB Code

11. MATLAB FUNCTION FOR POWER SYSTEM FILTER BANK

```
function [filterBank, numFilters, numHops] = Fcn_PSysFilterBank(DataStore,
DSparams, FFTparams)
```

Design Power System Filter Bank (PSysFilterBank)

Create Spectrograms to extract info for the filter bank

```
    [data,~] = read(DataStore);

    [S1,~,~] = stft(data{1,1}, DSparams.fs, 'Window', FFTparams.window,
'OverlapLength', FFTparams.noverlap, 'FrequencyRange', "onesided"); % presented
in dB
    S1 = abs(S1);
```

This is a power system filter bank

```
    % NEED TO ADDRESS THIS
    rowsOfS = size(S1,1);

    centerHarmonics = [1, 3, 5, 7, 9, 11, 13, 15, 21, 27, 33, 39, 45, 53, 59, 65,
71, 79, 85, 91, 97]; % odd to 15th then triplin Need DC at front end
    centerFrequencies = centerHarmonics * 60;
    frequencyList = [1, centerFrequencies, 6060]; % Add DC component to front and
101st harmonic to end
    freqRatio = rowsOfS/DSparams.fs;

    Centers = round(frequencyList * freqRatio);
    filterCount = numel(Centers)-1;

    % counts for cells surrounding the Center cell
    % counts for the cells to rise to the center
    riseCounts = zeros(1, filterCount);
    for i = 2:filterCount
        riseCounts(i) = Centers(i) - Centers(i-1);
    end
    % counts for the cells to fall from the center
    fallCounts = zeros(1, filterCount);
    for i = 1:filterCount
        fallCounts(i) = Centers(i+1) - Centers(i);
    end

    % Build the Filter Bank
    PSysFilterBank = zeros(filterCount,rowsOfS);
```

```
    for i = 1:filterCount
         % create vector of points on the rise curve
         filtRise = zeros(1,riseCounts(i)); % initialize vector of rise values
         for j = 1 : riseCounts(i)
             filtRise(j) = (j - 1) / (riseCounts(i));
         end

         % create vector of points on the fall curve
         filtFall = zeros(1,fallCounts(i)); % initialize vector of fall values
         for k = 1 : fallCounts(i)
             filtFall(k) = (k - 1) / (fallCounts(i));
         end
         filtFall = flip(filtFall);

         % combine rise/fall into filters and add them to the bank
         filter = [filtRise, 1, filtFall];
         leftPad = Centers(i)-riseCounts(i);
         rightPad = rowsOfS - numel(filter) - leftPad;
         PSysFilterBank(i,:) = [zeros(1,leftPad),filter,zeros(1,rightPad)];
    end

[numFilters, numHops] = size(PSysFilterBank*S1);
filterBank = PSysFilterBank;
end
```

12. MATLAB FUNCTION FOR CREATING THE SPECTROGRAMS

```
function X = Fcn_PSysSpectIV(DataStore, PSysFilterBank, DSparams, FFTparams)
```

Compute Power Spectrograms using the PSysFilterBank

Distribute the extraction of Power Spectrograms across multiple workers using `parfor (if possible)`.

First, determine the number of partitions for the dataset. If you do not have Parallel Computing Toolbox™, use a single partition.

```
    if ~isempty(ver('parallel'))
        pool = gcp;
        numPar = numpartitions(DataStore,pool);
    else
        numPar = 1;
    end
```

For each partition, read from the datastore, then extract the Power Spectrograms .

```
    numPar = 1;
```

```
    for partitionIDX = 1:numPar

%     parfor partitionIDX = 1:numPar




        subds = partition(DataStore,numPar,partitionIDX);
        partitionFileCount = numel(subds.Files);
        Xpartition = zeros(FFTparams.numHops,FFTparams.numFilters*2+1, ...
            DSparams.signalCount/2,partitionFileCount);
        for fileIDX = 1:partitionFileCount
            [data,~] = read(subds);
```

Normalize the voltage (rows 1-3) data. Put into PerUnit.

```
            for jj = 1:DSparams.signalCount/2
                data{jj,1} = data{jj,1} / (DSparams.BaseVoltage * 1e3) /
sqrt(2/3);
            end
```

