ARTIFICIAL INTELLIGENCE-BASED ARC FAULT DETECTION FOR AC AND DC SYSTEMS

by

Kamal Chandra Paul

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2024

Approved by:

Dr. Tiefu Zhao

Dr. Chen Chen

Dr. Robert Cox

Dr. Shen-En Chen

ABSTRACT

KAMAL CHANDRA PAUL. Artificial Intelligence-based Arc Fault Detection for AC and DC Systems. (Under the direction of DR. TIEFU ZHAO)

The prevalence of electrical fires due to arc faults in both AC and DC systems has necessitated advancements in detection technologies. This dissertation introduces innovative artificial intelligence (AI)-based methods for efficient arc fault detection, enhancing both safety and reliability in electrical systems. For AC systems, a convolutional neural network (CNN)-based arc fault detection algorithm has been proposed, which has an arc fault detection accuracy of 99.47% with a balanced sampling rate of 10 kHz. The performance of the proposed algorithm has been verified using the Raspberry Pi 3B platform.

Although traditional CNN algorithms exhibit high accuracy, they require optimization for real-time arc fault detection on resource-limited hardware. To address this, a lightweight CNN architecture combined with a model compression technique using a knowledge distillation-based teacher-student algorithm has been proposed. This model maintains a high detection accuracy of 99.31% and achieves a minimal runtime of 0.20 ms per sample on the Raspberry Pi 3B platform, demonstrating its suitability for commercial embedded microcontrollers (MCUs) with limited computational capability.

Extending the research to DC systems, a cost-effective, AI-driven Arc Fault Circuit Interrupter (AFCI) for DC applications has been proposed. Utilizing an STM32 MCU and a silicon carbide (SiC) MOSFET-based solid-state circuit breaker (SSCB), this system achieves a detection accuracy of 98.14% with an arc fault interruption time of 19 ms. To enhance robustness, instead of making decisions on a single cycle of arc, a few consecutive cycle wait time has been proposed. This research demonstrated an arc fault clearing time of 95 ms considering 5 cycles of arcs as one arc fault event. This AFCI solution stands out for its rapid response and high reliability, promising significant improvements in safety for

DC-powered installations.

Together, these contributions signify a leap forward in the domain of electrical safety, presenting viable solutions for the timely detection and interruption of arc faults in both AC and DC systems. The outcomes of this research are expected to influence future standards and practices in electrical safety management across residential, commercial, and industrial sectors.

DEDICATION

This dissertation is dedicated with heartfelt gratitude to my beloved parents, Khagendra Nath Paul and Amala Rani Paul, whose unwavering love, support, and encouragement have been the guiding light throughout my life. Their sacrifices have shaped me into the person I am today, and I am eternally grateful for their unconditional love and belief in my abilities. I also dedicate this work to my son, Rishaan Paul, whose laughter and boundless energy have brought immense joy and inspiration to my life. This work is a testament to their love and support.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my advisor, Dr. Tiefu Zhao, for his invaluable guidance, support, and patience throughout my PhD journey. His expertise and mentorship have been instrumental in shaping this dissertation, and his encouragement has kept me motivated.

I am also deeply thankful to the members of my dissertation committee: Dr. Chen Chen, Dr. Robert Cox, and Dr. Shen-En Chen, for their invaluable feedback, which has greatly improved my research. Special thanks to Dr. Chen Chen for his profound insights into artificial intelligence and its applications, which have significantly enriched my work.

I am grateful to the Graduate School of the University of North Carolina at Charlotte for awarding me the GASP grant, which provided essential financial support covering both tuition and health insurance. I also express my gratitude to Dr. Zhao, the Department of Electrical and Computer Engineering, and UNC Charlotte for the assistantship support I received throughout my PhD program. This support has been crucial in allowing me to focus entirely on my research.

I want to express my deepest gratitude to my parents, my wife, my sister, and my son for their unwavering support and encouragement. Their love and belief in me have been a constant source of strength and inspiration.

A special thanks to my fellow lab members at RE-PEARL: Yafeng Wang, Haichen Liu, Xiwen Xu, Jiale Zhou, Mason Sun, Qiang Mu, and Shangze Chen, for their stimulating discussions and incredible support during my PhD journey. My appreciation also extends to my colleagues and friends at the University of North Carolina at Charlotte, whose camaraderie and support have made this journey more enjoyable.

Thank you to everyone who has contributed to this dissertation. Your support has meant the world to me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC	Alternating Current
AFCI	Arc Fault Circuit Interrupter
AFDD	Arc Fault Detection Device
AI	Artificial Intelligence
ANN	Artificial Neural Network
ArcNet	AC Arc Fault Detection Network
CNN	Convolutinal Neural Network
CWT	Continuous Wavelet Transform
DC	Direct Current
DL	Deep Learning
DNN	Deep Neural Network
DT	Decision Tree
DW	Depthwise
DWT	Discrete Wavelet Transform
ECE	Electrical and Computer Engineering
EL	Ensemble Learning
EML	Ensemble Machine Learning
FC	Fully Connected

FFT	Fast Fourier Transform
GBDT	Gradient Boosted Decision Tree
HMM	Hidden Markov Model
kHz	Kilo Hertz
KNN	K-Nearest Neighbors
KS/s	Kilo Sampling Per Second
LSTM	Long Short Term Memory
MCU	Microcontroller Unit
ML	Machine Learning
MP	Max Pooling
NC	North Carolina
NEC	National Electric Code
PADNet	Photovoltaic Arc Fault Detection Network
PADNet-Lite	Lightweight Photovoltaic Arc Fault Detection Network
RF	Random Forest
RNN	Recurrent Neural Network
SMPS	Switch-Mode Power Supply
SSCB	Solid-State Circuit Breaker
SSTDR	Spread Spectrum Time Domain Reflectometry

STFT	Short-Time Fourier Transform
SVM	Support Vector Machine
US	United States
WPD	Wavelet Packet Decomposition
WPT	Wavelet Packet Transform

CHAPTER 1: INTRODUCTION

1.1 Background of Arc Fault

An arc fault is characterized by the luminous discharge of electricity between two conductors. This arc fault poses significant safety hazards across various electrical systems, including both alternating current (AC) and direct current (DC) circuits. These faults are often caused by many factors such as loose cable connections, wire aging, insulation breakdown due to external intrusion, or physical damage to cables (Fig. 1.1). Arc fault can happen even at low currents ranging from 3A to 12A. It can elevate the temperatures beyond 5000^oC, [1] and eventually may lead to electrical fire hazards, causing property damage, injury, or even fatalities. In the United States, arc faults are a leading cause of residential fire hazards. A survey by Zebra found that 36.3% of fire hazards are attributed to electrical issues [2], while the Industrial Safety and Hygiene News reports over 30,000 arc flash incidents annually, leading to significant casualties and hospitalizations. Series AC arc fault detection is challenging because it inserts series impedance when it occurs and reduces the load current instead of increasing it. Therefore, traditional overcurrent protective equipment can not detect arc faults. Rather, an arc fault detection device (AFDD) or Arc Fault Circuit Interrupter (AFCI) is required to prevent these fire hazards. However, the nature of the load current, switching noise, and environmental noise cause a lot of false detection resulting in nuisance tripping of AFDDs or AFCIs. Standards set by entities like the International Electrotechnical Commission (IEC), National Electrical Code (NEC), and Underwriters' Laboratories (UL) mandate arc fault detection devices in household and industrial systems for safety [3–5].

The U.S. solar industry has experienced remarkable growth, with over 140 GW of solar

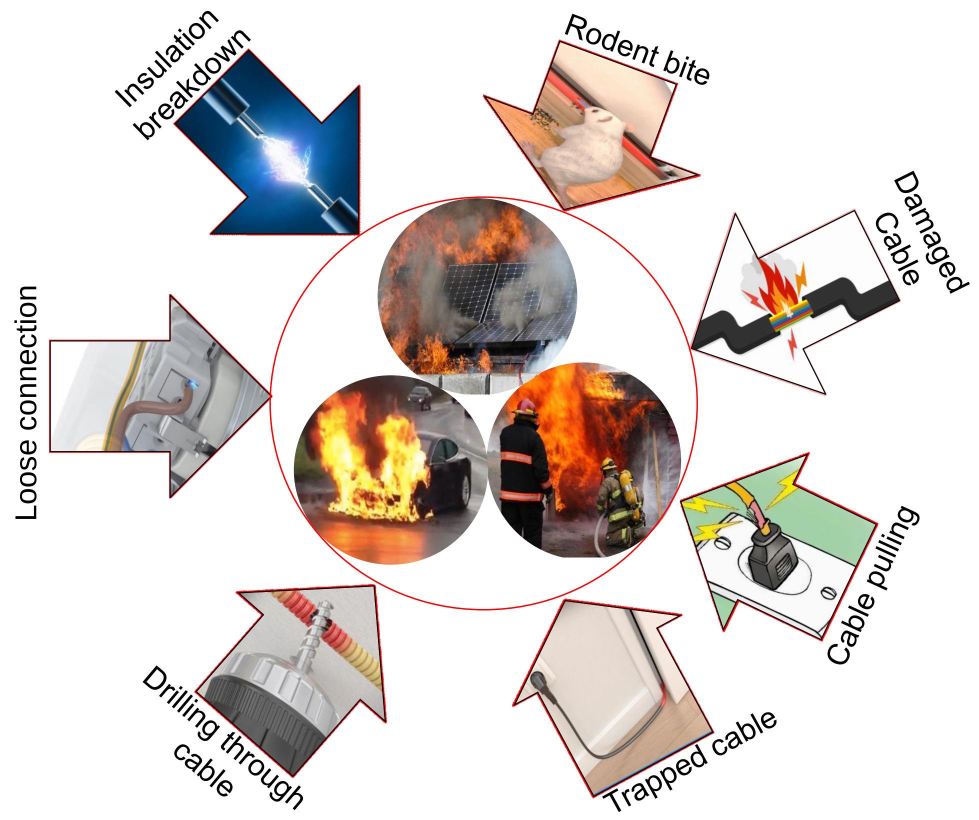


Figure 1.1: Causes of arc faults.

capacity installed and an ambition to greatly expand this by 2033 [8–10] (Fig. 1.4). This expansion underscores the increasing importance of addressing safety concerns due to series arc faults inherent in DC systems. Arc fault is stochastic and can happen anywhere in a system (refer to Fig. 1.5). A few recent incidents of PV arc fault fire hazards are depicted in Fig. 1.2 and Fig. 1.3. Series arc faults in PV systems are particularly challenging to detect due to noise interference from various sources such as radio frequency interference, power electronics noise, and electromagnetic interference (EMI). These conditions exacerbate the difficulty of discerning arc faults from normal operational signals [1, 11]. In response to these risks, the NEC 2020 sec. 690.11 mandates the use of PV AFCIs for systems operating at or above 80 V DC. Apart from this, the AFCI standard requires arc faults must be extinguished before dissipating 750 J of energy or within 2.5 s of arcing, whichever occurs first [4].

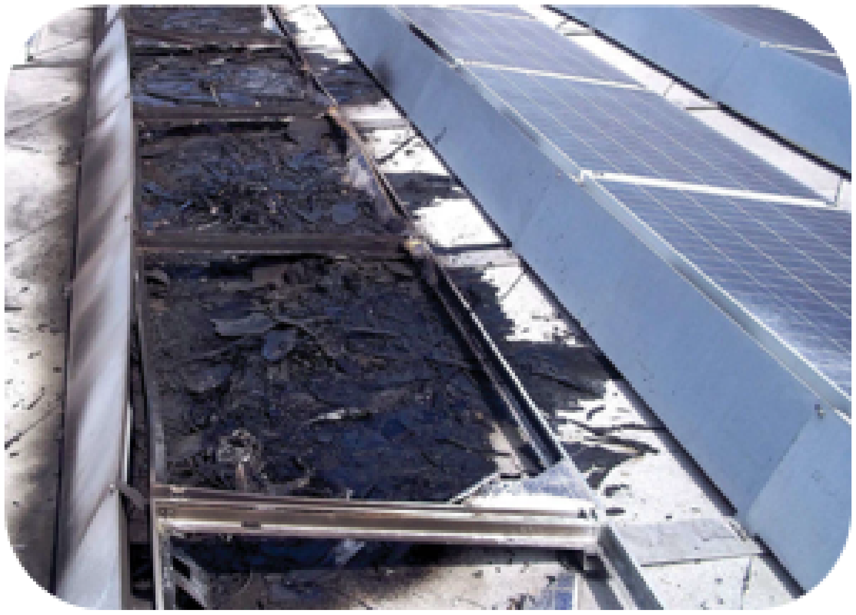


Figure 1.2: Arc fault fire hazard: 1,208kW PV array, Mount Holly, North Carolina, 2011 [6].

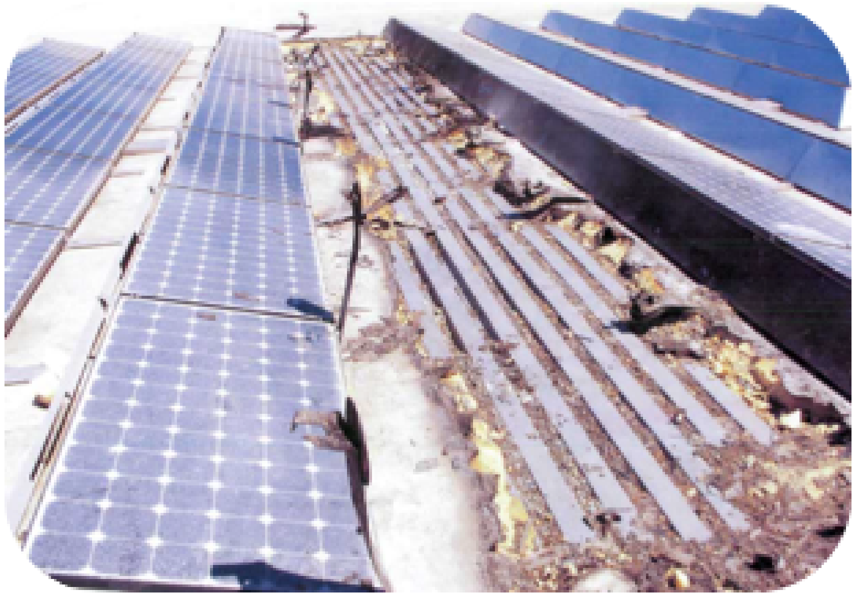


Figure 1.3: Arc fault fire hazard: 383 kW PV system, Bakersfield, CA, 2009 [7].

Arc faults are two types. One is a series arc fault and the other is a parallel arc fault (see Fig. 1.6). Unlike series arc fault, parallel arc fault heavily increases the line current and usually easily noticeable by an overcurrent protective device. As the load current reduces due to a series arc fault, it can not be detected by a regular circuit breaker.

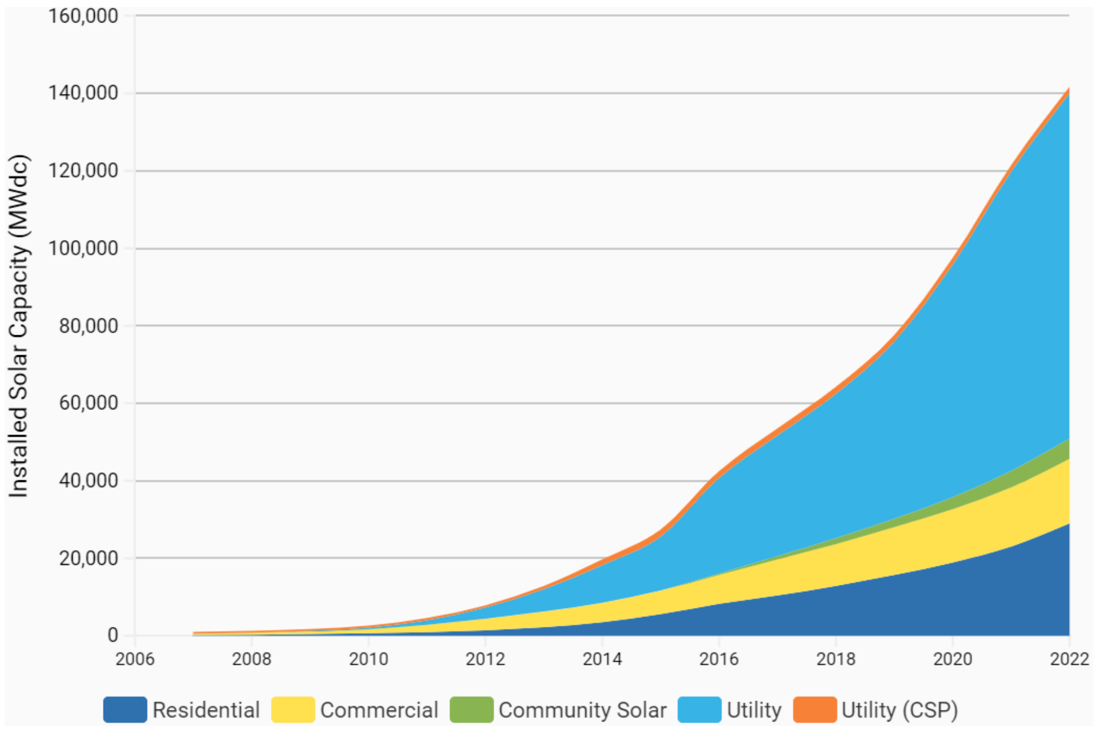


Figure 1.4: Cumulative U.S. solar installations over the period [8].

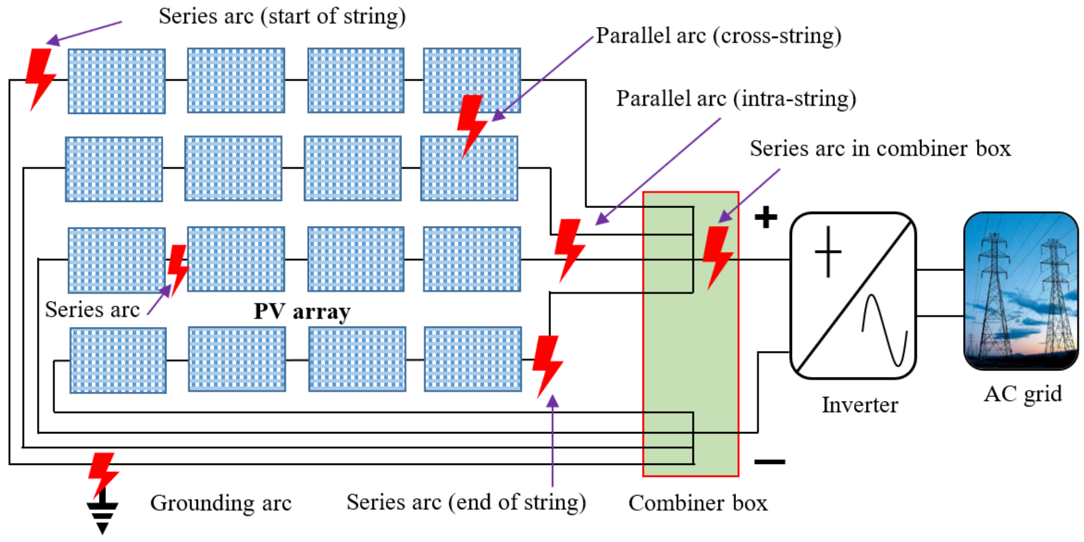

Series arc (start of string)
Parallel arc (cross-string)
Parallel arc (intra-string)
Series arc in combiner box
+
Series arc
PV array
Inverter
AC grid
−
Grounding arc
Series arc (end of string)
Combiner box

Figure 1.5: Power flow diagram of a photovoltaic system with locations where series and parallel arc faults can happen

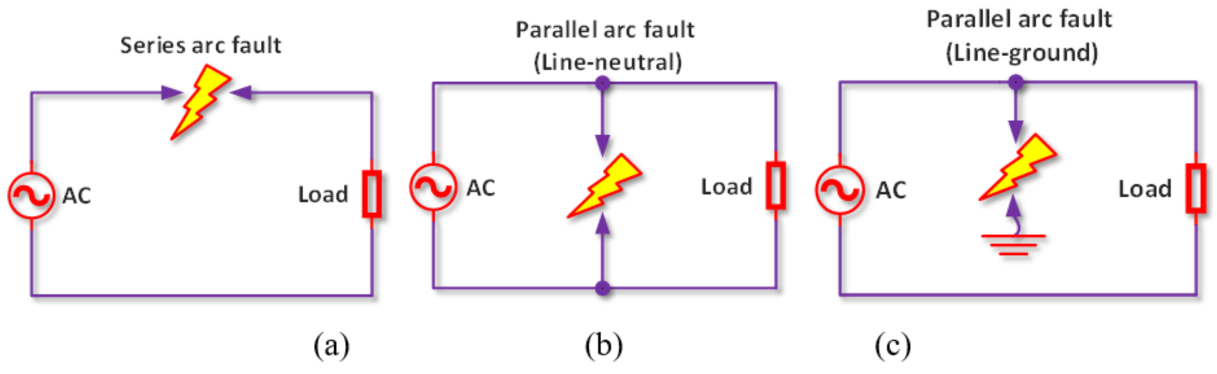


Figure 1.6: Types of arc faults- (a) series arc fault; (b) parallel arc fault (line-neutral); (c) parallel arc fault (line-ground).

1.2 Detection of Series Arc Faults

Conventional arc fault detection techniques, including Fast Fourier Transform (FFT), Wavelet Transform (WT), and Discrete Wavelet Transform (DWT), are employed in both AC and DC circuits but are limited by the necessity for manual threshold adjustments, which can lead to false positives under varying load conditions [12–16]. The challenges are further compounded by the presence of noise which distorts the frequency spectrum, affecting the accuracy of these methods [17].

Due to their high classification accuracy, artificial intelligence (AI) and machine learning (ML) based algorithms have increasingly become the focus of research in arc fault detection. Different classification algorithms, including Support Vector Machine (SVM), Particle Swarm Optimization combined with Self-Organizing Map Neural Network, Recurrent Neural Network (RNN), and Convolutional Neural Network [18], have been employed for arc fault classification, often coupled with data preprocessing techniques. These AI-based methods offer advantages in automatically generating features without the need for hand-adjusted thresholds, thereby increasing reliability and reducing the likelihood of false positives [11, 18–33].

1.3 Key Technology for Arc Fault Protection

In order to prevent arc fault fire hazards and enhance electrical safety and reliability, it is required to develop an efficient arc fault detection algorithm having accurate and fast detection capability. AI-based methods have high arc fault detection capability and they are flexible, and adaptable compared to traditional signal processing-based algorithms. Therefore, the development of efficient, lightweight deep learning (DL) models for arc fault detection is critical for both AC and DC systems [34–38]. An efficient and lightweight model can reduce the computational complexity and can easily be implemented in a resource-limited embedded device for real-time operation. Moreover, the platform to run the algorithm has to be cost-effective and affordable. After arc fault detection, the faulty part should be isolated from the healthy part as quickly as possible. A solid-state circuit breaker (SSCB) can be integrated into the model for quicker interruption of the circuit.

1.4 Overview of Proposed Research

The main goal of my research is to create a highly efficient arc fault detection algorithm that is both accurate and low-cost. This algorithm is designed to be easily integrated into affordable microcontrollers for quick fault interruption using a solid-state circuit breaker. An overview of the proposed works is provided below.

1. **Develop an AC arc fault detection algorithm utilizing traditional CNN methods:** This dissertation focuses on developing an AC arc fault detection algorithm using advanced CNN methods. This algorithm can directly process raw electrical current data, which simplifies the process by removing the need for data preprocessing. Tested on a Raspberry Pi 3B, this approach achieves impressive accuracy of over 99% and provides quick fault detection within 31 ms.

2. **Optimize the AC arc fault detection model for computational efficiency:** This research aims to make the AC arc fault detection model more computationally

efficient. By using a lightweight CNN architecture combined with a teacher student knowledge distillation (KD) method the model is optimized. The optimized model achieves an extremely fast runtime of just 0.2 ms on the Raspberry Pi 3B. This significant improvement ensures the model is well-suited for low-cost microcontrollers without compromising on performance. It requires only 57.93 KB of flash and only 13.04 KB of RAM when implemented on STM32 MCU.

3. **Design and validate a DC AFCI using a lightweight CNN approach and SSCB:** Finally, the research focuses on designing and validating a DC AFCI that uses a streamlined CNN algorithm for detecting PV arc faults and an SSCB for fast interruption. This AI-based AFCI is optimized for real-time use in embedded hardware and is verified using an STM32 microcontroller. The algorithm is highly efficient, requiring only 12.6 KB of RAM and 34 KB of flash memory, and it achieves the fastest arc fault clearing time of 19 ms. This design not only ensures rapid fault clearing but also enhances the overall reliability and robustness of the system.

CHAPTER 2: LITERATURE REVIEW

2.1 Arc Faults in Photovoltaic Systems

In photovoltaic (PV) systems, solar cells are typically interconnected in series and parallel configurations to amplify the voltage and current levels. However, this increase in interconnected solar panels also leads to more junction points, which raises the likelihood of loose connections and potential arc faults, posing significant safety risks. An illustration of a typical PV system setup, depicting the potential locations of PV arc faults, is shown in Fig. 1.5.

Detecting series arc faults in DC systems presents unique challenges. One significant issue is that PV systems can continue operating and supplying power even during an arc fault, which increases the risk of fires and electrical hazards. The challenge is further compounded by the nature of PV modules, which are current-limited devices. This makes it difficult to differentiate between normal operations and fault conditions. Additionally, the presence of numerous conductor joints within the PV system adds complexity to pinpointing the exact location of a series arc fault.

PV systems can experience two main types of arc faults: series and parallel, with ground faults being a subset of parallel faults. Parallel arc faults typically result in a surge in current, making them easier to detect with traditional circuit breakers. In extreme cases, this surge can be strong enough to melt a fuse. Conversely, series arc faults introduce additional impedance into the load, thereby reducing the load current and preventing fuse melting. This reduction in load current also makes series arc faults difficult to detect with standard circuit breakers.

While parallel arc faults are relatively straightforward to detect due to the high current they generate, detecting series AC arc faults is significantly more challenging. This is because the impedance created by the fault results in a decrease, rather than an increase, in current amplitude. Identifying series arc faults is further complicated by certain loads, such as those with switch-mode power supplies, which can exhibit arc-like current patterns under normal conditions. This often leads to false alarms and nuisance trips. Recognizing these challenges, researchers have utilized a variety of methods, both conventional and based on artificial intelligence, to address this complex issue [39].

An arc fault is stochastic by nature and can occur anywhere in the photovoltaic (PV) system. Terzija et al. [40] and Lu et al. [16] have described the random behavior of an arc as Gaussian noise:

$$x(t) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{t^2}{2\sigma^2}} \tag{2.1}$$

where $x(t)$ represents the signal in the time domain. In the frequency domain, it is expressed as:

$$X(f) = \mathcal{F}x(t) = \frac{1}{\sqrt{2\pi}} \cdot \sigma \cdot e^{-2\pi^2\sigma^2 f^2} \tag{2.2}$$

with $X(f)$ representing the signal in the frequency domain.

Johnson et al. [41] also stated that the signal exhibits white noise characteristics, meaning that it has a constant power spectral density (PSD) and contains all frequencies with equal intensity:

$$\text{Spectral density } S(f) = const. \tag{2.3}$$

Therefore, the arc can be considered as white Gaussian noise.

Moreover, the output voltage and current signal of the PV system are contaminated with various noise components from different sources, including radio frequency interference, power electronics noise such as inverter switching frequency, electromagnetic interference (EMI), and noise due to changes in irradiance, temperature, and load conditions. These noises distort the signals, further complicating the detection of series DC arc faults in PV systems. The stochastic nature of an arc fault's location and the series impedance it introduces reduce the signal quality, making the effective recognition of the series arc more challenging. This impacts the reliability of the PV system.

Currently, commercial products for PV arc fault detection (AFD) devices are scarce and not cost-effective. Their accuracy is also significantly affected by various noise and loading conditions [30, 42, 43].

2.2 Arc Fault Detection Standards

Standardization is key to ensuring the safety and effectiveness of arc fault detection devices (AFDDs) in photovoltaic (PV) systems. It guarantees that these devices work seamlessly across different PV systems and are compatible with equipment from various manufacturers. Standardization also ensures transparency and accountability in the development and deployment of AFDDs, fostering confidence in their performance and reliability.

The Underwriters Laboratories (UL) established the UL 1699B standards for AFCIs or AFDDs [44], pivotal for the safety of PV systems. According to these standards, an AFCI is required to interrupt an arc fault within 2.5 seconds or before the arc energy reaches 750 joules, whichever happens first. This critical requirement is aimed at providing prompt intervention to avert potential dangers. A device is deemed non-compliant if it fails to meet these standards. Additionally, UL 1699B specifies detailed testing conditions for AFCIs, including the arcing gap distance, minimum arcing current, and open circuit voltages, to ensure comprehensive evaluation and consistent reliability.

Furthermore, the National Electrical Code (NEC) mandates that AFDDs must be installed in all PV systems operating at voltages above 80 V [5]. The significance of effective arc fault detection in PV systems was recognized with its incorporation into article 690 of UL 1699B in 2011 [41,45,46]. These standards and regulations are instrumental in ensuring the safe functioning of PV systems and in reducing the risks associated with arc faults. Compliance with these standards is vital for upholding the safety and reliability of PV systems.

2.3 Traditional Methods of Arc Fault Detection

A variety of features can be harnessed for the detection of arc faults, ranging from observable characteristics like arc flash to the audible sounds produced by arcs. Depending on the approach, the input for detection can be in the time domain, frequency domain, or a combination of both. The most commonly utilized input types are:

- Time-domain current and voltage signals (raw values) [47–53]
- Derivatives of current and voltage, such as averages or the difference between maximum and minimum values, median, variance, RMS values etc. [54–56]
- Processed signals derived from current and/or voltage, including:
  - Fast Fourier Transform (FFT) [49,57]
  - Short Time Fourier Transform (STFT) [32,48,57,58]
  - Wavelet Transform (WT) [48,52,56,57,59–61]
  - Other signal processing techniques including Empirical Mode Decomposition (EMD), Variational Mode Decomposition (VMD), Hilbert-Huang transform (HHT), independent component analysis (ICA), power spectral density (PSD) analysis, correlation analysis, statistical signal processing, and adaptive filtering techniques [40,61–65]
- Additional detection modalities, including:

- Electromagnetic radiation (EMR) [66]
- Machine vision and image processing [67]
- Infrared imaging
- Power quality analysis
- Temperature, acoustic, and gas measurements
- Optical sensing [68, 69]
- Electrical impedance measurements

Among these, current and voltage measurements are the most frequently employed features. Voltage measurement poses particular challenges, especially when the precise fault location within the system is unknown [70]. Moreover, to thoroughly monitor a system, a larger array of voltage sensors is often required compared to current sensors.

As a result, the industry commonly focuses on current measurements for arc fault detection. Traditional methods extract key features from these measurements, such as the average value, peak-to-peak value, and standard deviation. In contrast, modern deep learning techniques use raw data, taking advantage of the powerful feature extraction capabilities of convolutional neural networks. These methods make arc fault detection more effective by avoiding the challenges of voltage measurement and reducing the need for an extensive network of voltage sensors.

2.3.1 Time-Domain and Statistics-based Algorithm

The occurrence of an arc fault typically results in an immediate drop in the line current, which serves as a distinct and measurable characteristic. Time-domain features and parameters are particularly beneficial for embedded systems due to their rapid computability and ease of implementation. Additionally, time-domain analysis can be conducted with lower sampling frequencies, which significantly cuts down on computational time [54]. A few feature extraction methods with mathematical expressions in the time domain are enlisted in Table 2.1.

Monitoring the current is one of the simplest detection methods. Current signals are straightforward to measure, rich in distinctive features, and do not require preprocessing. As a result, several studies [11, 18, 50, 51, 67, 71, 72] have employed raw DC current signals directly. Others [54, 73] have focused on specific features derived from the DC current, such as analyzing variations in the line current [70, 74, 75]. However, a notable drawback is that when calculating statistical metrics like mean, standard deviation, and variance, vital fault information that may be present in the raw data could potentially be obscured [60].

It is important to recognize that the fault current signal associated with nonlinear loads is unstable and varies randomly, which can hinder the accurate detection of arc faults solely based on time-domain characteristics. Chen et al. [49] discussed how the bus current experiences significant fluctuations following an arc fault. Furthermore, the presence of electromagnetic interference noise from converters/inverters and the switching of electrical drives in DC systems adds layers of complexity to arc fault detection, complicating the identification process.

2.3.2 Fourier Transformation

The Fast Fourier Transformation (FFT) is a widely utilized method for extracting useful features in DC arc fault detection. Dang et al. [77] have indicated that deep learning techniques achieve greater success with frequency-domain input. When an arc fault occurs, there is a noticeable increase in spectral noise across all frequency bands, independent of switching frequency, current level, or load conditions, which aids in the detection of arc faults [50, 78].

Research from Sandia National Laboratories [41] indicates that while there is minimal variation in arcing noise content between the low-frequency ranges (1 - 100 Hz) and the high-frequency ranges (100 kHz to 5 MHz), a clear distinction between arcing noise and

Table 2.1: Time-domain features

Feature	**Mathematical Definition**
Raw Values [50,52,54,60,67]	X
Average [54,55]	$\frac{1}{N}\sum_{i=1}^{N} x_i$
Median [54]	Middle value in a sorted set
Variance [54,55]	$\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2$
Standard Deviation [76]	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}$
Kurtosis [53]	$\frac{\frac{1}{N}\sum_{i=1}^{N}(x_i-\overline{x})^4}{\left(\frac{1}{N}\sum_{i=1}^{N}(x_i-\overline{x})^2\right)^2}$
RMS [54]	$\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$
Peak-to-Peak value [54,55]	Max - Min
Moving Average [56]	$\frac{1}{W}\sum_{i=t-W+1}^{t} x_i$

background noise is noticeable within the mid-frequency range of 100 Hz to 100 kHz. Specifically, the spectrum energy in the range of 1-100 kHz undergoes significant changes during arc faults [57]. This was also proven by Telford et al. [56], who stated a 25 dB increase in the 10 kHz spectrum during their experiments and in which showed a significant increase in 30 - 100 kHz, [78] with 5 to 40 kHz, [76] for lower frequencies like 10 kHz and [79] for 20 kHz to 100 kHz.

Arc faults progress through three distinct phases [50]:

1. In the initial phase, there is a sharp fluctuation in the arc current signal.

2. The second phase shows reduced fluctuations but includes occasional spikes.

3. In the final phase, the arc stabilizes with fewer fluctuations and spikes in the arc current signal.

For the transient state especially in the frequency band of 10 - 50 kHz the frequency is increased, so the Wavelet transformation (and the STFT) can be helpful to concentrate on the right time instead of mixing up the stable and the transient time by observing the whole time at once for a fixed window [57].

In the frequency domain, noise suppression can be achieved through prefiltering, a topic to be further discussed in the next chapter. Owing to switching frequencies, many methods prioritize dampening lower frequencies, like those around 10 kHz. PWM noise, produced by power electronic devices, exhibits a predictable pattern. This allows for the differentiation of current signals under varying operational conditions by analyzing the power spectrum, as shown in studies like [30]. Furthermore, several researchers [16,41,49,80] describe arc faults as pink noise, also known as 1/f noise or flicker noise. This type of noise signal maintains equal energy per octave or equal power across frequency intervals. Mathematically, it is defined by its power spectral density (PSD) function, following the formula:

$$PSD(f) = \frac{1}{f^{\alpha}} \tag{2.4}$$

This implies that monitoring lower frequencies can be particularly insightful.

In the high-frequency band, the amplitude of the arc current is notably small and can easily be obscured by other noise signals. Thus, it is advisable to set the upper frequency limit at 100 kHz [55]. The magnitude of components in the 5 to 40 kHz range tends to increase after arcing, particularly at higher switching frequencies. Also, lower currents tend to result in greater frequency increases within this same range post-arcing [78].

In the study "Frequency-Domain Characteristics of Series DC Arcs in Photovoltaic Systems with Voltage Source Inverters" [78], it was observed that under constant voltage and gap distance, high-frequency variations diminish at higher load current levels. Similarly, higher source voltages tend to stabilize arcs, resulting in fewer variations at these elevated voltage levels.

To demonstrate the differences in frequency spectra between arc and normal conditions in PV arc faults, FFT spectra of random samples from 3 A, 5 A, and 8 A loads were analyzed, as illustrated in Figure 2.1. Most samples showed clear differences in their frequency spectra within the 0 to 40 kHz range. However, at the higher frequency end, some overlap among samples became evident, attributable to various noise conditions in the system. It's important to recognize that because of these overlapping regions, an arc sample might closely resemble a normal sample. As depicted in the figure, a normal sample from the 5A load largely coincides with an arcing sample from the 8A load. As a result, solely depending on a manually adjusted FFT spectrum threshold for arc fault classification could lead to poor accuracy and inadvertent false triggers.

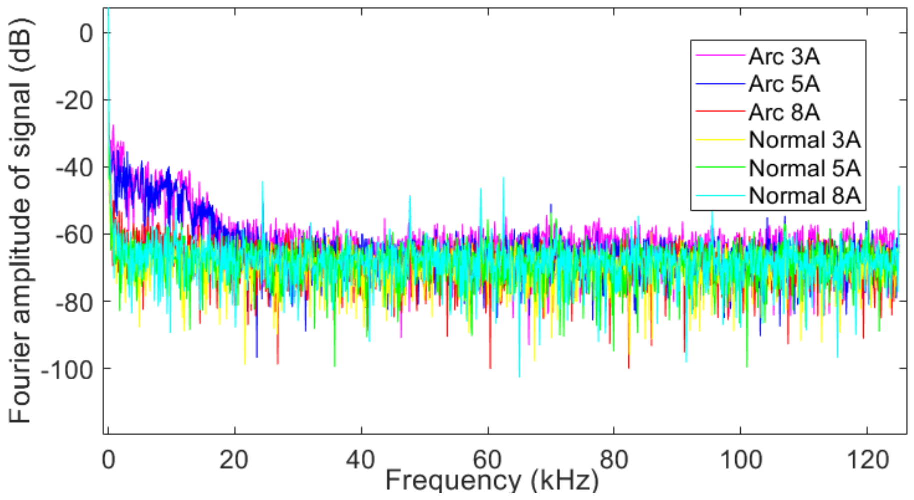


Figure 2.1: FFT spectrum of arc and normal signals at different current levels.

Arcing-induced high-frequency variations tend to increase as source voltage decreases, load current diminishes, and the gap distance between electrodes widens [30]. This pattern implies that in low-voltage or low-current systems, the impact of the arc is more pronounced and potentially more consistent, making detection more straightforward. In contrast, high-voltage systems pose greater challenges for arc fault detection due to the less stable nature of arc effects.

Additionally, high-frequency noise similar to that caused by arcing is generated when a photovoltaic (PV) inverter operates in the Pulse Width Modulation (PWM) state. In such cases, the amplitude of the frequency spectrum of this noise can match or even exceed that of the arc signal. This overlap makes it challenging to differentiate between normal operational states and arc faults based solely on amplitude differences.

The Fast Fourier Transformation (FFT) is the most widely used method in the frequency domain for various applications, including PV arc fault detection [20, 24, 81–86]. FFT transforms a time-domain signal into its frequency-domain counterpart, facilitating the identification of frequency components that may be obscured in the time-domain. The FFT of a discrete sequence of N complex numbers $x[n]$ (for $n = 0, 1, \ldots, N-1$) is computed using the equation:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi kn/N} \tag{2.5}$$

This efficient computation, with a complexity of $O(N \log N)$, makes FFT a valuable tool. However, FFT does have limitations, particularly in capturing transient signals and in frequency resolution which can be affected by the sampling rate. Manual threshold adjustments in FFT-based methods can also be less effective due to noise or changes in the system. Despite these limitations, FFT remains beneficial for PV arc fault detection, especially when combined with other signal processing techniques. A notable drawback of FFT is its inability to convey time information, which is essential in some diagnostic contexts.

2.3.3 Short-Time Fourier Transform

Series arc faults (SAFs) in PV systems exhibit distinctive characteristics in both the time and frequency domains. The Short-Time Fourier Transform (STFT) is an effective tool for analyzing these characteristics, as it provides a time-frequency representation of a continuous-time signal $x(t)$. STFT works by dividing the signal into overlapping frames and then applying the Discrete Fourier Transform (DFT) to each frame. The STFT of $x(t)$ is mathematically defined as:

$$X(t, f) = \int_{-\infty}^{\infty} x(\tau) \cdot w(t - \tau) \cdot e^{-j2\pi f\tau}, d\tau \tag{2.6}$$

In this equation, $w(t)$ is a window function, t denotes time, f denotes frequency, and $X(t, f)$ represents the complex magnitude or phase at time t and frequency f. This combination of time and frequency information distinctly highlights the characteristics of an arc, notably the increase in power across all frequencies during an arc event. Research has shown that a 2D CNN effectively isolates inverter switching noise in the high-frequency domain, primarily utilizing frequency information below 20 kHz [59]. Additionally, repeating time dependencies have been observed in power spectra, potentially influenced by experimental setups [87].

However, utilizing time information in STFT comes at the cost of reduced frequency information, and vice versa, due to the uncertainty principle. This principle is expressed as:

$$\Delta x \cdot \Delta p \geq \frac{1}{4\pi} \tag{2.7}$$

Here, Δx represents the uncertainty in position, and Δp represents the uncertainty in momentum. This trade-off means that arbitrarily high accuracy cannot be simultaneously achieved in both time and frequency representations of a signal.

Another consideration is the window size in STFT, which must be carefully chosen to balance time resolution and frequency resolution. While the signal within the window is assumed to be time-stationary, arc signals are typically nonstationary. Therefore, a smaller observation window is preferable to maintain the assumption of a stationary signal, though this can lead to poor spectral resolution in STFT. Consequently, the selection of the window size is another crucial parameter in STFT.

Studies have demonstrated that using a Hamming window in STFT provides a clearer distinction between normal conditions and fault detection compared to real-time Fast Fourier Transform (FFT) methods [59].

2.3.4 Wavelet Transformation

The Wavelet Transform is an effective mathematical tool for decomposing a signal into a series of basis functions, known as wavelets. This transformation is particularly useful for its computational efficiency and simplicity, making it well-suited for analyzing large data sets. In the realm of signal processing, two primary variations are employed: the Continuous Wavelet Transform (CWT) and the Discrete Wavelet Transform (DWT).

The formula for the Wavelet Transform of a signal $x(t)$ is defined as follows:

$$W(a,b) = \int_{-\infty}^{\infty} x(t) \cdot \psi^* \left(\frac{t-b}{a} \right), dt \tag{2.8}$$

In this equation, a and b represent the scaling and translation parameters, respectively, while $\psi(t)$ is the chosen wavelet function. Notably, wavelets provide improved temporal resolution for higher frequency analysis, which is particularly advantageous for signals with irregular changes, such as non-stationary arc fault signals [88].