Perform Spectrogram Calculations for each signal

```
            specSet =
zeros(FFTparams.numHops,FFTparams.numFilters,DSparams.signalCount);
            for j = 1:DSparams.signalCount
                [S,~,~] = stft(data{j,1}, DSparams.fs, 'Window',
FFTparams.window, 'OverlapLength', ...
                    FFTparams.noverlap, 'FrequencyRange', "onesided"); %
presented in dB
                S = abs(S);
                spec = PSysFilterBank*S;
                specSet(:,:,j) = spec';
            end
```

Flip the current signal spectrogram from left to right

```
            for i = DSparams.signalCount/2+1:DSparams.signalCount
                specSet(:,:,i) = fliplr(specSet(:,:,i));
            end
```

Combine Current and Voltage into single image separated by a colum of zeros. Put the current first and voltage second

```
            specImage =
zeros(FFTparams.numHops,FFTparams.numFilters*2+1,DSparams.signalCount/2);
            zeroColumn = zeros(FFTparams.numHops,1);
            for k = 1:DSparams.signalCount/2
                current = specSet(:,:,k+3);
                voltage = specSet(:,:,k);
                specImage(:,:,k) = [current, zeroColumn, voltage];
            end
```

Combine the spectrograms into one set called X

```
            Xpartition(:,:,:,fileIDX) = specImage;
        end
        XCombined{partitionIDX} = Xpartition;
    end
```

Convert the output to a 4-dimensional array with auditory spectrograms along the fourth dimension.

```
    X = cat(4,XCombined{:});
```

The spectrograms are post-processed by applying a logarithm. Taking a log of small numbers can lead to roundoff error.

Scale the features by the window power and then take the log. To obtain data with a smoother distribution, take the logarithm of the spectrograms using a small offset.

```
    epsil = 1e-6;
    X = log10(X + epsil);
end
```

13. MATLAB CODE FOR POWER SYSTEM NEURAL NETWORK (MAIN)

Power System Waveforms Analyzed using Spectrograms and CNN

```
clc
clear
PATH = Fcn_PSysPATH(0); % use 0 for full dataset, use 1 for limited dataset
```

Setup & Load Data Set

```
% Datastore Parameters
DSparams = Fcn_PSysDSparams();


% FFT Parameters
FFTparams = Fcn_PSysFFTparams(DSparams);


X = struct;
Y = struct;
DS = struct;
```

Create Datastores

Train Datastore

```
[DS.Train, Y.Train] = Fcn_PSysDatastore(PATH.train, DSparams);
DSparams.labels = categorical(categories(Y.Train))';
```

Validation Datastore

```
[DS.Validate, Y.Validation] = Fcn_PSysDatastore(PATH.validate, DSparams);
```

Test Datastore

```
[DS.Test, Y.Test] = Fcn_PSysDatastore(PATH.test, DSparams);
```

Design Power System Filter Bank (PSysFilterBank) using the function

```
[PSysFilterBank, FFTparams.numFilters, FFTparams.numHops] = ...
    Fcn_PSysFilterBank(DS.Validate, DSparams, FFTparams);
```

Compute Power Spectrograms

Compute Power Spectrograms using the PSysFilterBank on the Train Set

```
X.Train = Fcn_PSysSpectIV(DS.Train, PSysFilterBank, DSparams, FFTparams);
```

Compute Power Spectrograms using the PSysFilterBank on the Validation Set

```
X.Validation = Fcn_PSysSpectIV(DS.Validate, PSysFilterBank, DSparams, FFTparams);
```

Compute Power Spectrograms using the PSysFilterBank on the Test Set

```
X.Test = Fcn_PSysSpectIV(DS.Test, PSysFilterBank, DSparams, FFTparams);
```

Define Neural Network Architecture

Create a simple network architecture as an array of layers. Use convolutional and batch normalization layers, and down sample the feature maps "spatially" (that is, in time and frequency) using max pooling layers. Add a final max pooling layer that pools the input feature map globally over time. This enforces (approximate) time-translation invariance in the input spectrograms, allowing the network to perform the same classification independent of the exact position in time. Global pooling also significantly reduces the number of parameters in the final fully connected layer. To reduce the possibility of the network memorizing specific features of the training data, add a small amount of dropout to the input to the last fully connected layer.

The network is small, as it has only five convolutional layers with few filters. `numF` controls the number of filters in the convolutional layers. To increase the accuracy of the network, try increasing the network depth by adding identical blocks of convolutional, batch normalization, and ReLU layers. You can also try increasing the number of convolutional filters by increasing `numCnnFilters`.