Wavelet Transformation is thus particularly apt for handling non-stationary signals.

Researchers like Lu et al. [57] have emphasized the sensitivity of wavelet functions to irregular signals and the concentration of energy in signal compression and reconstruction as reasons for choosing specific wavelets, such as the Daubechies series with variable branch length. However, it's important to note that when applied to inverter-based signals, certain detection limitations exist [89].

In the field of signal energy computation, the wavelet transformation is frequently utilized, relying primarily on two parameters: the choice of the mother wavelet and the scale parameter, which dictates the frequency range of focus in the signal. Selection of the mother wavelet is often driven by the need to detect specific types of discontinuities. For arcing faults detection, many researchers prefer the Daubechies family of wavelets due to their beneficial properties [14].

For instance, a real-time arc fault detection method in PV systems employing 1-level DWT has been proposed, using the ratio of the average power of the first level DWT coefficients (50-100 kHz) to the reference average power [90]. Furthermore, a 4-level DWT is utilized to extract energy information and the modulus maximum of the 126-250 kHz frequency band for detecting parallel arc faults in PV systems [91].

Wang et al. [92] analyzed the spectrum of DC series arc current using DWT and FFT. Their findings indicated that DWT could effectively capture the spectrum's temporal changes, a task at which FFT fails, especially when arc durations are significantly short compared to the sampling window. Moreover, Sung et al. [32] pointed out the limitations of FFT and used STFT data for training their network, allowing for the retention of temporal information. Their methodology involved analyzing the data via STFT and implementing a 1D CNN module with dropout on the raw current data.

It has been stated that Wavelet Transformation generally yields higher accuracy compared to STFT, while also being more computationally efficient [59]. However, it can suffer from issues like oversampling and redundancy, leading to increased memory usage and processing time. Sung et al. [32] addressed this by employing the Mallat algorithm [93], which aims to reduce computational complexity.

The Mallat algorithm's core principle is to separate the input data into low-frequency and high-frequency components, each representing different levels of detail or resolution. Traditional wavelet transform methods calculate all wavelet coefficients at every decomposition level, resulting in O(N) complexity for a 1D signal of length N. Conversely, the Mallat algorithm's filtering and downsampling steps in each iteration reduce the data size, leading to a complexity of O(N log N). However, this downsampling process can cause information loss and frequency aliasing between components [60]. The computational complexities of different algorithms are provided in Table 2.2.

In the context of PV arc fault detection, researchers have explored both DWT and Wavelet Packet Transform (WPT) to extract relevant features from arc fault signals [90, 90, 94–96]. DWT is preferred for its computational efficiency, while WPT is chosen for its higher frequency resolution. Each approach, however, has its own set of advantages and limitations.

Table 2.2: Computational Complexities for 1D Signal with length N and a window length of size M (if used) of the different transformations and algorithms

Transform/Algorithm	**Complexity**
Fourier Transform (FT)	$O(N^2)$
Fast Fourier Transform (FFT)	$O(N \log N)$
Short-Time Fourier Transform (STFT)	$O(NM \log M)$
Wavelet Transform (WT)	$O(N)$
Mallat Algorithm (DWT)	$O(N \log N)$

2.3.5 Other Methods

The wavelet packet transformation extends the capabilities of the wavelet transform, providing a more intricate analysis of a signal's frequency content. It decomposes both the detailed coefficients (from high-pass filters) and approximation coefficients (from low-pass filters), yielding a comprehensive set of new coefficients.

Mathematically, the wavelet packet transformation for a discrete-time signal $x(n)$ involves recursively applying wavelet decomposition to both detailed and approximation coefficients at each decomposition level. Initially, at level $j = 0$, the approximation coefficients A_0 represent the original signal $x(n)$. At the next level, $j = 1$, wavelet decomposition is applied to D_0 and A_0, resulting in new detailed coefficients D_1 and approximation coefficients A_1. This recursive process creates a tree-like structure of coefficients, offering both low and high-frequency insights. However, to ensure effective analysis, it is crucial to maintain maximum flat pass-band characteristics and good frequency separation. Wavelet functions with more coefficients tend to exhibit less distortion than those with fewer coefficients.

The wavelet packet transformation can be represented by the following equations: At each level j, the wavelet decomposition is given by:

$$D_j = W_j \cdot D_{j-1}, \tag{2.9}$$

$$A_j = V_j \cdot D_{j-1}, \tag{2.10}$$

where D_j represents the detailed coefficients at level j, and D_{j-1} represents the detailed or approximation coefficients at level $j - 1$. Similarly, A_j represents the approximation coefficients at level j, and D_{j-1} represents the detailed or approximation coefficients at level $j - 1$.

This transformation enables a more detailed frequency analysis, allowing access to a broader range of coefficients representing various frequency bands. It's especially beneficial in applications requiring an in-depth understanding of a signal's frequency content.

The STFT and wavelet transform, employed for analyzing non-stationary signals, operate under the assumption that the frequency content of a signal changes minimally within a short time interval. The window width in STFT, determined by the signal's temporal variability, influences the frequency resolution.

The Wigner-Ville distribution offers a joint time-frequency analysis of a signal, providing a two-dimensional representation of the signal's energy distribution across both domains. Mathematically, it's defined as the time-frequency representation of the signal's autocorrelation function, computed by Fourier transforming the time-varying autocorrelation function:

The Wigner-Ville distribution can be expressed as follows:

$$W(t, \omega) = \int_{-\infty}^{\infty} x(t + \tau/2) x^*(t - \tau/2) e^{-j\omega\tau} d\tau \tag{2.11}$$

where $W(t, \omega)$ represents the Wigner-Ville distribution of the signal $x(t)$ at time t and frequency ω, $x^*(t)$ denotes the complex conjugate of $x(t)$, and the integration is performed over all time delays τ.

The resulting Wigner-Ville distribution $W(t, \omega)$ provides information about the time-varying frequency content of the signal. The magnitude of $W(t, \omega)$ represents the energy or power of the signal at a specific time t and frequency ω. The phase of $W(t, \omega)$ provides information about the phase relationship between different frequency components of the signal at a particular time. It is important to note that the Wigner-Ville distribution has some limitations. One of the main issues is that it can exhibit interference patterns known as cross-terms, which can make the interpretation of the distribution challenging. These cross-terms arise due to the quadratic nature of the distribution and can introduce spurious components in the time-frequency representation. Additionally, it has a higher computational cost than all other methods

Empirical Mode Decomposition (EMD) is a signal processing technique that decomposes a given signal into a set of intrinsic mode functions (IMFs). It was proposed by Norden E. Huang and his colleagues in the 1990s as a method to adaptively analyze non-stationary and nonlinear signals. EMD is a data-driven decomposition technique that does not rely on predefined basis functions or assumptions about the signal's characteristics. It aims to decompose the signal into oscillatory components with well-defined instantaneous frequencies. Each IMF represents a narrowband component with a well-behaved oscillatory behavior.

Mathematically, the EMD process can be expressed as follows:

$$x(t) = \sum_{k=1}^{N} c_k(t) + r(t)$$

where $x(t)$ represents the original signal, $c_k(t)$ denotes the k-th intrinsic mode function (IMF), $r(t)$ is the residue signal, and N represents the total number of IMFs obtained.

Each IMF $c_k(t)$ satisfies the IMF criterion and captures a specific oscillatory component in the signal. These IMFs are sorted in descending order of their instantaneous frequencies, with the first IMF representing the component with the highest frequency. Empirical Mode Decomposition (EMD) is renowned for its versatility and has been widely applied in various fields, such as biomedical signal analysis, environmental signal processing, and financial time series analysis. EMD's data-adaptive nature enables it to extract meaningful components from complex, non-stationary signals, enhancing both the analysis and interpretation of these signals.

However, it's important to recognize that EMD has certain limitations, including mode mixing and endpoint effects, which can impact the accuracy of the decomposition. To address these issues, various modifications and extensions have been developed, such as Ensemble EMD (EEMD) and Complete EEMD (CEEMD), which aim to improve the efficacy of EMD-based signal analysis. Overall, Empirical Mode Decomposition is a powerful analytical tool for dissecting non-stationary signals into their intrinsic components, offering valuable insights into their underlying oscillatory behavior.

The voltage or current information alone can be effectively utilized to detect arc faults. Specifically, a variation in the voltage of the DC optimizer can serve as an indicator of a PV arc fault [97]. Typically, the presence of a series arc fault, which involves the insertion of a series impedance, results in a decrease in the terminal voltage. However, it's important to note that relying solely on voltage measurements for fault detection can sometimes lead to false positives. This is because the output voltage of PV panels can fluctuate due to a variety of factors, including high temperatures, accumulation of dust and dirt, partial shading, issues with the inverter, or failures in other components.

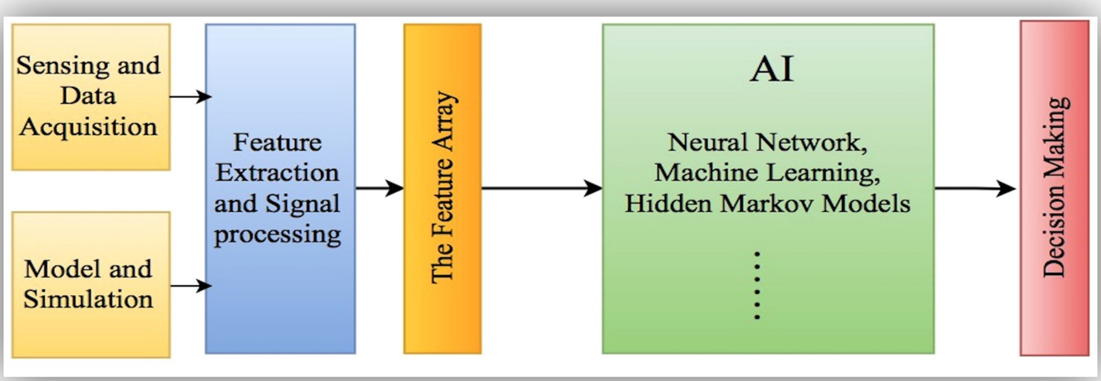

Sensing and Data Acquisition
Model and Simulation
Feature Extraction and Signal processing
The Feature Array
AI
Neural Network, Machine Learning, Hidden Markov Models
Decision Making

Figure 2.2: Generic AI-based method for arc fault detection [16].

2.4 Artificial Intelligence and Machine Learning-based Methods

In general AI-based methods acquire some data, extracts features from the input data and process using AI algorithm and finally gives a decision. A generic AI-based arc fault detection algorithm is provided in Fig. 2.2.

2.4.1 Fuzzy Logic

Fuzzy Logic (FL) is an AI-based method for PV arc fault detection algorithm [76, 98]. The FL-based method uses linguistic variables to describe the system status, and these variables are then mapped onto fuzzy sets. The fuzzy sets are then processed using fuzzy rules to classify the system state as either normal or faulty. These methods have high accuracy and does not require a large amount of training data. The response time is slower than the ANN-based method. Also, the hand adjusted threshold makes it difficult to adjust to different scenario and resulting in false tripping. A cascaded Fuzzy logic based algorithm is presented in [76] where the inputs are frequency analysis, peak detection and operating point. The detection logic is split into 4 sub-detectors (peak evaluator, window near, window wide and power analyzer) followed by a Master Fuzzy Arc Fault Detector (MFAFD). The sub-detectors analyze peaks, signal energy change, and power change, and their outputs are fed to the MFAFD. The MFAFD output delivers a number between 0 and 1 and is a mass for the arc fault probability. A threshold has to be defined to detect an arc fault. Similarly, [52] used different WT features, CT and Voltage features to fuzzy it into a single scalar classified by a rule based engine. [53]. Wang et al. [53] used for this classification task a support vector machine. This algorithm is cost-effective and can be implemented directly in the inverter's controller. It is more complex to implement as the rules of different sub-detectors and input variables have to be defined. Another major drawback is that FL method is based on hand adjusted threshold, which may need to be adjusted based on the specific application and operating conditions. The algorithm is very sensitive and needs clear value to do a precise description. [52] stated, that as increasing fault impedance increases, the fault detection becomes more challenging since the impact of shorted panels on the whole system is minimal.

2.4.2 Support Vector Machine

Support vector machine (SVM) is a popular AI-based method for classification tasks, including PV AFD [62, 99–102]. The SVM-based method uses a decision boundary to separate the normal and faulty states. The SVM optimization problem for binary classification can be formulated as follows:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i - b)) \tag{2.12}$$

subject to the constraint: $y_i(\mathbf{w}\cdot\mathbf{x}_i - b) \geq 1, \quad \forall i = 1, \ldots, n$. The objective is to find the optimal hyperplane represented by $\mathbf{w} \cdot \mathbf{x} - b = 0$ that separates the two classes with the maximum margin, where $\mathbf{w}$ is the weight vector and b is the bias term. Here, C is a hyperparameter that controls the trade-off between maximizing the margin and minimizing the classification error. The first term in the objective function represents the margin maximization, and the second term is the hinge loss function that penalizes misclassifications. The solution to this problem can be computed using Lagrange multipliers [103]. By selecting an appropriate kernel, the power of SVMs can be leveraged to find non-linear decision boundaries and handle complex patterns in the data. The SVM-based method has high accuracy and can handle nonlinear relationships in the data. It requires a large amount of training data to develop a reliable model. In combination with particle swarm optimization algorithm, SVM model is used in [64] for DC arc fault detection.

2.4.3 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a simple, yet effective, machine learning algorithm used for classification and regression tasks. KNN operates on the principle of similarity, where the class or value of a data point is determined by the classes or values of its nearest neighbors in the feature space. In classification, a data point is assigned to the class most common among its k nearest neighbors, where k is a user-defined constant. For regression, the output is the average of the values of its k nearest neighbors. KNN is a non-parametric, instance-based learning algorithm, meaning it makes predictions based on the memorized training data without assuming an underlying distribution. Despite its simplicity, KNN can achieve competitive performance in various applications, provided that the data is properly normalized and the choice of k is appropriate. Given a training dataset $D = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, where x_i are feature vectors and y_i are class labels the k-nearest neighbors algorithm assigns a class label or predicts a target value for an unlabeled data point x_0 as follows:

1. Find the k training examples in D that are closest to x_0 based on a distance metric.

2. 2. If $k = 1$, assign the class label or target value of the closest neighbor to x_0.

3. 3. If $k \geq 1$, assign the class label or predict the target value of x_0

based on a majority vote or average of the k nearest neighbors' labels or values.

Reference [99] used 250 kHz sampling rate and 2 ms sample length of the input current to analyze and test the DC arc fault using several machines learning based methods including LSTM, DNN, SVM, DT, KNN, gated recurrent unit (GRU) as an RNN method, RF, Naive Bayes (NB). The models are tested at different switching frequencies and current levels of the loads. It is found that different models works better for different combination of switching frequency and current levels. Reference [104] demonstrates the integration of load classification with the KNN algorithm, complemented by the use of event-based methodologies and sequential forward floating selection techniques.

2.4.4 Artificial Neural Network

A fully connected network, also known as a dense network or a multi-layer perceptron, is a type of artificial neural network (ANN) in which each neuron is connected to every neuron in the adjacent layers and is responsible for transforming the high-level features into the final output. Let X be the input vector to the fully connected network, and let W represent the weight matrix connecting the input to the first hidden layer. Each neuron in the hidden layer receives the weighted sum of the inputs and applies an activation function f to produce the output. The output of the first hidden layer can be computed as follows:

$$H_1 = f(X \cdot W + b) \tag{2.13}$$

where H_1 is the output of the first hidden layer, b denotes the bias vector, and f is the activation function.

One of the early attempts to utilize neural networks for arc fault detection was conducted by Momoh et al. in 2003 [86]. In this study, the authors computed spectral energies and used them to train a neural network for fault identification. The anti-aliased digital voltage signal, with a fixed window length of 10ms, was passed into a DSP module to calculate the Fast Fourier Transform (FFT). Subsequently, ten components of the spectral energy were fed into a fully-connected neural network to estimate the fault probability. By employing prefiltering, power spectrum analysis, and applying an ANN to the output, Momoh et al. took the first steps towards a high error-aware detection. No specific accuracy was determined in the study. Ma et al. [105] developed a comparable network based on wavelet decomposition at level 4. Their network architecture consisted of a 5-6-6-1 fully connected network with two hidden layers, where the numbers denote the respective number of neurons in each layer. Their research demonstrated that even with just five input neurons, the network achieved an accuracy of over 95%, surpassing traditional approaches. However, the examined cases were closely clustered. To overcome this,

J. Siegel et. al [24] used a 3-16-32-16-1 fully connected network with 4 different loads by using the Discrete Fourier Transform, the spectral signature of the current (Mel Frequency Cepstral Coefficients MFCC) and the Discrete Wavelet Transform as input and five types of load as data. The model failed to detect unseen device configurations by testing the previously trained model on loads in parallel. Han et al. [25] conducted three 2-5-1 three-layer artificial neural network with prejudging the categories on the waveform based on the wavelet transformation. The loads were divided into resistive, resistive-inductive, capacitive category. Wang et al. [20] used a similar way with prejudging before applying a specific BP network with resistive category, capacitive-inductive category, and switching category, but in difference with prejudging based on the FFT. Although both papers achieved a 100 % accuracy in categorizing the faults, it is important to note that training three networks and the associated processing costs of pretraining and a fully connected network are quite high. The major drawback lies in the fact that the features used in the network are handcrafted, and the overall quality of the network heavily relies on the specific choices made during feature selection. To overcome this, a convolution neuronal network can be used to extract certain features.

2.4.5 Hidden Markov Model

Hidden Markov Model (HMM) is a statistical model widely used for modeling sequential data, where the underlying system is assumed to be a Markov process with hidden states original designed for speech models.
They consist of a set of hidden states: $S = S_1, S_2, \ldots, S_N$ and a set of observed emissions: $O = O_1, O_2, \ldots, O_T$. The probabilities are defined as:
Transition probabilities: $A = [a_{ij}], \quad 1 \leq i, j \leq N$
Emission probabilities: $B = [b_{ik}], \quad 1 \leq i \leq N, \quad 1 \leq k \leq M$
Initial state probabilities: $\pi = [\pi_i], \quad 1 \leq i \leq N$
The classification can be done by the viterbi algorithm by backtracking. The computational complexity of the Viterbi algorithm is $O(N \cdot K)$ for N states and a sequence of length K. HMMs can be used in classification problems associated with noisy time-series data even though they do not have exact domain knowledge of the problem. HMMs assume that the system modeled is a Markov process with unobserved (hidden) states and that system data is a noisy observation of this process [56]. HMM are well-designed for time series. It shows good accuracy of 98.3% in [56]. However, only limited data is available (60 Tests in 1 study) and its training is based on Maximum a posteriori estimation [106].

2.4.6 Convolution Neural Network

Convolutional neural network (CNN) is an AI-based method that has been widely used in image classification tasks. A convolutional layer applies a set of filters, denoted by $\mathbf{W}$, to the input $\mathbf{X}$ using the convolution operation denoted by $(*)$ [107]. Each filter is a set of learnable parameters that are optimized during the training process. The output feature maps, denoted as $\mathbf{Y}$, are computed as follows:

$$\mathbf{Y}_{i,j,k} = \sum_{m,n,l} \mathbf{X}_{i+m,j+n,l} * \mathbf{W}_{m,n,l,k} \tag{2.14}$$

where $\mathbf{Y}_{i,j,k}$ is the value of the feature map at position (i, j) and channel k, $\mathbf{X}_{i+m,j+n,l}$ represents the input value at position $(i+m, j+n)$ and channel l, and $\mathbf{W}_{m,n,l,k}$ denotes the weight of the filter at position (m, n) and channel l for the output channel k.

The pooling layers reduce the dimensionality of the feature maps by downsampling the data. Common pooling operations include max pooling or average pooling. Let $\mathbf{P}$ denote the pooling operation, and $\mathbf{Z}$ be the output of the pooling layer. The pooling operation can be defined as:

$$\mathbf{Z}_{i,j,k} = \mathbf{P}(\mathbf{Y}_{i,j,k}) \tag{2.15}$$

Finally, the fully connected layers consolidate the high-level features extracted by the convolutional and pooling layers to generate the final predictions.

Schweitzer et al. [48] and Zhang et al. [108] utilized a 2D CNN architecture based on the LeNet5 model, employing frequency analysis to convert the data into 2D images in the frequency domain. Schweitzer et al. employed the Short-Time Fourier Transform (STFT) to calculate the DC spectrogram, which was then fed into the network. Their network consisted of an average pooling layer as the first layer, followed by 2 convolutional layers and 2 fully connected layers. Zhang et al. mapped the wavelet coefficients to a 2D matrix and interpreted it as HSV color index. To make the network more stable, he used image occlusion to make a new dataset. Omran et al. [33] and Wang et al. [71] adopted a similar approach by mapping the raw data to a 2D matrix, treating it as an image. The convolutional network was responsible for automatically extracting relevant features from the data, followed by a fully connected network for classification. S. Lu [89] adapted a 2D feature map based on the normalized wavelet packet entropy with overlapping signal windows. Compared different settings and suggested a 3 Max Poling layer with ReLU activation function followed by 8-5-1 Fully connection. Thus, the optimal filter size for DC arc fault detection is proposed as 5x5. [33] showed that this approach has a high accuracy (98% in the simulation). He used 3 convolution layers, the first and the last one with a 5x5 filter, each followed by a 2x2 pooling layer to extract the features, followed by a 4 fully connected layer. The main drawback is that the network was only tested on simulated DC data. Qi et al. [109] achieved high accuracy in their research, as reported in their paper. They used a stacked CNN-layer network with a fully connected implementation and demonstrated that a three CNN-layer network followed by a fully connected network is enough to achieve a 99.908 % accuracy, but when the data is clean and provided by a simulator. Li et al. [59] showed that instead of using short time Fourier transform (STFT), the use of DWT as the input to the CNN model works better for arc fault detection.

There are certain drawbacks to consider. The 2D CNN architecture tends to have a high number of parameters, leading to increased computational complexity. Moreover, the effectiveness of the approach heavily depends on the method used to transform the data into a 2D image, introducing additional preprocessing steps and computational costs. Additionally, it is important to note that the data being processed is not an actual image but rather a representation of the input data in a 2D format. Mapping the raw data to a 2D matrix requires preprocessing steps, which can introduce additional computational cost. The process of converting the data to an image format may result in a loss of information or a change in the data representation. If deep CNN is used, it is easy to fall into overfitting, resulting in poor effect. On the contrary, if only shallow CNN is used, it cannot obtain good results due to insufficient fault feature extraction [55]. Even there is a lacked of evaluation for DC systems, 1D CNNS shows an excellent feature extraction and classification ability tested on AC data even with only a few layers. Wang et al. proposed a novel CNN-based algorithm called ArcNet [11], which directly applies a 1D- CNN to the raw current data. By adapting the AlexNet network, ArcNet is able to categorize normal and abnormal load conditions using eight different types of generalized loads, with 31 ms of examination time per test sample at a 10 kHz sampling rate, making it feasible for real-time arc fault detection applications. four convolution layers each followed by a MaxPooling layers and three FC layers were used. Moreover, the design was refined to create Efficient-ArcNet [18], which utilizes EffNet building blocks to achieve an arc fault detection accuracy of 99.36 % and reduces the runtime to 1.17 ms. Each block in Efficient-ArcNet includes a 1D pointwise convolution using Ch/2 filters, followed by a 1D depthwise convolution with a filter size of 5x1 and Ch/2 filters. Finally, a Max Pooling layer of size 2x1 is applied. After the third block, the data is flattened and three fully connected layers with 64, 32, and 8 neurons are added. The last layer serves as the output layer, with 8 possible outcomes. This refinement makes it possible to run the algorithm in real-time on a Raspberry Pi 4B. Shan et al. [63] composed the signal into eigenmode function signals and calculated the Spearman correlation coefficient . A study [110] shows that this approach fails when it comes to unknown combination of known single loads.

2.4.7 Attention Algorithm

The attention [111] method is a mechanism used in deep learning models to focus on specific parts or features of the input data. It assigns different weights or importance values to different elements of the input, allowing the model to selectively attend to relevant information.
The attention mechanism can be defined mathematically as follows: Given an input sequence $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ and a target element $\mathbf{Y} = (y_1, y_2, \ldots, y_m)$, the attention mechanism computes the attention weights $\mathbf{A} = (a_1, a_2, \ldots, a_n)$, where a_i represents the weight assigned to the i-th element of $\mathbf{X}$.

The attention weights are typically calculated using a scoring function, which measures the similarity or compatibility between the target element and each element in the input sequence. One common scoring function is the dot product:

$$\text{score}(x_i, y_j) = \mathbf{W}_a \cdot (x_i \odot y_j) \tag{2.16}$$

where $\odot$ denotes element-wise multiplication and $\mathbf{W}_a$ is a learnable weight matrix. The attention weights are obtained by applying the softmax function to the scores:

$$a_i = \frac{\exp(\text{score}(x_i, y_j))}{\sum_{i=1}^{n} \exp(\text{score}(x_i, y_j))} \tag{2.17}$$

Finally, the context vector $\mathbf{C}$ is computed as the weighted sum of the input elements:

$$\mathbf{C} = \sum_{i=1}^{n} a_i \cdot x_i \tag{2.18}$$

Although Chabert et al. [112] utilize the raw current data, Wang et al. focus on the power spectrum for arc fault detection. It is worth noting that in the case of power spectrum analysis, selecting the appropriate frequencies can be advantageous without the need for extensive preprocessing. When employing an attention mechanism, there is a trade-off between losing a priori knowledge of the frequency band with the most significant changes and increasing computational effort. An important aspect to highlight is that Wang et al. estimated the power spectrum through the utilization of an autoregressive model, leading to a decrease in computational effort. It is crucial to emphasize that the selection of optimal parameters for the model is of utmost importance and must be approached with careful consideration.

2.4.8 Other CNN Techniques

One of the newer publications, Yan et al. [51] from 2023, proposed a method with an high accuracy of 99.88 %, called ArcNN. The network is applied to the raw current data and is based on dilated convolution, which avoids the Max pooling Layer and allows every neuron to have a higher receptive field. For an input time series sequence $X = x_0, x_1, ..., x_t$

and a filter $f()$, the dilated convolution is defined as:

$$F(t) = (X * f)(t) = \sum_{n=0}^{k-1} f(i) \cdot X_{t-d \cdot i} \tag{2.19}$$

where **d** is the dilation factor, **k** represents the filter size.

This allows neurons to have a wider spread of historical data and eliminates the need for pooling layers. To avoid neural network degradation, residual blocks were added The residual block can be defined as follows:

$$\mathbf{y} = F(\mathbf{x}, \mathbf{W}) + \mathbf{x}, \tag{2.20}$$

where **x** is the input to the block, F is a function that transforms the input, **W** are the learnable parameters of the function, and **y** is the output of the block. The addition of **x** to the output of $\mathcal{F}(\mathbf{x}, \mathbf{W})$ creates a residual connection that allows gradients to flow through the network more easily.

Principal component analysis (PCA) was applied to reduce the input dimensionality. The resulting data was then fed to a single hidden layer network for classification. To optimize the network, multiple layers were fused and quantized to 8-bit integers, resulting in an overall size of 127 kb and a total of 15875 parameters.

The system's major drawback is the expensive Jetson Nano board, which is not applicable in industry, and the long communication latency. Therefore, the authors recommend implementing the system using an FPGA. Also, shallow CNN networks showing good results and don't have the need of residual blocks which cost more computational effort.

2.4.9 Ensemble Machine Learning

Ensemble machine learning (EML) is a method that combines multiple machine learning models (different features or different methods) to improve the accuracy and reliability of predictions [101]. In the context of PV arc fault detection, ensemble methods can be used to combine the predictions of multiple individual machine learning models, such as decision trees or support vector machines, to generate a final prediction that is more accurate than any individual model.

The random forest method is a popular supervised machine learning algorithm used for both classification and regression. Ensemble learning combines multiple individual models to make predictions. The main idea behind random forests is to create an ensemble of decision trees [113], where each tree is built using a different subset of the training data and a random subset of features. This randomness helps to reduce overfitting and increase the model's robustness.

The reference in [54] used ensemble machine learning based algorithm for series DC arc fault detection. The method is validated using experimental data from buck and boost converter constant power loads (CPLs), and a set of time domain features (average, median, variance, RMS, and difference between maximum & minimum value) are extracted from the data to train the machine learning algorithms. The data sampling rate is 80 kS/s and window size is 2 ms long. A two-step algorithm is proposed to recognize the arc fault in different load types. The stacking ensemble algorithm is found to maintain superior performance compared to other EML algorithms. The prototype arc fault detector is realized using low-cost hardware and has achieved remarkable versatility, reliability, and low detection latency time. One of the main limitations of the paper is that the experiments were conducted only on two specific types of constant power loads, and it is unclear how well the proposed method would perform with other types of loads. The detection latency time is also relatively long, ranging from 37 to 69 ms, and this may not be sufficient for certain applications.

Another ensemble method that has been used for PV arc fault detection is the AdaBoost algorithm, which combines multiple weaker classifiers into a stronger classifier. The basic idea behind ensemble methods is that by combining the predictions of multiple models, the limitations of any individual model can be overcome, thereby improving the overall accuracy and robustness of the predictions.

Decision Tree (DT) is an AI-based method that uses a tree-like model of decisions and their possible consequences. The DT-based method uses a set of rules to classify the system state as either normal or faulty. The DT-based method has a fast response time and can handle large datasets. However, it requires a large amount of training data to develop a reliable model [99, 114].

Instead of relying on manually set decision methods, another approach is to process different features individually and then combine their outputs. Liu et al. [70] utilized features from the time domain, frequency domain, and energy entropy domain, each processed by its own network. Remarkably, these networks shared the same structure as the other networks. Chen et al. [49] employed DC Current and FFT decomposition with CNN models and an attention algorithm. In order to capture both local and global features simultaneously, each network utilized an Inception layer. The Inception architecture is designed to effectively capture information at multiple spatial scales by incorporating filters of different sizes within the same layer. It comprises parallel branches with varying filter sizes, allowing for the capture of information at different levels of detail. Similarly, Yu et al. [72] utilized two adapted AlexNets trained in parallel and applied them to the same features. The overall accuracy of the Network was only 92%.

In contrast to this, the ensemble method employed different methods while utilizing the same features. Rather than solely relying on CNNs, a variety of methods were employed.

V. Le employed in his papers [47] [54] [73] ensemble machine learning techniques. The same feature was applied to different models such as SVM, KNN, RF, and ET, and their outputs were fused for the final prediction. This approach achieved a high accuracy of up to 99.9%.

Combining different features and methods can yield benefits due to the high variation in the signal. It is crucial to carefully select features that provide high accuracy and offer orthogonal information. Nevertheless, utilizing multiple features and methods significantly increases computational efforts. Additionally, employing too many features may lead to overfitting and a reduction in accuracy.

One disadvantages of the CNN networks is that are not constructed for time data, only using context of the neighboring data. They are not involving the ordered time information. In contrast, recurrent neuronal networks like LSTM are constructed for time series. By compression past input data and using it for the new processing cycle, it can use information of the past for the actual output.

2.4.10 Recurrent Neural Network

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture that is designed to address the challenges of capturing and retaining long-term dependencies in sequential data [107]. The key idea behind LSTM is the introduction of memory cells and gating mechanisms. These mechanisms allow the LSTM to selectively remember or forget information, which helps in preserving relevant information over long sequences. The LSTM module consists of several components: the input gate, the forget gate, the candidate value, the cell state, and the output gate.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.21}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.22}$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.23}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.24}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.25}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.26}$$

In these equations, x_t represents the input at time step t, h_{t-1} represents the previous hidden state, f_t, i_t, $\tilde{c}_t$, and o_t represent the gates. c_{t-1} represents the previous cell state, and c_t and h_t represent the updated cell state and output at time step t respectively. The σ symbol represents the sigmoid activation function, which squashes the values between 0 and 1. The $\odot$ symbol denotes element-wise multiplication. The LSTM-based method has high accuracy and can handle time-varying data [99]. It requires a large amount of training data to develop a reliable model. X. Liu [87] used in the previous mentioned paper a LSTM layer to proceed the features extracted from the convolutional layer TL-LEDarcNet [32] used the same for a 1D CNN with an classifier network trained without a LSTM. Lu Xing [58]proceeded a LSTM on the STFT Yu et al. [72] used the eigenmatrix from the CNN features to feed it into a LSTM.

Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) architecture which is commonly used in ANN for processing sequential data, such as time series or natural language. The GRU is a variation of the standard RNN that addresses some of the limitations of traditional RNNs, such as the vanishing gradient problem. In [115], GRU has been used for detecting series arc fault in building integrated photovoltaic (BIPV) systems.

2.5 Advantage and Limitations of Different Methods

When comparing the traditional methods and machine learning (ML) approaches for arc fault detection, each comes with their advantages and limitations.

Traditional Methods like time or frequency domain analysis (using tools like Fast Fourier Transform, FFT, and Wavelet Transform, WT) are appreciated for their straightforward implementation in real hardware. However, they may not be well-suited for dynamic systems as they're hard to adapt to new scenarios. They also suffer from a high rate of false positives due to noise interference, and a significant drawback is the need for manual threshold adjustments, which can be subjective and limit their effectiveness.

Conventional Machine Learning Methods such as SVM and Random Forest (RF) offer higher accuracy and can manage a moderate to high runtime. However, like traditional methods, they encounter difficulties when it comes to adapting to new systems. Ensemble ML methods, which combine the predictions of multiple learning algorithms, may enhance results but introduce complexity that could be a challenge to manage in the low-end microcontrollers.

Advanced Machine Learning Methods including ANN and CNN are known for their high accuracy levels. Nonetheless, they come with the cost of high computational complexity and runtime, which might not be practical for implementation in commercial microcontroller units (MCUs) due to limited resources. They also have the risk of missing samples, potentially due large model size and the eventual large inference time due to its complexity of computation.

In essence, while advanced ML methods offer high accuracy, they require significant computational resources, and their complexity can make them less accessible for deployment in hardware-constrained environments. On the other hand, traditional methods, while simpler to deploy, may not perform as well in complex or noisy environments and lack the adaptability that machine learning methods can provide. Conventional ML strikes a balance with respectable accuracy and runtime, though it still faces challenges in adaptation and potentially increased complexity with ensemble techniques.

CHAPTER 3: AC ARC FAULT DETECTION USING CNN-BASED ALGORITHM

In the pursuit of enhancing the electrical safety and reliability of AC systems, the first objective is to develop an AC arc fault detection algorithm utilizing a CNN-based algorithm. Convolution layers can automate the feature extraction from arc fault current therefore the complex data preprocessing can be avoided. The algorithm is designed to have an arc fault detection accuracy of more than 99%. The proposed method is verified using Raspberry Pi 3B platform for arc fault detection.

3.1 Data Collection And Analysis

3.1.1 Arc Fault Experiment Platform

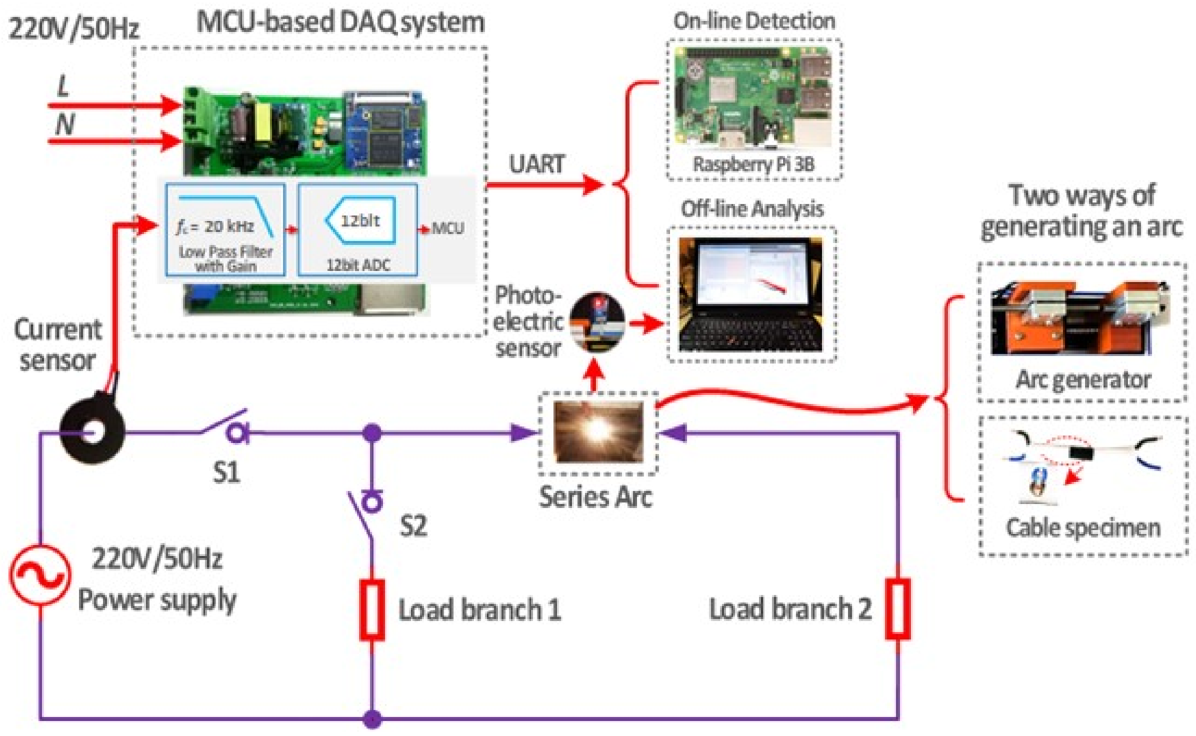


Figure 3.1: Arc Fault Data Collection Process and Circuit Arrangement [11].

Since real arc faults are always random and difficult to capture or reproduce, a specific test platform has to be used to generate arcs with different types of real loads. According to the IEC60606 standard, a series arc fault experiment platform has been designed, as shown in Fig. 3.1. The arcs can be generated in two different ways: one with the arc generator, the other by using a cable specimen [11]. The arc generator can create the real arc that occurs during the loose connection of an electric cable, whereas the arc generated by the cable specimen can simulate an arc caused by insulation breakdown in the electric circuit. Although it is allowed to choose either method as per the standard, both methods have been adopted to closely simulate real scenarios. The input for this setup is a 220 V, 50 Hz power supply. One current transformer (CT) is used to sense currents carried by the loads. Additionally, a well-designed hardware circuit board is used for DAQ rather than an oscilloscope to ensure the DAQ is consistent with a commercial arc fault detection device. The proposed DAQ board is based on the STM32H743 MCU, and the cut-off frequency of the active low-pass filter is 20 kHz with gain. The original sampling rate of the collected data is 83.33 kHz with a resolution of 12 bits. The recorded data are collected on board and stored in an SD card, which can also be transferred to a PC or an online detection device (such as Raspberry Pi) via the UART interface. Furthermore, a photoelectric sensor is integrated to provide a timestamp to the data, aiding in post-processing.

3.1.2 Types of Loads and Visible Arc Features

Since arc fault protection is mainly used in residences, different household appliances are selected as experimental loads. Eight different types of loads are considered for collecting diversified load currents along with arc and normal conditions. The loads include resistive load, capacitor start (air compressor type) motor, vacuum cleaner, fluorescent lamps, electronic light dimmer (thyristor type), electronic switch mode power supply, electric hand tool (drill), and halogen lamps. The specifications of the loads are complied with the IEC62606 standard [116]. Each load has a sub-load category of different brands, for example, vacuum cleaner of Midea and vacuum cleaner of Philips brands. The arc and normal load currents are collected from single load as well as from different loads connected in parallel.

Although it is difficult to recognize most series arc data by visualization, nevertheless, arc data can be visualized and understood in many ways. For example, Fig. 3.2 shows the current waveforms and corresponding frequency spectrum of a resistive load. From the time domain perspective, in the arc current there almost always has a current zero at the ”zero crossings” for a period of time as shown in Fig. 3.2 (a). This is because the arc is extinguished as the voltage across the discharge gap decreases. Then, as the voltage rises, the arc reignites and the current suddenly changes at that moment. From the frequency domain perspective, the high-frequency component of arc current is higher than that of non-arc current as shown in Fig. 3.2 (b). In order to avoid the interference of harmonic currents of normal operating loads, 3 kHz - 12 kHz is set as the frequency bandwidth for arc fault detection, which also allows using a relatively low sampling rate to save the RAM required.

Moreover, whether it is a linear load or a non-linear load, its non-arc current waveform has relative stability within a short observation window (such as 2-3 power cycles). However, the arc current lacks significant stability. As shown in Fig. 3.2 (b), even in adjacent periods, the current waveforms are significantly different, and no stable periods can be observed. This is because the arc discharge itself is a very complicated physical and chemical process, affected by various factors such as loads, electrode materials, and more. Therefore, 2 or 3 cycles of the current signals can be taken as one calculation unit for the algorithm.

The visible features mentioned earlier might be challenging to identify using time-frequency domain approaches. However, AI methods can effectively learn these features. Particularly with a large amount of data, the proposed method can achieve high accuracy.

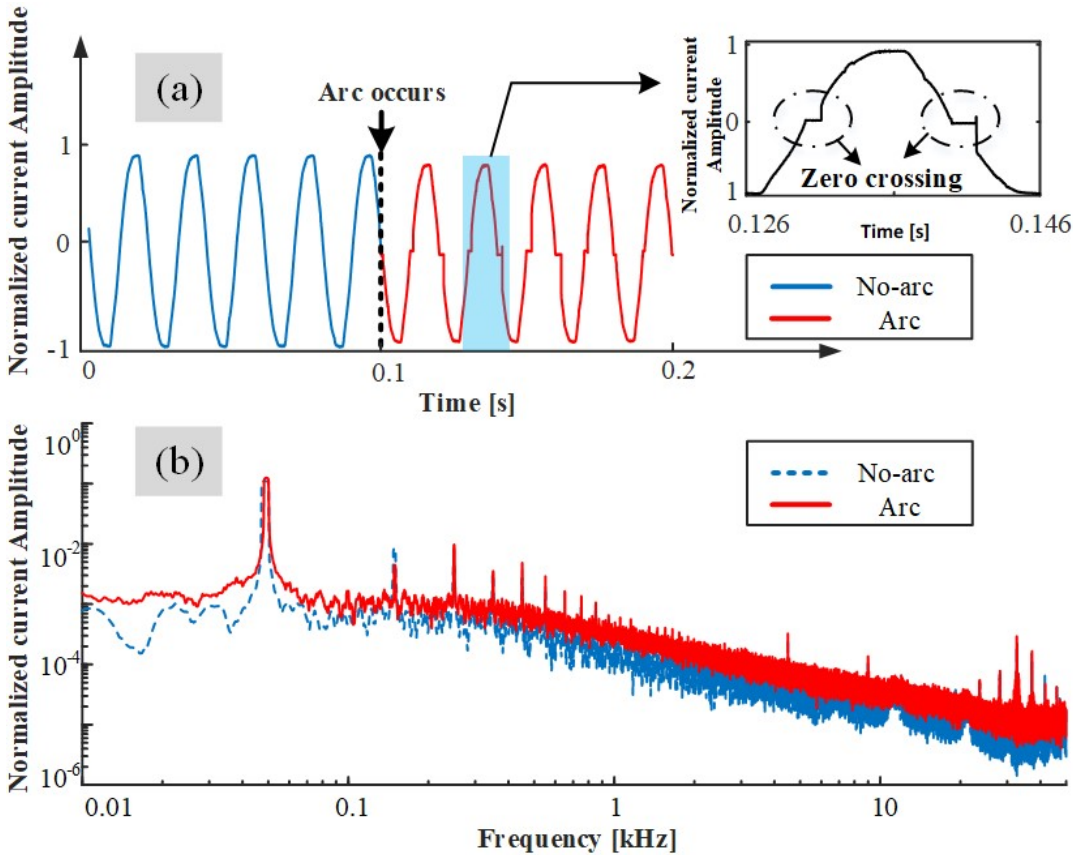


Figure 3.2: Visualization of (a) the arc and the normal current of a resistive load, (b) its corresponding frequency spectrum [11].