Use a weighted cross entropy classification loss. `weightedClassificationLayer(classWeights)` creates a custom classification layer that calculates the cross entropy loss with observations weighted by `classWeights`. Specify the class weights in the same order as the classes appear in `categories(Y.Train)`. To give each class equal total weight in the loss, use class weights that are inversely proportional to the number of training examples in each class. When using the Adam optimizer to train the network, the training algorithm is independent of the overall normalization of the class weights.

Dimension Parameters

```
% Network Variables
NNvars = Fcn_PSysNNvars(X.Train, Y.Train);
```

Training Options

Specify the training options. Use the Adam optimizer with a mini-batch size of 128. Train for 25 epochs and reduce the learning rate by a factor of 10 after 20 epochs.

```
NNoptions = trainingOptions('adam', ...
    'InitialLearnRate',NNvars.initialLearnRate, ...
    'MaxEpochs',NNvars.maxEpochs, ...
    'MiniBatchSize',NNvars.miniBatchSize, ...
    'Shuffle','every-epoch', ...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'ValidationData',{X.Validation,Y.Validation}, ...
    'ValidationFrequency',NNvars.validationFrequency, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',floor(NNvars.maxEpochs*2/3), ...
    'ExecutionEnvironment',"auto");
```

Network Architecture

```
% Network Layers
NNlayers = [
    imageInputLayer([NNvars.numHops NNvars.numBands NNvars.numChannels])

    convolution2dLayer(3,NNvars.numCnnFilters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(3,'Stride',2,'Padding','same')

    convolution2dLayer(3,2*NNvars.numCnnFilters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(3,'Stride',2,'Padding','same')

    convolution2dLayer(3,4*NNvars.numCnnFilters,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(3,'Stride',2,'Padding','same')

    convolution2dLayer(3,4*NNvars.numCnnFilters,'Padding','same')
```

```
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer([NNvars.timePoolSize,1])

    dropoutLayer(NNvars.dropoutProb)
    fullyConnectedLayer(NNvars.numClasses)
    softmaxLayer
    weightedClassificationLayer(NNvars.classWeights)];
```

Train the network.

```
NN = trainNetwork(X.Train,Y.Train,NNlayers,NNoptions);
```

Evaluate Trained Network

Calculate the final accuracy of the network on the training set (without data augmentation) and validation set.

```
Y.PredValidation = classify(NN,X.Validation);
Y.PredTrain = classify(NN,X.Train);
Y.PredTest = classify(NN,X.Test);


ERROR = struct;
ERROR.validation = mean(Y.PredValidation ~= Y.Validation);
ERROR.train = mean(Y.PredTrain ~= Y.Train);
ERROR.test = mean(Y.PredTest ~= Y.Test);


disp("Epochs: " + NNvars.maxEpochs)
disp("Dropout: " + NNvars.dropoutProb)
disp("CNN Filters: " + NNvars.numCnnFilters)


disp("Training error: " + ERROR.train*100 + "%")
disp("Validation error: " + ERROR.validation*100 + "%")
disp("Test error: " + ERROR.test*100 + "%")
```

Plot the confusion matrix

Display the precision and recall for each class by using column and row summaries. Sort the classes of the confusion matrix.

```
figure('Units','normalized','Position',[0.2 0.2 0.5 0.5]);
cm = confusionchart(Y.Test,Y.PredTest);
cm.Title = 'Confusion Matrix for Test Data';
cm.ColumnSummary = 'column-normalized';
cm.RowSummary = 'row-normalized';
sortClasses(cm, [DSparams.labels])
```

```
% Finish by sounding a train whistle
load train
sound(y);
clear y
```

14. MATLAB FUNCTION FOR DATA-STORE PARAMETERS

```
function Return = Fcn_PSysDSparams()


    DSparams = struct;
    DSparams.variableNames = ["VA","VB","VC","IA","IB","IC"];
    DSparams.signalCount = length(DSparams.variableNames);
    DSparams.powerFreq = 60; % power system frequency in cycles/sec
    DSparams.BaseCurrent = 1; % power system Base Current to convert to PU
    DSparams.BaseVoltage = 12.5; % power system Base Voltage in kV to convert to
PU
    DSparams.DS_factor = 1;
    DSparams.sampFreq = 20000; % Sample Frequency of the original data (not
Downsampled)
    DSparams.fs = DSparams.sampFreq / DSparams.DS_factor; %  Sample Frequency ; 1
(default) | positive scalar
    DSparams.labels = categorical();
    Return = DSparams;
end
```

15. MATLAB FUNCTION FOR FFT PARAMETERS

FFT Parameters

```
function Return = Fcn_PSysFFTparams(DSparams)
    sampCycle = Fcn_PSysSampCycle(DSparams);

    % Window
    FFTparams = struct;
    FFTparams.Nw = sampCycle.Two; % Window Width in samples; integer | vector |
[]
    FFTparams.window = parzenwin(FFTparams.Nw); % Nw | windowHann | windowHamm |
...

    % Hop/Overlap
    FFTparams.hop = sampCycle.Quarter;
    FFTparams.noverlap = FFTparams.Nw - FFTparams.hop; % Number of overlapped
samples; positive integer | []
```

```
    % Number of DFT points
    FFTparams.p = ceil(log2(FFTparams.Nw));
    FFTparams.nfftCalc = max(256, 2^FFTparams.p + 1);
    FFTparams.nfft = 1024; % Number of DFT points; positive integer scalar |
FFTparams.nfftCalc | []

    % find the harmonic range and idividual harmoincs of interest (odd)
    FFTparams.topHarmonic = 15; % using odd harmonics up to 15th,  51st or 101th
    FFTparams.oddHarmonics = (1:2:FFTparams.topHarmonic);
    FFTparams.f = (FFTparams.oddHarmonics * 60); % Cyclical Frequencies; vector
    FFTparams.w = pi./(FFTparams.f * sampCycle.One);  % Normalized frequencies;
vector

    FFTparams.freqrange = 'onesided'; % Frequency range for PSD estimate;
'onesided' | 'twosided' | 'centered'
    FFTparams.spectrumType = 'power'; % Power spectrum scaling; 'psd' (default) |
'power'
    FFTparams.freqloc = 'yaxis'; % Frequency display axis; 'xaxis' (default) |
'yaxis'

    % Based on the Filterbank that is created in code later
    FFTparams.numFilters = 1; % initialize value to be updated in code later
    FFTparams.numHops = 1; % initialize value to be updated in code later

    Return = FFTparams;
end
```

16. MATLAB FUNCTION FOR NEURAL NETWORK VARIABLES

Dimenstions of the Spectrograms

```
function Return = Fcn_PSysNNvars(XTrain, YTrain)
    NNvars = struct;
    [NNvars.numHops,NNvars.numBands,NNvars.numChannels,NNvars.numSpec] =
size(XTrain);

    % Training Options Variables
    NNvars.maxEpochs = 25; % initial: 25

    NNvars.miniBatchSize = 128; % initial: 128
    NNvars.initialLearnRate = 3e-4; % initial: 3e-4
    NNvars.validationFrequency = floor(numel(YTrain)/NNvars.miniBatchSize   );

    % Network Variables
    NNvars.dropoutProb = 0.2; % Started with 0.2
    NNvars.numCnnFilters = 25; % Started with 12. Had good success with 42

    NNvars.classWeights = 1./countcats(YTrain);
```

```
    NNvars.classWeights = NNvars.classWeights'/mean(NNvars.classWeights);
    NNvars.numClasses = numel(categories(YTrain));
    NNvars.timePoolSize = ceil(NNvars.numHops/8);



    Return = NNvars;
end
```

17. MATLAB FUNCTION FOR SAMPLE CYCLE

```
function Return = Fcn_PSysSampCycle(DSparams)


    % Window Width
    sampCycle = struct;
    sampCycle.Two = floor(DSparams.fs / DSparams.powerFreq /2)*4; % rounded to
nearest even
    sampCycle.One = floor(DSparams.fs / DSparams.powerFreq /2)*2; % rounded to
nearest even
    sampCycle.Half = floor(sampCycle.One / 2 /2)*2; % rounded to nearest even
    sampCycle.Quarter = floor(sampCycle.One / 4 /2)*2; % rounded to nearest even
    sampCycle.Eigth = floor(sampCycle.One / 8 /2)*2; % rounded to nearest even
    sampCycle.SixTeenth = floor(sampCycle.One / 16 /2)*2; % rounded to nearest
even
    sampCycle.ThirtySecond = floor(sampCycle.One / 32 /2)*2; % rounded to nearest
even
    sampCycle.SixtyFourth = floor(sampCycle.One / 64 /2)*2; % rounded to nearest
even
    Return = sampCycle;
end
```