3.2 Description of Database

For this research, more than eight different types of loads are grouped into four major classes based on their specific characteristics. All the loads which are resistive types have nearly sinusoidal types of load currents and hence they are kept in the resistive group (RE). Capacitor start motor, electric hand drill, and vacuum cleaner loads classified as the motor (MO) type loads, because they have high inrush current during the starting period. Halogen lamps and fluorescent lamps load currents look like arc when the gas is discharged. Therefore, those two loads are categorized into gas discharge lamps (GDL) load class. Power electronics type of loads such thyristor type electronic lamp dimmer and switch mode power supply loads such as computers have wide bandwidth of harmonics in the load currents. These types of loads are kept in the power electronics-enabled (PE) loads group.

The primary database is created by down-sampling to 10 kHz from the original data. Each sample of data having 1 complete cycle of the current wave is converted into a 1D vector. The dataset is composed of 9623 samples of arcing and 20528 samples of normal load current resulting in a total of 30151 samples. Although the data number is relatively large, they can be automatically labeled with the help of a computer program and the timestamp provided by the photoelectric sensor. All data are labeled in such a way so that the proposed algorithm can not only able to identify the arc and normal load currents but also it can specify the load type (i.e., RE, MO, GDL, PE). The dataset is randomly split in order to have 75% data for training, 10% data for validation and 15% data for testing. The load categories, their corresponding labels, and number of respective training, validation, and test samples are depicted in TABLE 3.1. The primary database is normalized between 0 and 1 within each of 200 sample points. Besides having the primary database of 10 kHz sampling rate, to check the accuracy of arc detection with a different sampling frequency, different databases were created containing same data samples sampled at 40 kHz, 5 kHz, 2.5 kHz, 2 kHz, and 1 kHz respectively. In each of the databases, the training, testing and validation split is kept as per the primary database.

Table 3.1: Overview of the database [11].

Load Group	**Signal Type**	**Labels**	**Training Samples**	**Test Samples**	**Validation Samples**	**Total Samples**
Resistive Loads	Arc	0	3164	632	423	4219
	Normal	1	6342	1268	846	8456
Motor Type Loads	Arc	2	1437	285	196	1918
	Normal	3	3340	667	448	4455
Power Electronics-enabled Loads	Arc	4	2166	433	290	2889
	Normal	5	3390	677	455	4522
Gas Discharge Lamp Loads	Arc	6	447	88	62	597
	Normal	7	2321	463	311	3095
Total number of samples=			22607	4513	3031	30151

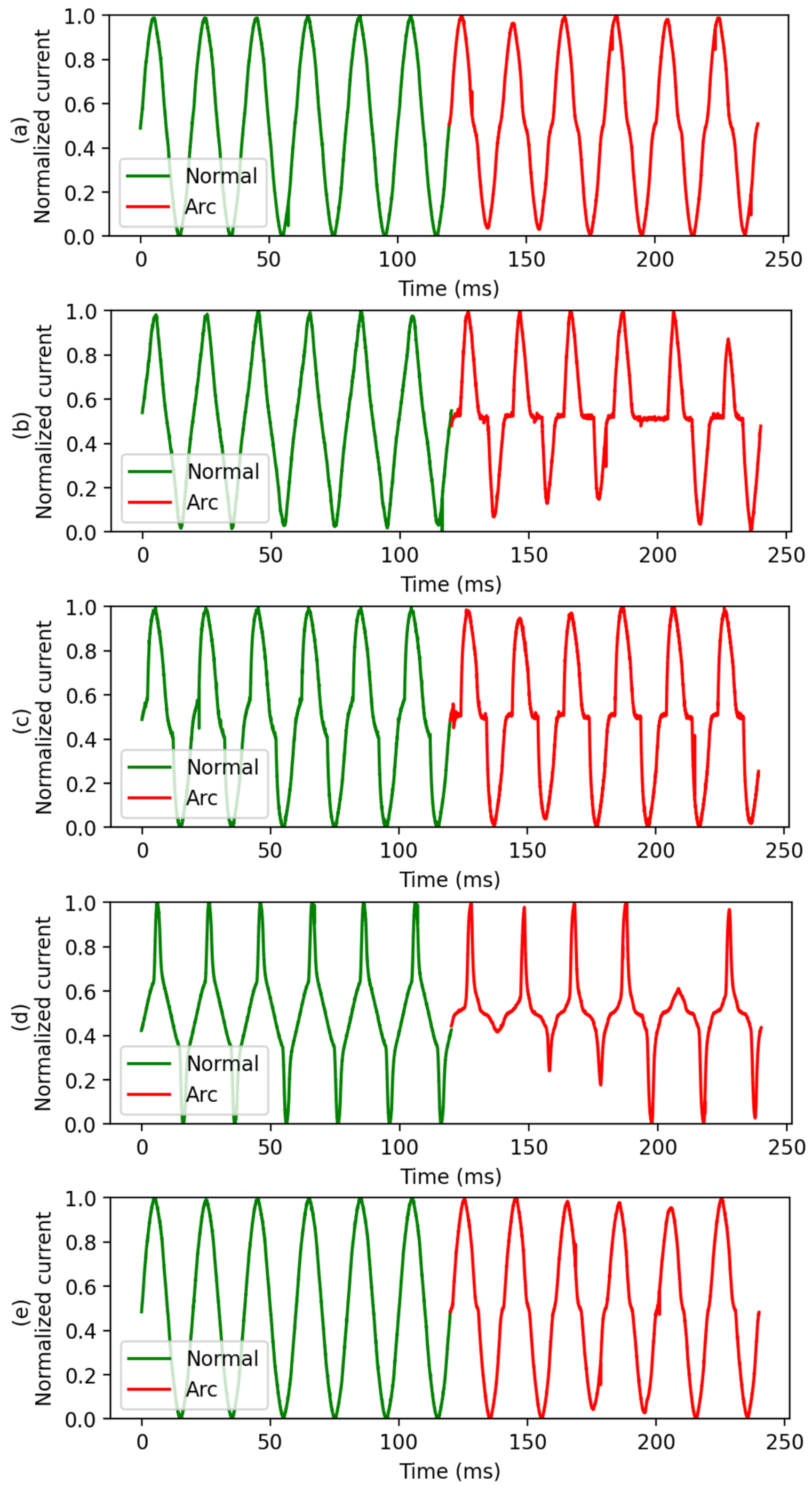


Figure 3.3: Visualization of normal & arcing currents of a) resistive, b) electric Drill, c) thyristor type dimmer, and d) switch-mode power supply

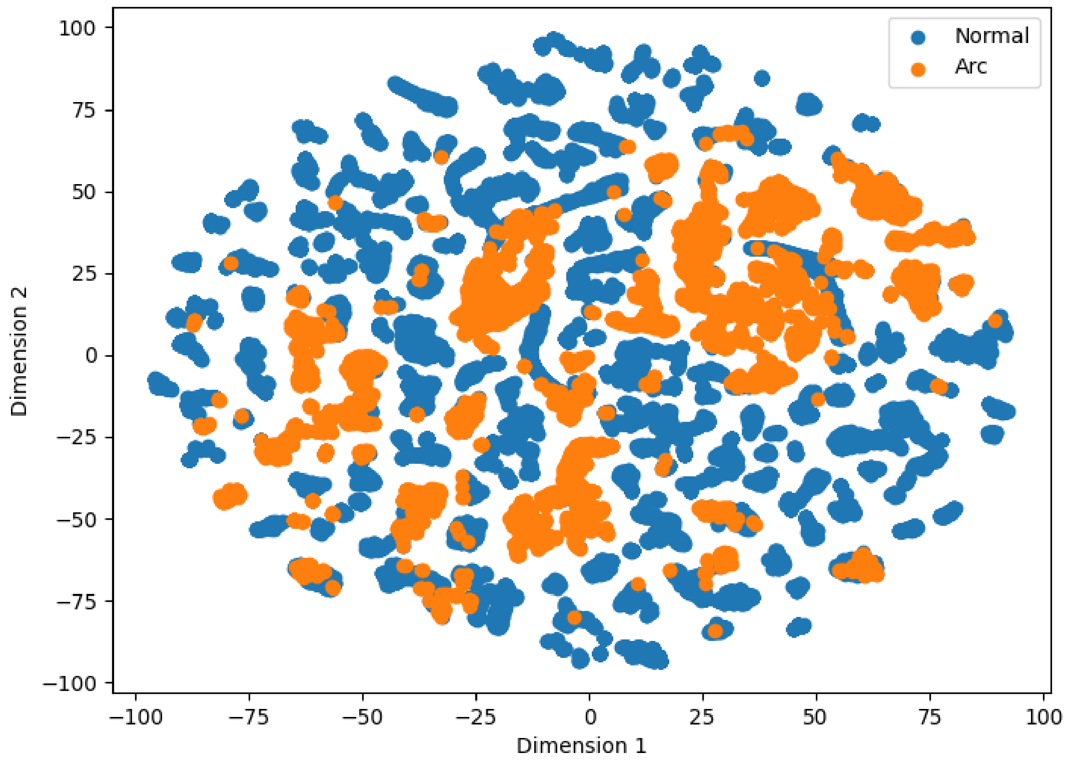


Figure 3.4: t-SNE projection of arc fault data (all load categories) in the frequency domain.

Data for this research were collected using a real test bench designed according to the IEC62606 standard [3], which includes a microcontroller unit (MCU)-based Arc Data Acquisition (DAQ) board, as illustrated in Fig. 3.1. Arc faults were generated in two ways: one with the help of an arc generator and the other by using a cable specimen. A current transformer (CT) was employed to sense the load current. The arc generator was equipped with a stationary graphite electrode and a movable copper electrode, and it was placed in load branch 2. The database includes 30,151 samples across four major load categories from a 220 V, 50 Hz power system, each representing one cycle and initially sampled at 83.33 kHz. For research purposes, the data was downsampled to 10 kHz to optimize arc fault detection, resulting in 200-point 1D time vectors per sample, normalized via Min-Max scaling.

Realistic arc faults were generated using both the arc generator and cable specimens to emulate typical arcing conditions like loose connections and insulation failures. Resistors were added in parallel to meet IEC62606 power requirements. The loads are categorized into four groups: Resistive (RE), Motor (MO), Power Electronics-Enabled and Switch-mode Power Supplies (PESMPS), and Gas Discharge Lamps (GDL). RE loads are near-sinusoidal like heaters; MO includes high inrush current devices such as drills and motors; PESMPS, with broad harmonic content, includes items like dimmers and computers; GDL covers lamps. Each sample is thoroughly labeled - arcing currents assigned even numbers (0, 2, 4, 6, 8) and normal currents odd numbers (1, 3, 5, 7).

3.2.1 Characteristics of Arc Faults and Data Visualization

Arcing currents differ from normal load currents, exhibiting unique characteristics across various load types. Some loads produce predominantly sinusoidal currents, whereas others show flat shoulders and sharp changes, known as catastrophe points. Arcing currents are characterized by distorted waveforms, reduced amplitude, increased high-frequency harmonics, and shorter conduction angles. Importantly, normal currents from certain loads, such as dimmers, may mimic arcing currents from others, like heaters. Arcing typically extends the stagnation period-the time the current 'zero' state or the flat shoulder length in a cycle-and intensifies waveform distortions and high-frequency harmonics. Figure 3.3 compares normal and arcing currents across different loads.

For a deeper analysis, arc fault data has been converted to the frequency domain using Fast Fourier Transformation (FFT) at a 40 kHz sampling rate. This higher sampling rate is chosen to retain more high-frequency details compared to a 10 kHz rate. To make this high-dimensional data more manageable, the t-Distributed Stochastic Neighbor Embedding (T-SNE) method has been applied, as it is known for preserving data point relationships in a low-dimensional space. The result, shown in Fig. 3.4, displays a widespread distribution of arcing and normal currents, highlighting the complexity of setting a clear threshold for distinguishing between them in the frequency domain.

3.3 CNN-based Arc Fault Detection Methodology: ArcNet

To address the complex behavior of arcing current and to classify the arcing and normal state of the load current, a CNN-based algorithm, ArcNet, has been proposed [11]. ArcNet is designed to categorize the normal and abnormal conditions of load currents using eight different types of generalized loads. There is a difference between the arcing current waveforms and the current of normally operating circuits. Although it is sometimes very difficult to differentiate between an arcing and a normal state of the current waveform, often there are shoulders as well as high-frequency components of the current signal in the arc current waveforms, indicating a clear difference from the usual load current. Sometimes high-frequency currents at the zero-crossing position of the current waveshape are evident. From these phenomena of currents, different useful features can be extracted. Additionally, the current signal is a one-dimensional time series data. An arc can happen in any segment at any time of the time series current data.

CNN-based models, e.g. AlexNet [117] were developed for image classification but can also be used for one-dimensional time series data analysis. It can automatically extract informative features as well as map the internal extracted features of the data sequence to a fixed-length representation.

The classification is also included in the same network, leading to an end-to-end framework. It is worth noting that CNN can be applied to the raw current data without any data preprocessing such as discrete wavelet transforms, Fourier transform, or Chirp Zeta transform. Given these advantages of CNN, a 1D CNN-based model, namely ArcNet, has been built to detect arc and non-arc states as well as load types.

A CNN is a trainable hierarchical network composed of multiple stages and has forward and reverse transmission. During the forward transmission, the input features are propagated through a few layers followed by an activation function in each layer, resulting in output features in the output layer. During the reverse transmission process, the error, which is calculated between the results of the forward transmission and the sample tag, is transferred back to each layer through the error function. The network weights and bias parameters are updated using the gradient descent formula.

For 1D convolution, if the input vector is f of length l and the kernel is k of length r, then the i-th convolution of f and k is *(f*k)* using j-th kernel can be defined as:

$$(f * k)(i) = \sum_{j=1}^{r} k(j).f(i - j + \frac{r}{2}) \tag{3.1}$$

During the forward propagation in the fully connected (FC) layer having m inputs and n outputs are computed using the following expression

$$Y = ReLU(\omega_{n\times m} x_{m\times 1} + b_{n\times 1}) \tag{3.2}$$

where ω and b represents the weights and biases respectively. The non-linear activation function, $ReLU$ is given by,

$$ReLU(x) = Max(0, x) \tag{3.3}$$

The categorical Cross Entropy is calculated using the following equation:

$$CategoricalCrossEntropy = -\frac{1}{N} \sum_{j} y_j * log(\hat{y}_j) \tag{3.4}$$

where y represents the true label for a specific class and Å· represents the predicted label, N is the total number of samples.

The network architecture of the proposed ArcNet is illustrated in Fig. 3.5. The proposed network has 4 convolution layers and 4 MaxPooling layers followed by 3 fully connected (FC) layers. Each convolution layer is followed by a Rectified Linear Unit (ReLU) and

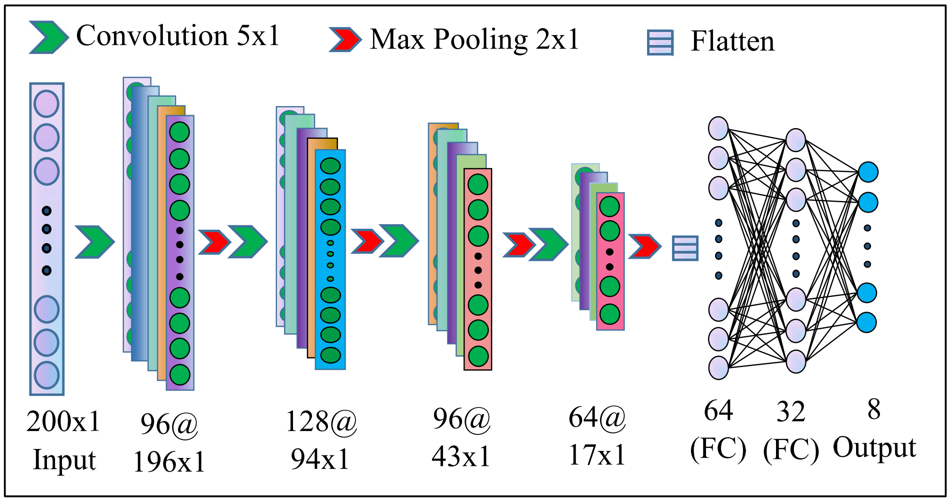


Figure 3.5: The network architecture of the proposed ArcNet model

a MaxPooling layer of pooling size 2x1 with stride 1 to reduce the feature map size for computational efficiency. The input size is 200x1. The 1st & the 3rd convolution layers have 96 filters each. 2nd convolution layer have 128 filters. The 4th convolution layer has 64 filters. All the filters in the 4 convolutional layers have the same kernel size of 5x1. At the end of the last MaxPooling layer, 2 FC layers having 64, and 32 neurons respectively are appended. The last MaxPooling layer is flattened before feeding into the 1st FC layer. Each FC layer is followed by a ReLU operation. Each neuron in the FC layer is connected with all the neurons in its previous layer with learnable weights. The response of a hidden layer unit is calculated by summing over the product of each input signal and its corresponding weight and passing the summation through a ReLU. The output layer has 8 neurons and it is followed by a Softmax layer for categorical multiclass classification (i.e., arc, non-arc and load category). The Softmax function $S(yi)$, as expressed in equation (3.5), turns the logit outputs scores (raw prediction values) into probabilities so that the sum of all the probabilities becomes equal to 1. Finally, the one having the highest probability is considered as the output category. Categorical cross-entropy is employed as the loss function in ArcNet.

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{3.5}$$

The ArcNet model is implemented in Keras using the TensorFlow backend. The ArcNet is trained in 120 epochs with a batch size of 100 using adaptive learning rate. The adaptive learning rate monitored the ”validation_loss” with a patience of 10. The initial learning rate is set to 0.001 and the minimum learning rate is set to 0.00001.

3.4 Experimental Result and Analysis

This section illustrates the experimental results conducted to evaluate the performance of the proposed ArcNet algorithm. In all the experiments the databases containing arc and non-arc signals are used. The samples in the testing dataset are exclusive of the training or validation datasets.

3.4.1 ArcNet Result and Analysis

ArcNet model is trained with the primary database. During the training time, the labels of the 8 classes of the database are converted into categorical data using One-Hot encoder. ArcNet is trained for 120 epochs with a batch size of 100 using the adaptive learning rate strategy with an initial learning rate of 0.00001. The training and validation accuracy plots are shown in Fig. 3.6. The validation accuracy closely follows the training accuracy which validates that the model is not overfitting. While the overall classification accuracy is 99.05%, the maximum arc fault detection accuracy of ArcNet is 99.47% in terms of classifying arc and normal load currents. The confusion matrix for 10 kHz data of different load types is provided in TABLE 3.2. The boldface values in the diagonal of the confusion matrix (from top-left to the bottom-right) designate the samples/percentage of samples which are correctly classified.

It can be observed that most of the resistive arc samples (9) are confused with arc of other loads. This holds true for motor type loads and gas discharge loads too. Apart from

Table 3.2: Confusion matrix of the ArcNet model for 10 kHz data

		Predicted Class							
True Class	Label	0	1	2	3	4	5	6	7
Resistive Load	0	**613** **96.99%**	8 1.27%	3 0.47%	1 0.16%	3 0.47%	1 0.16%	3 0.47%	0 0.00%
	1	2 0.16%	**1266** **99.84%**	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
Motor Type Loads	2	3 1.05%	0 0.00%	**278** **97.54%**	2 0.70%	2 0.70%	0 0.00%	0 0.00%	0 0.00%
	3	0 0.00%	0 0.00%	3 0.45%	**664** **99.55%**	0 0.00%	0 0.00%	0 0.00%	0 0.00%
Power Electronics Loads	4	0 0.00%	0 0.00%	2 0.46%	0 0.00%	**426** **98.38%**	5 1.15%	0 0.00%	0 0.00%
	5	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	**677** **100%**	0 0.00%	0 0.00%
Gas Discharge Loads	6	3 3.41%	0 0.00%	0 0.00%	1 1.14%	0 0.00%	0 0.00%	**83** **94.32%**	1 1.14%
	7	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	**463** **100%**

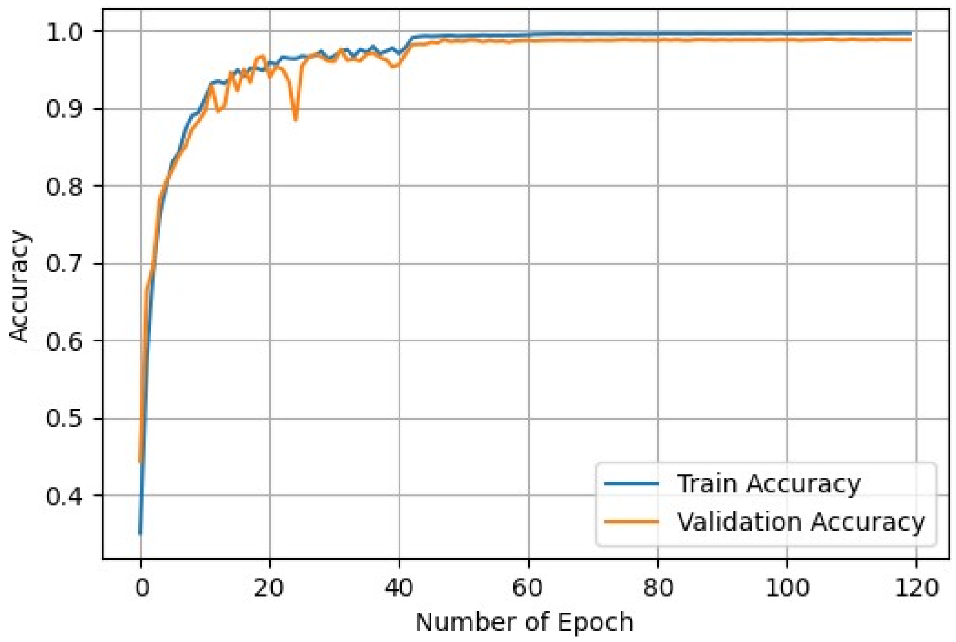


Figure 3.6: Train and validation accuracy plot of ArcNet model.

this, it can be noted that a few samples are misclassified among arc and normal state of the same load type. The numbers/percentage except in the diagonal represent all the misclassified samples.

In machine learning based classification system, a model is evaluated using some performance measures such as accuracy, precision, recall, and F_β scores. These performance metrics are calculated using the confusion matrix as given in TABLE 3.4 which is computed from TABLE 3.2 by grouping all the classified arcs as arc category and all the classified normal currents as normal category. The precision of ArcNet is 99.65% and recall is 98.68% whereas the arc detection accuracy is 99.47%. The F0.5 score of the model is 99.16% and F0.95 score is 99.60%. Near perfect values of precision, recall and accuracy validates ArcNet as a good model for arc fault detection applications. It is also evident from the table that 98.68% samples of arc of the 4 group of loads are correctly recognized as arc. On the other hand, 99.84% normal samples are correctly predicted as normal currents.

Table 3.3: 4-fold cross-validation of ArcNet using 10 kHz data.

k-Fold	Load group & arc classification accuracy (%)	Overall accuracy (%)
1	99.15	99.69
2	98.97	99.32
3	98.94.	99.47
4	99.00	99.29
Mean	**99.02**	**99.44**

Table 3.4: Precision, recall, & accuracy of the ArcNet model (10 kHz)

		Predicted Class		Total
Actual Class		Arc	Normal	
	Arc	1419	19	1438
	Normal	5	3070	3075
	Total	1429	3089	4513

Precision =	99.65%
Recall =	98.68%
Accuracy =	99.47%

3.4.2 Cross-validation

Cross-validation is a resampling procedure used in machine learning models to check the performance metric of the model. In this research, k-fold cross-validation has been used where k=4. The whole dataset is randomly shuffled and split into 4 groups. Each group of the dataset is evaluated while the model is trained with the remaining 3 groups. The overall result of the 4-fold cross-validation is tabulated in Table 3.3. Using each fold of data, the load as well as arc fault classification accuracy along with overall arc fault classification accuracy are determined and tabulated as shown in the table. Using the proposed ArcNet model and the 10 kHz data, the achieved mean accuracy for load group and arc fault classification accuracy is 99.02% and the overall mean accuracy of arc fault detection is 99.44%. These results indicate the suitability of the ArcNet algorithm for practical use.

3.4.3 Analysis for Misclassified Samples

Despite of having near perfect arc fault detection accuracy of ArcNet (10 kHz), a few samples are misclassified. This section digs deeper into the very cause of misclassification by the proposed ArcNet. By analyzing closely, three samples have been selected to find out the reason of misclassification. The first sample is labeled as resistive arc which has been predicted as normal signal (refer to Fig. 3.7). Second sample is labeled as resistive arc and is correctly classified. The final feature vectors after the 4^th max pooling layers of those two samples are extracted. While both samples are classified as resistive loads the first sample is misclassified. A closer look in to the two samples reveal that even though there is a minor signature of arc in the first sample, both a and b samples have a lot of similarity in them which is very obvious from their corresponding extracted features (Fig. 3.7. c & d). Clearly the feature vectors of the first two samples are similar. This makes the model to predict the first sample as normal signal. There are a few other samples in the database which have very minor arc signatures and thus misclassified as normal. In the similar fashion, some current signals of one load type have similarity with the load current of other load group and causing a misclassification by ArcNet.

3.4.4 Model Optimization Based on Sampling Rate and Runtime

The reduction in the sampling frequency of the data results in the reduction of the input size of the CNN which, in turn, is supposed to reduce the computation time as well as less burden to the RAM. This could lead to a simpler network structure as well. In this ablation study the sampling frequency of the data is varied and tested with ArcNet or modified ArcNet(s) to check the performance as well as to find the optimal network

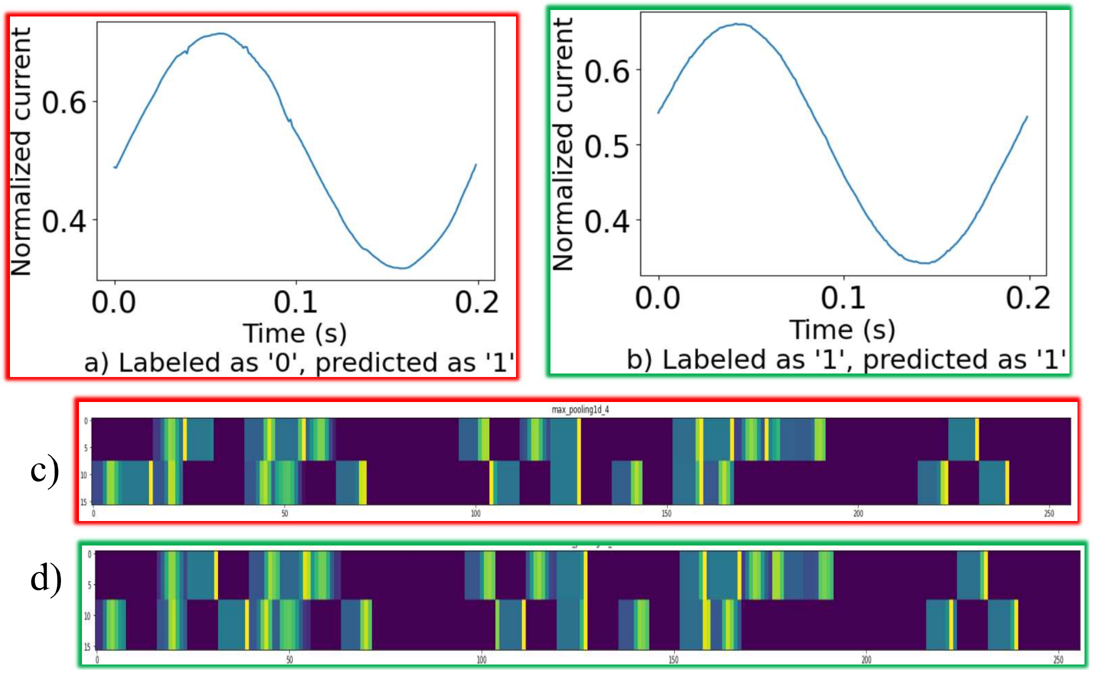


Figure 3.7: Feature visualization for misclassified samples by ArcNet. a) resistive arc recognized as normal current; b) resistive normal current correctly identified; c) feature vectors of a; d) feature vectors of sample b.

Table 3.5: Arc fault detection model architectures, accuracy, and detection time for various sampling rates.

Sampling rate (kHz)	Proposed CNN architecture for arc fault detection	Arc fault detection accuracy(%)
40	$C_1^{96}MC_2^{128}MC_3^{96}MC_4^{64}M + FC(64, 32, 8)$	99.47
10	$C_1^{96}MC_2^{128}MC_3^{96}MC_4^{64}M + FC(64, 32, 8)$	99.47
5	$C_1^{96}MC_2^{96}MC_3^{64}M + FC(64, 32, 8)$	99.45
2.5	$C_1^{96}MC_2^{96}MC_3^{64}M + FC(64, 32, 8)$	99.29
2	$C_1^{96}MC_2^{64}M + FC(64, 32, 8)$	99.00
1.5	$C_1^{96}MC_2^{64}M + FC(64, 32, 8)$	98.69
1	$C_1^{96}MC_2^{64}M + FC(64, 32, 8)$	97.56

architecture and optimal sampling frequency of the data. As the proposed arc detection model considers the raw input data for classification, number of inputs to the model would be decreased with the decrease of the sampling frequency of the data. ArcNet architecture is modified for different data sampling rate by deleting convolution and max pooling layers keeping the same FC layers for all. All the proposed arc detection models along with corresponding fault detection accuracies are illustrated in TABLE 3.5. In the CNN architecture column of the table, in C_1^{96} the letter C indicates conv layer, subscript 1 indicates 1^{st} conv layer, and the superscript 96 indicates that particular conv layer has 96 filters each of them having a kernel size of 5x1. Letter M indicates Max Pooling layer of size 2x1. FC (64, 32,8) denotes 3 fully connected layers having 64, 32, and 8 neurons respectively. Besides, the last FC layer is the output which is followed by Softmax function. The models are trained from the scratch and the hyperparameters are fine-tuned to get the best possible arc fault detection accuracies for 40 kHz, 5 kHz, 2.5 kHz, 2 kHz, 1.5 kHz, and 1 kHz data. Experimental results indicated that as the data sampling rate is reduced, the prediction accuracy starts decreasing except the models for 40 kHz and 10 kHz data both of which ensued maximum arc fault detection accuracy of 99.47%. Although the accuracies of the first two models are same, the later model gave an added benefit which is the reduction of the runtime because of the smaller input size. The model structure for 2 kHz, 1.5 kHz and 1 kHz data are kept same but simpler compared to that of 10 kHz model. In each case, ArcNet models are trained from scratch with the adaptive learning rate and then tested with the corresponding test dataset.

The drop of accuracy with the declination of the data sampling rate is obvious and expected. Because, reduction in the sampling frequency reduces the number of data points in one cycle which essentially causes to lose some important arcing signature in the current signal and the number of extracted features is hampered as well. Therefore, the drop of the accuracy is reasonable. Nevertheless, the recognition accuracy is very high (97.56%) even with a very low sampling frequency of 1 kHz. This indicates two things. Firstly, the decrease in the sampling frequency did not cause a loss of significant arc signatures. Secondly, the proposed models are robust. Even with a lower sampling frequency of only 1 kHz of the data the reduction in the arc fault detection accuracy is not very significant. However, the highest accuracy of the 10 kHz ArcNet model evidenced that the model could be an optimal model for arc fault identification.

In this research, CNN based ArcNet algorithm considers raw data of different load currents to classify the arc and normal state. Features are extracted directly from the raw input data and thus it avoids extra data preprocessing like DFT, DWT, or MFCC which requires extra processing time and increases the computational complexity.

Arc fault detection in real-time scenarios requires faster detection time so that the faulty circuit can be isolated from the healthy portion as quickly as possible without having any damage or loss of property as well as human. For example, the arc fault should be cleared within 0.12s to 1s according to IEC and UL standards. To evaluate such performance of the ArcNet, pre-stored data has been used to verify the algorithm on Raspberry Pi 3B V1.2 (served as an edge device), which has a 1.2 GHz 64-bit quad-core Arm V8 Cortex-A53 CPU. During the runtime checking process in the Pi, the data samples are read from the stored memory of the edge device and then tested by fetching every data sample into the proposed ArcNet model. The reported runtime is the average runtime of all the test samples. The model with 40 kHz sampling rate on an average takes 51 ms to classify every sample. The average runtime per sample reduces with the reduction of the data sampling rate. The average runtime per sample for 10 kHz, 5 kHz, 2.5 kHz, 1.5 kHz, and 1 kHz are 31, 26, 25.5, 21, and 20 ms respectively. This analysis indicates that the proposed models are suitable for real-time operation of arc fault detection application. It is to be noted that, detection runtime is not reduced much with the reduction of the sampling frequency unless the number of convolution layers and max pooling layers are reduced. This is because a convolution layer consumes the major computation time. Therefore, the reduction of sampling frequency from 10 kHz to 5 kHz, which is associated with a reduction of the number of conv layers, reduces the runtime by 5ms whereas the reduction of sampling rate from 5 kHz to 2.5 kHz reduces average runtime by only 0.5 ms which is insignificant.

An optimal arc detection model should have maximum possible arc detection accuracy and minimum possible detection time per sample. UL standard for arc detection specifies that if the load current becomes at most 63 A of a 230 V system, the maximum break time could be 120 ms for a series ac arc fault. The maximum break time for a 120 V system with the same load current is 140 ms. The arc fault detection accuracy and detection time per sample for different arc detection models are plotted in Fig. 3.8. From the plot it can be seen that the prediction accuracy is maximum with the model having 10 kHz sampling rate. The hardware implementation of this model also indicates only 31 ms per sample detection which satisfies the UL standard. Therefore, this study recommends the model of 10 kHz sampling rate as the proposed optimal arc fault detection model (ArcNet) which have 4 conv layers, 4 max pooling layers, and 3 FC layers as depicted in Fig. 3.5.

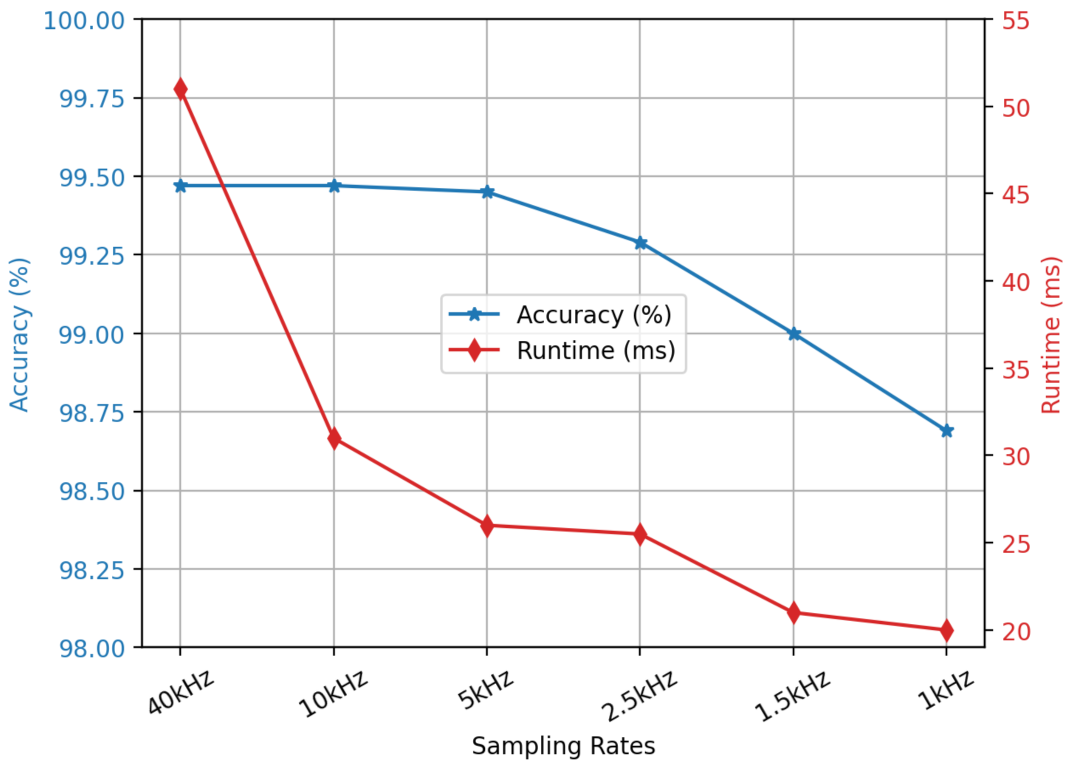


Figure 3.8: Classification accuracy and runtime vs data sampling rate. Mentioned runtime is obtained using Raspberry PI 3B.

Table 3.6: Comparison of LArcNet attributes with prior methods.

Particulars	Qiongfang et al. [72]	Yangkun et al. [20]	Siegel et al. [24]	ArcNet (Ours)
Method framework	Parallel AlexNet (Used two AlexNet in parallel).	HTFNN (Applied time& frequency analysis before feeding to a neural network).	Applied (DFT + DWT + MFCC) to the data for processing and then fed to the deep neural network.	Applied CNN based network directly to the raw data without processing it.
Max. detection accuracy	97%	99%	99.95%	99.47%
Number of types of loads	3	5	3	8 (in 4 groups)
Data sampling rate	50 kHz	25 kHz	5.865 kHz	10 kHz
Arc generation methods	1	1	1	2
Suitable for industrial MCU	No	Yes	Yes	Yes

3.5 Comparison of ArcNet Attributes With Prior Methods

In order to evaluate and compare the ArcNet method with the other published methods, a few key evaluation metrics are established.

1. Can the detection method achieve high recognition accuracy, faster detection speed which can be implemented in real-time applications and should detect arc fault for a wide variety of load types?
2. Does the method consider signals of a wider bandwidth so that the signal can contain more detailed information of arcs which will help to avoid nuisance tripping when some interference is present in the current signal?
3. Is the method able to detect the most common types of arcs including arc from a loose connection of cables and those caused by the failure of the insulation?

A comparative analysis of the proposed ArcNet [11] method and other methods has been summarized in TABLE 3.6. It can be seen that the proposed ArcNet algorithm can have arc detection precision of 99.47% and considered 8 types of loads in the database. The accuracy of ArcNet is slightly less compared to some other methods indicated in the comparison table. This is because the ArcNet focused on the robustness of the fault detection for a wide variety of loads instead of focusing on smaller number of loads. A model with a fewer number of loads in the database though makes the model easier to detect arc faults but the algorithm loses its robustness and makes it impractical.

A data sampling rate of 10 kHz, has been considered in the ArcNet database so that the raw input current signal can have detailed but optimal features of the arc to avoid unwanted tripping for interferences. Two types of arc generation methods, including loose connection of cables and failure of the insulation, are considered in the database so that the model can learn the most common types of arc faults and can detect them in real-world applications.

3.6 Summary

In summary, a novel convolutional neural network-based algorithm named ArcNet is proposed. ArcNet, comprised of four convolutional layers followed by three fully connected layers, is designed for series AC arc fault detection. Using samples of 8 types of generalized household load currents, ArcNet achieved an arc fault detection accuracy of 99.47% and load classification along with arc fault detection accuracy of 99.05% for 4 major load groups. 4-fold cross-validation is performed for the ArcNet model with the 10 kHz data which also showed a mean overall arc fault detection accuracy of about 99.44%. Besides, modified ArcNet models are proposed which can achieve a reasonably good performance with a smaller sampling rate of the training set. The arc generation method of this research simulated the most common arc situations including the loose connection of cable and insulation breakdown that happens in the real-world. Therefore, the proposed arc identification algorithm will be more robust in nature in real-world applications. Besides, a higher number of load types have been considered for the experiment to make it more appropriate for practical application. The proposed algorithm can not only detect arc faults but also can identify the load types for which the fault has occurred. ArcNet model is implemented using Raspberry Pi 3B. The time taken to examine one test sample by ArcNet is only about 31 ms at a 10 kHz sampling rate which indicates the feasibility of this algorithm in real-time arc fault detection applications.

Even though a satisfactory arc fault detection accuracy has been achieved by ArcNet, the computational efficiency is still very high for low-end MCUs. Therefore, it needs further optimization. Future work of this research includes model optimization and compression using a lightweight CNN algorithm and knowledge distillation method.

CHAPTER 4: MODEL OPTIMIZATION USING EFFICIENT & LIGHTWEIGHT CNN ALGORITHM

The use of traditional CNN-based algorithms can help to get rid of complex data preprocessing time and yield a high arc fault detection accuracy. Despite their effectiveness, the computational complexity of these models makes them unsuitable for implementation in low-cost commercial microcontroller units (MCUs) having limited computing resources [26, 118]. Slower models may miss critical faults and also safety and reliability may be hampered. The attempt to model simplification by reducing the sampling frequency to too small [71] or lowering the window size to less than 1 cycle hurts efficient arc fault detection.

Therefore, developing an efficient, lightweight model for arc fault detection is crucial. Advanced AI techniques like pointwise and depthwise separable convolutions [34, 36] can help to build lightweight and efficient AI models. Additionally, model compression methods like parameter pruning and knowledge distillation [37] can minimize model size, enhance efficiency, and reduce latency.

In this chapter, an optimized arc fault detection algorithm is proposed that combines a lightweight CNN architecture with a teacher-student knowledge distillation technique. This model achieves high accuracy in arc fault detection. Furthermore, it is optimized using the TensorFlow Lite (TF-Lite) tool, resulting in a reduced binary size suitable for implementation in resource-limited edge devices. The performance of the optimized model is evaluated on a Raspberry Pi 4B device, demonstrating its practicality and real-time operational capability.

The key contributions of this work are as follows:

1. An extremely fast and lightweight AI algorithm, LArcNet, has been proposed to detect series AC arc faults using raw current data. This model combines a teacher-student knowledge distillation method with an efficient CNN architecture, achieving a high fault detection accuracy of 99.31%. By avoiding the pitfalls of traditional bottleneck designs, it ensures low computational demand. LArcNet was trained and tested using real-world data collected from a test bench as per the IEC62606 standard.

2. To optimize LArcNet for real-time applications on resource-constrained devices, model compression techniques such as knowledge distillation and mixed precision training have been employed. These enhancements, coupled with TensorFlow Lite optimization, have substantially reduced the model's memory usage and computational overhead. The optimized LArcNet model achieved an inference latency as low as 0.20 ms per sample when deployed on a Raspberry Pi 3B MCU. With its lightweight design and rapid response time on embedded devices, LArcNet is exceptionally well-suited for real-time fault detection applications.

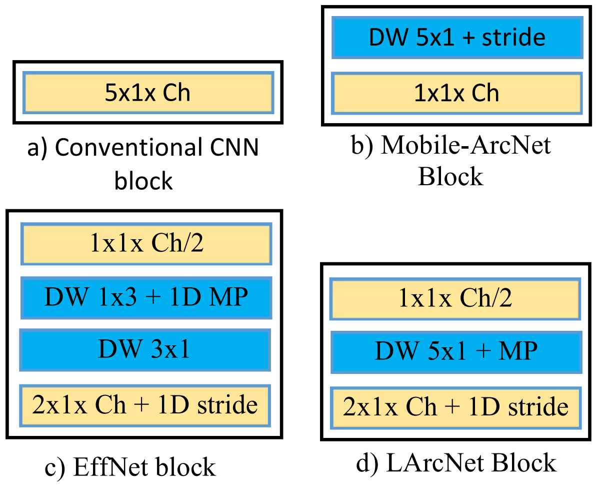


Figure 4.1: Neural network model building blocks a) conventional CNN, b) Mobile-ArcNet [34], c) EffNet [36] and d) LArcNet. 'DW' means depthwise convolution, 'MP' refers to Max Pooling, 'Ch' denotes the number of output channels. The number of strides was set to 2.

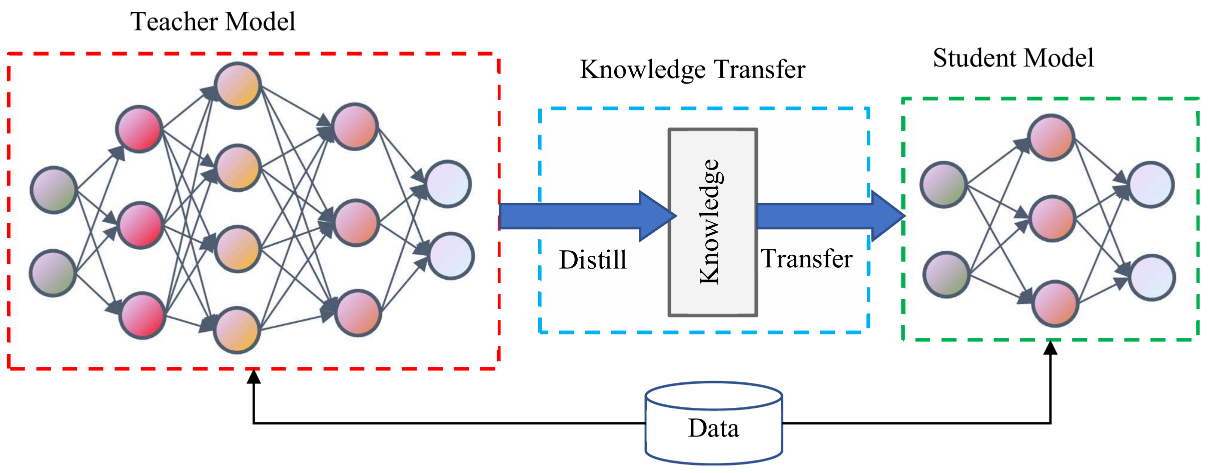


Figure 4.2: Framework of the teacher-student knowledge distillation process.

4.1 Proposed Methodology of Model Optimization

This section elaborates on the fundamental building block of LArcNet, architecture and methodology underlying the proposed LArcNet model. LArcNet integrates advanced design strategies from both efficient neural network architectures, such as EffNet [36] and network compression techniques including knowledge distillation. The LArcNet model is constructed using the knowledge distillation method and a series of LArcNet blocks (as shown in Fig. 4.1(d)), each with varying numbers of filters. This hybrid approach combines the strengths of these methods to optimize performance and reduce computational load. The model is compared against a baseline CNN model constructed using conventional CNN blocks and the Mobile-ArcNet model which was constructed following the MobileNet architectural block (Fig. 4.1(b)).

4.1.1 Efficient Model Building Block of LArcNet

The LArcNet model employs an efficient network architecture inspired by the EffNet block [36] (illustrated in Fig. 4.1(d)). Designed to minimize computational complexity, the LArcNet block begins with a pointwise convolution layer, followed by depthwise convolution and Max Pooling in the subsequent sub-layer. This configuration significantly reduces computational demands and the number of trainable parameters compared to traditional CNN structures. Notably, the pointwise convolution layer uses half the number of filters typically found in standard CNN models. The depthwise separable layer, sized 5x1, maintains the initial filter count and is followed by a 1D Max Pooling layer with a pool size of 2. A subsequent 2x1 convolutional layer employs the full filter count with a 1D stride of 2. Diverging from MobileNet's design, LArcNet avoids stringent bottleneck structures, leading to a lighter model. Focused on 1D time-series data, LArcNet omits certain sub-layers from the EffNet block. The experiments demonstrated that a 5x1 kernel size in the depthwise sub-layer achieves a better balance between computational demand and performance efficiency than the 3x1 size.

4.1.2 Knowledge Distillation Algorithm

The LArcNet model employs a teacher-student knowledge distillation (KD) method, a model compression strategy that improves efficiency significantly. In this paradigm, a complex teacher network trains a simpler student network to replicate its functionality, often achieving comparable performance. This KD technique includes three core components: the knowledge, the teacher-student framework, and the distillation process, depicted in Fig. 4.2. LArcNet uses response-based knowledge, where a robust teacher model imparts refined information to a simpler student model. The student model mirrors the teacher CNN and fully connected (FC) layers but with fewer filters and neurons, maintaining essential structural features.

The output layer of LArcNet uses a 'softmax' activation function, which transforms class logits, z_i, into normalized probabilities, p_i, ensuring that the probabilities sum to one. This process is described by $p_i = \frac{e^{\frac{z_i}{T}}}{\sum_j e^{\frac{z_j}{T}}}$ Here, T is the temperature parameter in the softmax function. Raising T creates a softer distribution over classes, smoothing the output probabilities. The teacher model is trained with a higher T to produce these softened distributions, which are then used as targets for training the student model. When labels are known, the student's training can be enhanced by combining the cross-entropy from the softened targets with the cross-entropy from the actual labels, using logits at $T = 1$.

Suppose the comprehensive teacher model yields a softened probability denoted by q_i and possesses logits labeled as u_i. The cross-entropy loss in this framework can be expressed by $C(x) = -\sum_i q_i(x) \log p_i(x)$ where, x is the input feature. The cross-entropy gradient $\frac{\partial C}{\partial z_i}$, of the distilled model [37, 119] can be expressed as

$$\begin{aligned}
\frac{\partial C}{\partial z_i} &= \frac{\partial}{\partial z_i}\left(-\sum_i q_i \log p_i\right) \\
&= -q_i \frac{1}{p_i} \frac{\partial}{\partial z_i}\left(\frac{e^{\frac{z_i}{T}}}{\sum_j e^{\frac{z_j}{T}}}\right) \\
&= -q_i \frac{1}{p_i}\left[\frac{\frac{1}{T} e^{\frac{z_i}{T}} \sum_j e^{\frac{z_j}{T}} - \frac{1}{T}\left(e^{\frac{z_i}{T}}\right)^2}{\left(\sum_j e^{\frac{z_j}{T}}\right)^2}\right] \\
&= \frac{1}{T}(p_i - q_i) \\
&= \frac{1}{T}\left(\frac{e^{\frac{z_i}{T}}}{\sum_j e^{\frac{z_j}{T}}} - \frac{e^{\frac{u_i}{T}}}{\sum_j e^{\frac{u_j}{T}}}\right)
\end{aligned} \tag{4.1}$$

where, u_i represents the logits derived from the teacher model, while q_i signifies the softened target probabilities calculated at a transfer training temperature T. When employing a high temperature setting, the gradient of the cross-entropy loss with respect to the logits, z_i, can be approximated as:

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T}\left(\frac{1 + \frac{z_i}{T}}{N + \sum_j \frac{z_j}{T}} - \frac{1 + \frac{u_i}{T}}{N + \sum_j \frac{u_j}{T}}\right) \tag{4.2}$$

Under the assumption of zero-mean logits for each transfer case as per Hinton et al. [37], where $\sum_j z_j = \sum_j u_j = 0$, equation (4.2) simplifies to:

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{NT^2}(z_i - u_i) \tag{4.3}$$

In the distillation process, assuming logits have zero mean for each transfer case, the focus is on minimizing the squared error $\frac{1}{2}(z_i - u_i)^2$ as the temperature T approaches higher values. At lower temperatures, emphasis on matching logits decreases. The teacher model's loss function is unconstrained during training, potentially introducing noise. Notably, negative logits are valuable as they reveal insights about the knowledge encoded by the teacher model.

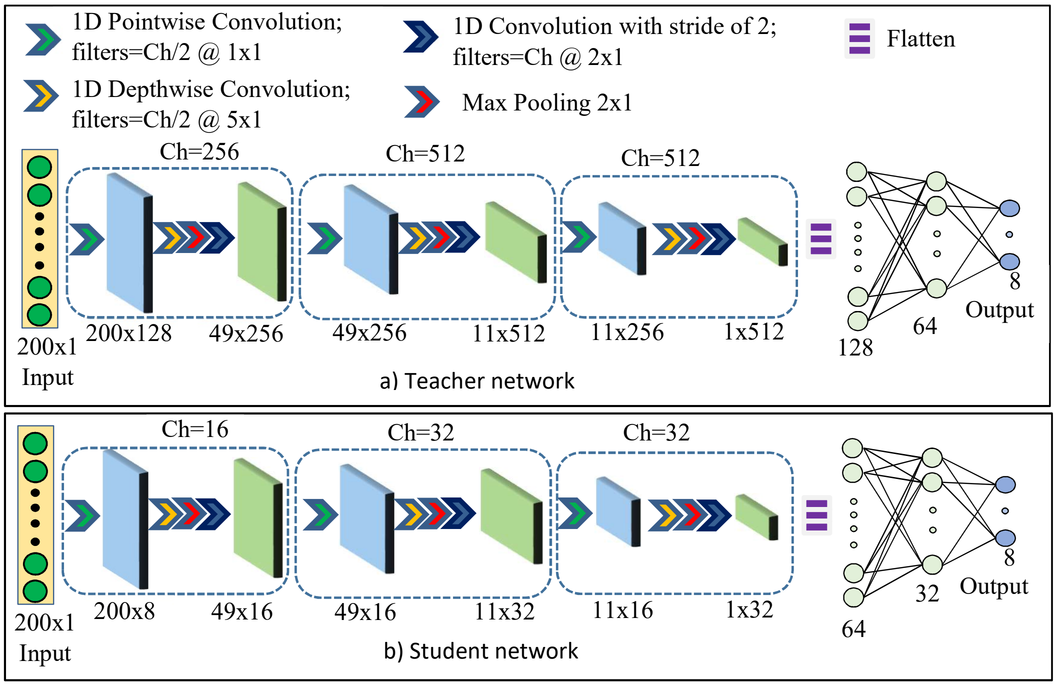


Figure 4.3: Architectures of the LArcNet (a) teacher and (b) student networks consist of three LArcNet blocks. Each block includes a pointwise convolution with Ch/2 filters, a 5x1 depthwise convolution, followed by a 2x1 Max Pooling layer, and a 2x1 convolution with Ch filters. Both architectures conclude with three fully connected layers, including an output layer with 8 neurons. The student model mirrors the teacher structure but with fewer filters and neurons.

4.1.3 Network Architecture of LArcNet Model

The LArcNet model includes both a teacher and a student network, each featuring three LArcNet blocks and three fully connected (FC) layers. The teacher network uses 256, 512, and 512 filters across the blocks, and the FC layers have 128, 64, and 8 neurons, respectively. The student network, designed for efficiency, uses fewer filters (16, 32, & 32) and maintains the same neuron configuration in the FC layers (64, 32, and 8) to balance

Table 4.1: Network architecture and data flow of LArcNet (student model) compared to the baseline model, Mobile-ArcNet, and Efficient-ArcNet. All feature three fully connected layers (64, 32, and 8 neurons) for 8 output classes across 4 load groups. Depthwise convolution ('DW') and max pooling ('mp') are labeled. Efficient-ArcNet uses a bottleneck factor of 4 (red), while LArcNet uses a maximum of 2 (green).

Baseline		Mobile-ArcNet		Efficient-ArcNet [18]		LArcNet (Proposed)	
Layers	Params	Layers	Params	Layers	Params	Layers	Params
5x1x96 + mp of 2	576	5x1x96 + mp of 2	576	1x1x48 dw 5x1 + mp of 2 2x1x96 + stride of 2	96 2592 9312	1x1x8 dw 5x1 + mp of 2 2x1x16 + stride of 2	16 112 272
5x1x128 + mp of 2	61568	dw 5x1 + stride of 2 1x1x128	12896 16512	1x1x64 dw 5x1 + mp of 2 2x1x128 + stride of 2	9208 4480 16512	1x1x16 dw 5x1 + mp of 2 2x1x32 + stride of 2	272 352 1056
5x1x128 + mp of 2	82048	dw 5x1 + stride of 2 1x1x128 + mp of 4	17152 16512	1x1x64 dw 5x1 + mp of 2 2x1x128 + stride of 2	4480 16512 8256	1x1x16 dw 5x1 + mp of 2 2x1x32 + stride of 2	528 352 1056
FC (64,32,8)	174440	FC (64,32,8)	41568	FC (64,32,8)	10600	FC (64,32,8)	4,456
Total Params	318,016		107,016		79,048		8,472

performance and simplicity without sacrificing accuracy. Figures 4.3 and Table 4.1 depict their architectures data flow respectively.

An aggressive bottleneck for data flow through a network can cause a huge reduction of significant features along the way. This might have a destructive effect when applied to a smaller deep model. The LArcNet model reduces computational complexity and kFLOPs by using an efficient block design and knowledge distillation, which minimizes the count of trainable parameters. It avoids the high bottleneck factor seen in the Efficient-ArcNet model, which has a factor of 4 (shown in red in Table 4.1). Instead, the student model of LArcNet limits the bottleneck factor to a maximum of 2 (highlighted in green), facilitating smoother data flow and enhancing efficiency. This approach is crucial because narrower models cannot afford significant filter reduction without losing important features.

4.2 Experimental Settings and Implementation

This section details the setup and implementation of the LArcNet model. The dataset was split into training, testing, and validation sets in a 75:15:10 ratio, with labels converted to OneHot encoding. Model inputs consisted of raw current data sampled at 10 kHz, normalized using min-max normalization.

The training process utilized a mixed precision system, combining 16-bit half-precision for data processing and 32-bit single-precision for weight updates and loss scaling. The LArcNet teacher and student models were trained for 300 epochs with a batch size of 100, using an Adam Optimizer at a learning rate of 0.001. Parameters for the distillation process included an alpha value of 0.5 and a temperature of 20, facilitating nuanced guidance and preventing overfitting. Training and validation accuracies and losses for both models were closely monitored, as shown in Fig. 4.4.

Baseline and Mobile-ArcNet models underwent 250 epochs of training with an adaptive learning rate strategy starting from 0.001, adjusted based on validation loss. Both models used a batch size of 100, a patience parameter of 10, and a reduction factor of 0.1 for learning rate adjustments.

All models were trained using TensorFlow with Keras, employing categorical crossentropy as the loss function and ReLU in the convolution layers. The output layers used the softmax activation function. Mixed-precision training involved converting data samples to 16-bit floating points for efficiency. Extensive hyperparameter tuning optimized the model's performance.

Table 4.2: Experimental results of LArcNet and other models. Best results are boldfaced.

Model	**8 Class Accuracy**	**Accuracy (%)**	**Trainable Params**	**kFLOPs**	**Factor**	**Runtime (ms)**
Baseline	98.96 %	99.22	318.63 k	19205.4	1	3.27
Mobile-ArcNet	99.00 %	99.29	107.02 k	4542.93	0.24	1.43
ArcNet [11]	99.85 %	99.47	189.64 k	18209.5	0.24	31
Efficient-ArcNet [18]	-	99.36	79.05 k	3620.0	-	1.17
GBDT[a]	-	99.65	-	-		10.22
LArcNet (Proposed)[b]	98.85 %	99.31	**8.4 k**	**182.27**	**0.01**	**0.20**

[a]Gradient Boosted Decision Tree model was developed for this research
[b] Input to the model is raw data (sampled at 10 kHz) with Min-Max normalization.

4.3 Experimental Validation, Result and Discussion

This section presents the experimental results of the LArcNet model along with the Baseline and Mobile-ArcNet models. All models were trained on the same dataset. The Baseline model, comprising only conventional convolutional blocks, has the highest number of trainable parameters and limited computational efficiency, despite the inclusion of Max Pooling layers. Conversely, the Mobile-ArcNet model has a higher parameter count (107.2 k) compared to LArcNet and was not trained using knowledge distillation. The student model of LArcNet, however, utilized the knowledge distillation technique, benefiting from the comprehensive teacher model. The LArcNet student model features the smallest

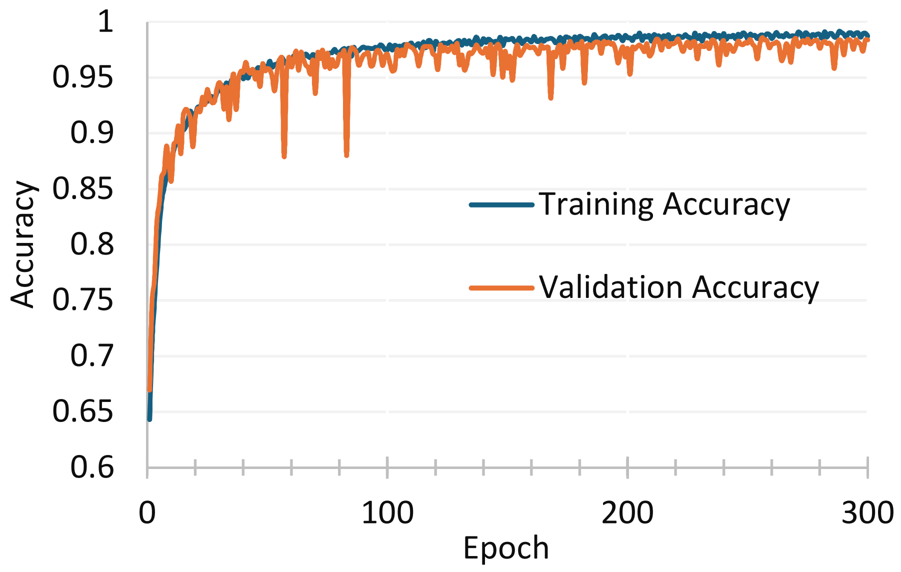


Figure 4.4: Train & Validation accuracy plot of the LArcNet student model.

number of trainable parameters among the models, totaling only 8.4k. The Raspberry Pi 4B served as the microcontroller unit (MCU) for implementing the models.

The kFLOPs (floating-point operations per second) for each model were estimated to understand the computational complexity of the models. FLOPs is a measure of the performance of a computer system, used to estimate the amount of computational resources required to perform a certain task. The higher the FLOPs, the more complex the computation of the model. Compared to the kFLOPs of the Baseline model, Mobile-ArcNet has a factor of 0.24, and the LArcNet model has a factor of only 0.01. The LArcNet has only 182.27 kFLOPs of computational burden. This indicates that the proposed LArcNet model has the least computational complexity and is therefore lightweight. Detailed results are tabulated in Table 4.2.

Accuracy-wise, LArcNet achieved the highest arc fault detection accuracy of 99.31%. The baseline model scored the lowest in arc fault detection at 99.22% (Table 4.2). The Load classification and arc fault classification accuracy using the LArcNet model is depicted in the confusion matrix (Table 4.3), highlighting the accuracy for 8-class classifications of different load groups and misclassification percentages. Even-numbered labels indicate arc faults, and odd-numbered labels normal currents. The GDL loads demonstrated the lowest accuracy due to their characteristics, where normal currents can resemble arc faults, leading to significant misclassification. Approximately 10.23% GLD arcs are mistakenly classified as resistive arcs. Overall binary classification accuracy for arc fault detection was calculated by summing correctly classified arc faults and dividing by the total number of test samples.

Precision, recall, and binary classification accuracy for arc and normal current samples were determined without considering load classification. The precision and recall matrix of the LArcNet model, presented in Table 4.4, was derived from Table 4.3. High precision and recall values indicate the model's practicality and low rate of false positives.

The LArcNet model boasts a simpler yet efficient structure. It includes a large and comprehensive teacher model that effectively learns arc features with high accuracy. This design allows the student model to be significantly more lightweight, benefiting from the knowledge distilled from its larger counterpart. Notably, the Mobile-ArcNet model retains its original first layer without a depthwise layer, whereas LArcNet replaces its first layer with efficient design blocks, significantly reducing computational load. Additionally, the proposed model employs a maximum bottleneck factor of 2, enhancing data flow and reducing computational complexity. The integration of a moderate bottleneck structure, pointwise and depthwise separable convolutions, and the teacher-student knowledge distillation technique makes the model extremely lightweight and reduces the number of kFLOPs substantially. As a result, LArcNet achieves a lightweight framework that maintains high performance in detecting series AC arc faults accurately, minimizing both computational costs and runtime, and making it an efficient solution for its intended application.

Table 4.3: Confusion matrix for 8-class classification using the LArcNet model. Correctly predicted percentage values are boldfaced diagonal numbers. Even-numbered labels represent arc classes, and odd-numbered labels represent normal classes.

		Predicted Class							
True Class	Label	0	1	2	3	4	5	6	7
Resistive loads	0	**97.63%**	1.74%	0.47%	0%	0%	0%	0%	0.16%
	1	0.47%	**99.29%**	0%	0%	0%	0%	0%	0.24%
Motor Loads	2	0.35%	0%	**98.25%**	0.70%	0.35%	0%	0.35%	0%
	3	0%	0%	0%	**100%**	0%	0%	0%	0%
Power electronics & SMPS loads	4	0.23%	0%	0.46%	0%	**97.23%**	2.08%	0%	0%
	5	0%	0%	0%	0%	0%	**100%**	0%	0%
Gas Discharge lamps	6	10.23%	0%	0%	1.14%	0%	0%	**87.50%**	1.14%
	7	0%	0%	0%	0%	0%	0%	0%	**100%**

Table 4.4: Precision, recall, and accuracy of the LArcNet model.

		Predicted Class		
Actual Class		Arc	Normal	Total
	Arc	1413	25	1438
	Normal	6	3069	3075
	Total	1419	3094	4513
	Precision		99.58%	
	Recall		98.26%	
	Overall accuracy		99.31%	

4.3.1 Model Optimization and Hardware Implementation

TensorFlow Lite (TF-Lite) [120], an open-source framework, facilitates the running of TensorFlow models on edge devices with limited resources such as microcontrollers and mobile devices. TF-Lite optimizes models to reduce binary size and latency. The student model of LArcNet was optimized to decrease its binary size from 600 kB to 49 kB resulting in enhanced performance for resource-limited edge computing devices.

The optimized LArcNet model was then implemented on a Raspberry Pi 4B to evaluate its real-time performance. The process involved initializing the TF-Lite interpreter, allocating tensors to set model inputs and outputs, and testing the model using sample sets. The model performance, tested five times using all samples, showed an average inference time of only 0.20 ms per sample without sacrificing accuracy. The test time included reading a sample, normalization, label fetching, and testing. Fig. 4.5 shows that LArcNet outperforms other state-of-the-art models in runtime with comparable accuracy.

Parallel assessments were conducted for the Baseline and Mobile-ArcNet models, with their inference times summarized in Table 4.2. LArcNet demonstrated superior performance in both accuracy and runtime, confirming its suitability for real-time practical implementation in commercial MCUs. The runtime assessments were performed over one cycle of power frequency, showcasing the model's effectiveness in real-time applications.

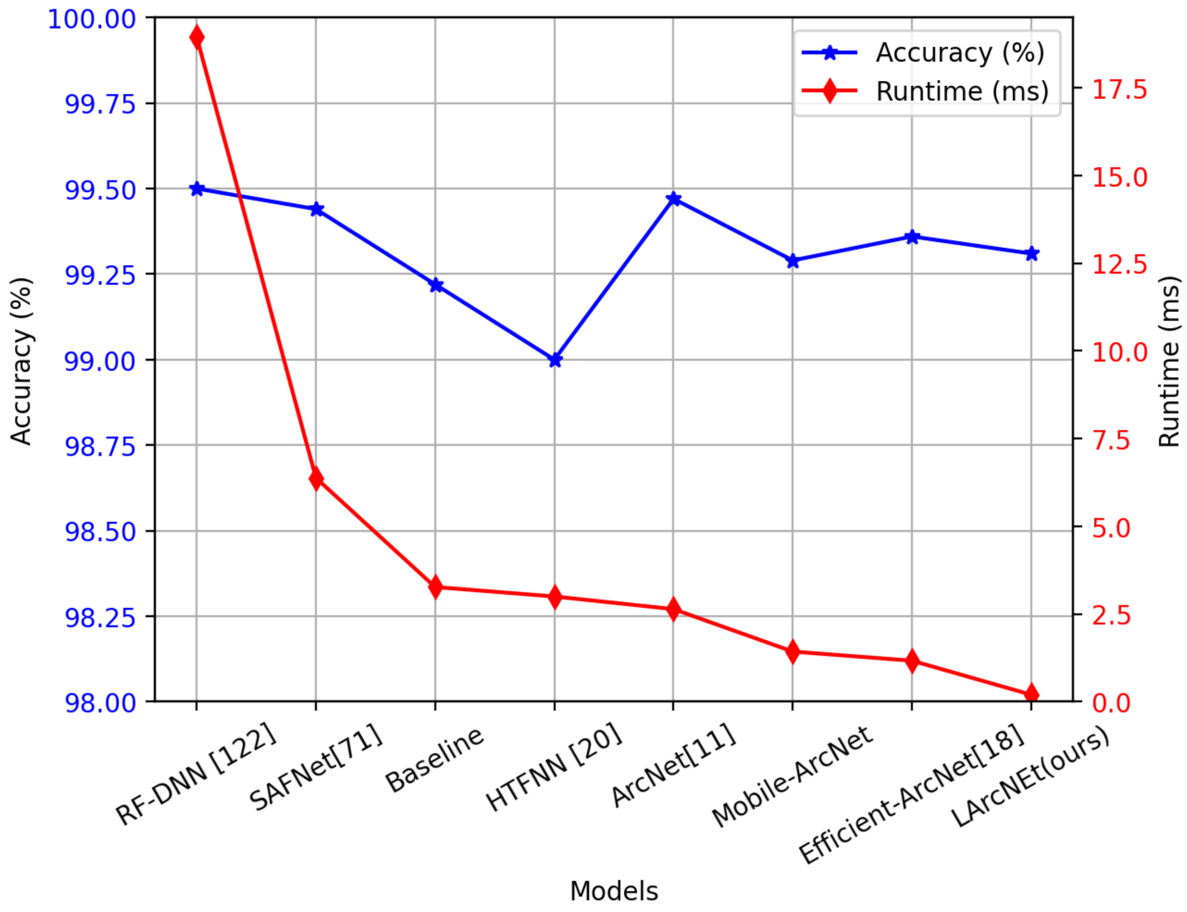


Figure 4.5: Comparison of the LArcNet with the state-of-the-art models.

4.4 Comparative Analysis of LArcNet with Contemporary Methods

This section offers an in-depth comparison of LArcNet with other recent methodologies in AC arc fault detection, as presented in Table 4.5. This comparison is crucial for showcasing the advancements and efficiencies of LArcNet in the field.

Reference [11] initially reported the longest runtime among the compared techniques at 31 ms per sample due to a lack of optimization. This research has optimized the method, reducing its runtime significantly to 2.64 ms, demonstrating potential efficiency gains through optimization. Other studies, such as those in References [21] and [121],

Table 4.5: Comparison of attributes with prior methods. Best results are boldfaced.

Model	**Samples Tested**	**8 Class Accuracy (%)**	**Accuracy (%)**	**Trainable Params**	**kFLOPs**	**Sampling Rate (kHz)**	**Runtime (ms)**	**Implementation Device**
RNN [21]	5029	-	98.70	-	-	-	-	Not implemented
Sparse coeff. & NN [121]	8000	-	94.30	-	-	-	-	Not implemented
SAFNet [71]			99.44	593.32 k	12202.24	-	6.38	Jetson Nano
ArcNet [11]	4513	**99.05**	99.47	189.64 k	18209.5	10	31	Raspberry PI 3B
RF-DNN [122]	-	-	**99.5**	-	-	-	18.95	TMS320F28335
HTFNN [20]	1000	-	99.00	-	-	-	3	STM32F407ZG
FCNN [123]	-	-	98.05	-	-	5	6.38	Jetson Nano
LightGBM [19]	2400	-	97.06	-	-	-	300	Jetson Nano
LArcNet (Ours)	4513	98.85	99.31	**8.4 k**	**182.27**	10	**0.20**	Raspberry PI 4B

reported respectable arc fault detection accuracies (98.70% and 94.30%, respectively). However, they do not provide comprehensive implementation details on low-end commercial MCUs, crucial for practical applications. SAFNet [71] employs a 2D input that increases computational load, resulting in a slower inference time of 6.38 ms per sample compared to LArcNet. Additionally, SAFNet's testing on the Jetson Nano, which has superior GPU capabilities compared to the Raspberry Pi 4B, focuses on a single load type, limiting its applicability compared to LArcNet's multi-load capacity. Reference [19] proposed a method with lower accuracy and a higher computational burden when implemented on the same platform as LArcNet.

In contrast, LArcNet not only identifies various load groups but also detects arc faults across four major load groups with an exceptional accuracy of 99.31%. Most notably, it achieves a remarkably low runtime of only 0.2 ms per sample. This efficiency results from LArcNet's triple simplification approach: architectural optimization to reduce computational load, model compression via knowledge distillation to enhance performance, and TF-Lite optimization to decrease binary size and latency. This comprehensive strategy makes LArcNet highly suitable for real-time applications in resource-constrained environments.

Overall, LArcNet stands out as a leading solution in arc fault detection, offering a perfect balance between high accuracy and low computational demand, making it ideal for practical implementation in edge devices.

4.5 Summary

In this study, an innovative, ultra-fast algorithm for high-performance arc fault detection, has been developed, combining efficient network architecture with advanced model compression techniques. Using a teacher-training knowledge distillation approach, the proposed LArcNet model processes raw current inputs with minimal computational complexity and high efficiency.

LArcNet's lightweight yet robust structure provides arc fault detection accuracies comparable to or better than traditional models like the Baseline CNN and Mobile-ArcNet. It achieved a load classification accuracy of 98.85% and an arc fault detection accuracy of 99.31%. Although models like Efficient-ArcNet may offer similar accuracy, they involve increased architectural complexity.

Transforming LArcNet into a TensorFlow Lite (TF-Lite) model significantly reduced its binary size and latency, optimizing it for real-time applications. Tested on a Raspberry Pi 4B MCU, LArcNet excelled, demonstrating superior performance with an average runtime of just 0.20 ms per sample. This efficiency of the proposed model not only highlights its practicality for commercial deployment but also establishes it as a leading solution in series AC arc fault detection.

In summary, LArcNet demonstrates high accuracy and reduced inference time, proving its viability for practical use in embedded hardware systems. Future efforts will focus on refining LArcNet to handle unknown load scenarios, enhancing its applicability across various real-world conditions.

CHAPTER 5: DC ARC FAULT CIRCUIT INTERRUPTER USING LIGHTWEIGHT CNN ALGORITHM AND SSCB

AI-based methods have demonstrated high accuracy in detecting photovoltaic (PV) arc faults. However, their computational complexity poses a significant challenge for implementation in commercial microcontroller units (MCUs) which have limited memory and RAM. This necessitates an arc fault detection algorithm that is not only highly accurate but also lightweight enough to be implemented on edge devices for real-time operation. Industrial applications of AI models face strict constraints on model size due to multitasking, run times, and cost considerations. Reducing model sizes also decreases computational burden, making it feasible to run these models on resource-limited computing devices.

Building on the previous research on AC arc fault detection algorithms and lightweight models such as the Efficient-ArcNet [18], this chapter proposes a lightweight algorithm, referred to as PADNet, capable of reliably detecting series arc faults in PV (DC) systems with high accuracy and efficiency. The primary objective of this research is to embed the AI model into an edge device with limited computational power compared to devices like the Raspberry Pi, ensuring real-time operation and integrating an SSCB for rapid arc fault interruption. The entire process, including data acquisition, min-max normalization, frequency domain conversion using FFT, AI model inference, and trip signal generation for arc faults, is performed within an STM32 MCU. Data acquisition for one sample takes a total of 12.8 ms. The main contributions of this research are:

1. The development of a lightweight and computationally efficient CNN for accurate detection of series PV arc faults, achieving a high accuracy of 98.15%. The network

architecture employs pointwise and depthwise convolution layers, along with strategic convolution and Max-Pooling layers and a carefully designed bottleneck structure, to optimize accuracy. The model is highly effective in detecting series arc faults in PV systems.

2. The introduction of a novel approach using a solid-state circuit breaker based on MOSFETs to promptly interrupt the circuit during arc fault events, enhancing the operation speed of the intelligent AFCI and effectively protecting the system from potential fire hazards.

3. Optimization of the proposed PADNet model using the TensorFlow-Lite tool to reduce binary size and latency, enabling implementation in resource-limited edge devices. Real-time performance is verified using a commercial Microcontroller Unit (MCU), the STM32H743ZI2, which features a 32-bit Arm Cortex architecture with 2 MB of internal memory and 1 MB of internal RAM. The model demonstrates real-time operational capabilities for commercial use, with an estimated inference time per sample of 12.2 ms on the STM32H743ZI2. The SSCB takes only 33 microseconds to break the circuit during an arc fault, resulting in a total arc fault clearing time of just 25 ms, including data acquisition, preprocessing, model evaluation, and circuit tripping.

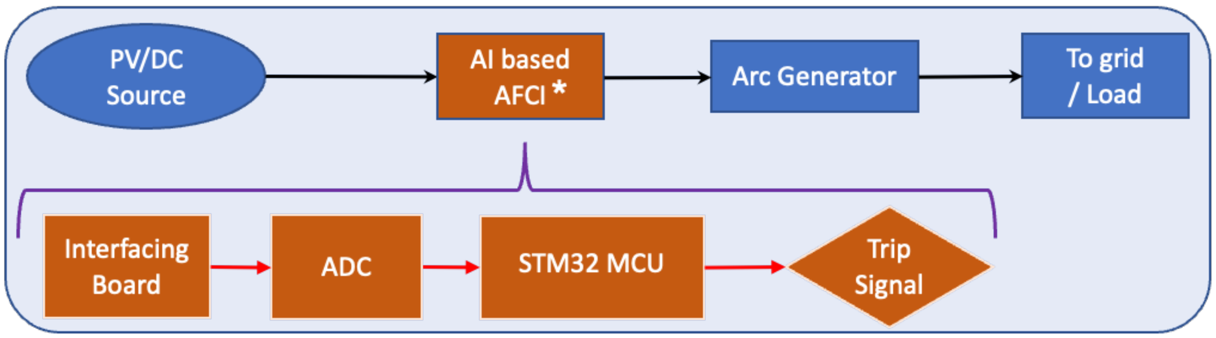


Figure 5.1: Block diagram of PV arc fault detection system.

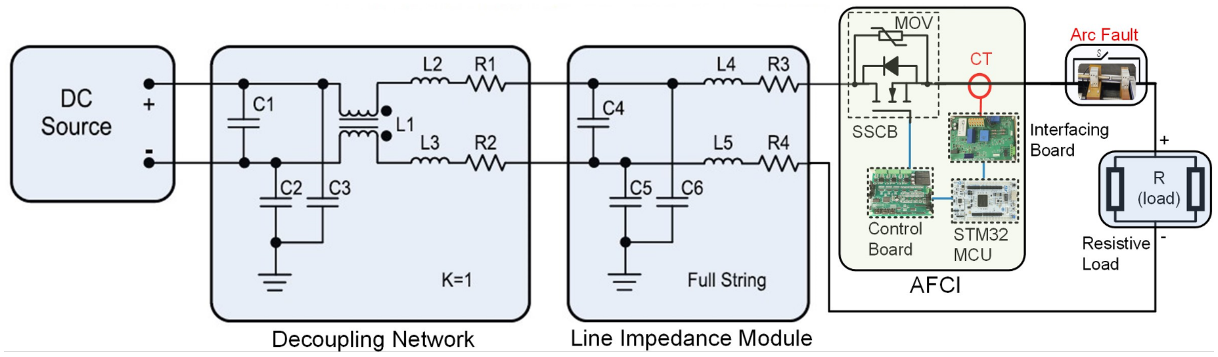


Figure 5.2: Schematic diagram of PV arc fault data collection test bench.

5.1 Data Collection and Analysis

5.1.1 Experimental Platform of PV Arc Fault Generation

Collecting PV arc fault data from a real PV system is inconvenient and dangerous. On the other hand, model-based PV arc fault data is impractical to use with the real system. Following the UL1699B standard, a test platform has been designed for PV arc fault data collection in a lab environment, as shown in Figure 5.2 (decoupling and impedance module are omitted in the setup). The test bed is built using a programmable PV simulator (DC source), an arc fault generator, a grid-tied inverter (GTI), and the AC grid. The PV simulator is of 10kW, 1000V, and 10A rating (Model ETS100X10C-PVF). For testing the experiment, resistors and GTI with grid were used as load, respectively. The AFCI was not included during data collection. However, for testing, the AFCI was incorporated, which includes an interfacing board to sense the load current, sending the current signal to the STM32 microcontroller. The STM32 MCU has a built-in analog-to-digital (ADC) converter to read those input data. The full block diagram of the AFCI system is provided in Fig. 5.1.

The irradiance profile and temperature can be varied in the program of the PV simulator. According to the UL1699B-2018 standard, the PV string can be replaced with a PV simulator for the experiment. The used SMA brand PV inverter is of 5 kW capacity and grid-tied type. The output terminal of the inverter has an L1-N-L2 configuration.

Therefore, L1 and L2 terminals are connected to the two phases of the 208V, 3-phase 4-wire utility grid, and the N terminal is connected to the neutral wire of the grid. The inverter is furnished with built-in AFCI functionality. During the data collection operation, the AFCI functionality was turned off. The inverter output was connected to the utility grid (AC). A detailed description and rating of different components are provided in Table 5.1. The test-bed is one string, single module, and one MPPT-based PV arc fault generation and detection system.

Table 5.1: Specifications of the components used in the PV arc fault test-bed setup.

Equipment	**Model**	**Ratings**
PV Simulator	Chroma32180D-1800	18 kW, 1.8 kV, 40A
PV Inverter	SB5.0-1SP-US-40	5kW 208/240VAC GTI *
AC Source	-	208V, 3-Phase
AC Grid (Load)	-	208V, 3-Phase
Mixed Signal Oscilloscope	Tektronix MSO58	500 MHz, 6.25 GS/s
Resistive Load	-	(10-32) Ohms, 1 kW

*GTI stands for Grid Tied Inverter

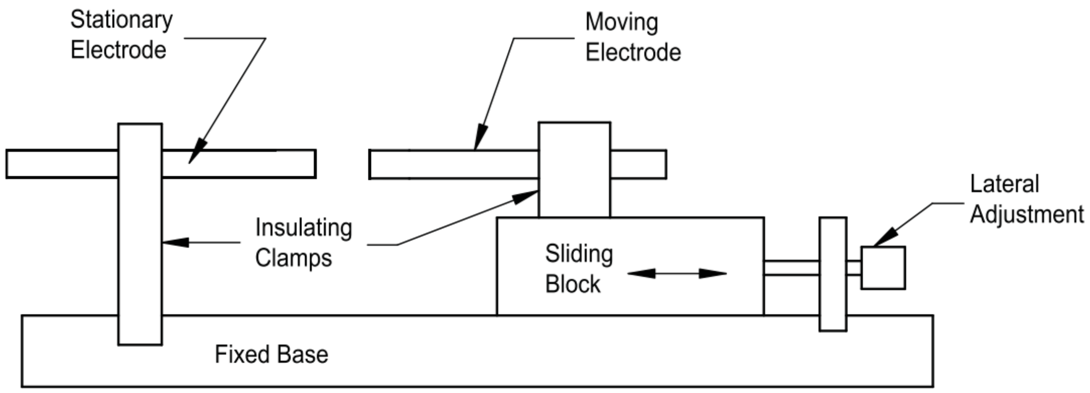


Figure 5.3: Architectural diagram of arc fault generator.

The arc generator is built using a stationary electrode made of copper and the moving electrode which sits on top of a sliding block is made of carbon graphite rod. Both of the electrodes have 1/4 inch diameter. The lateral adjustment of the sliding block is performed using a programmable motor controller attached to the sliding block. The maximum current rating of the arc generator is 20A. The architectural diagram of the arc fault generator is presented in Fig. 5.3. During the arcing experiment, a sustainable arc makes the tips of the electrodes to be dirty, which is cleaned periodically using sandpaper. The arc generator is furnished with a parallel switch, S. Closing that switch, normal load current data were collected.

5.1.2 Operating Conditions of Arc Fault Experiments

UL1699B-2018 standard specified different conditions of PV AFCI tests that have to be followed. According to the standard, a number of arc fault tests were performed to collect PV arc fault data at different operating conditions. This helps to make the algorithm more robust. Arc faults were generated at the start of the string as well as at the end of the string as shown in Fig. 5.2 (one configuration at a time). A stepper motor was used to create an arc gap between the arcing electrodes at a constant speed of 5mm/s to generate arc fault. The data were collected at different voltage and current levels. The current range was (3∼4) A and the voltage range was (480∼480) V. Table 5.2 depicts various arcing tests performed under different levels of maximum power point voltages (V_{mpp}) and maximum power point currents (I_{mpp}). Minimum arc currents were always higher than I_{arc} that is specified in the Table. It also shows the arc gap distances and number of samples collected under various testing conditions. From zero arc gap distance, the arcing electrodes are separated using the stepper motor at a speed of 5 mm/s to generate arc fault. The data sampling rate was 250 KS/s. A mixed signal oscilloscope (MSO58) was used to record and collect data. A current probe was used to sense the arcing current. The voltage across the arc gap was also recorded using a differential voltage probe. The arc gap voltage was recorded to verify the status of the arc and normal current during data collection. For two arcing tests, 7000 samples of arc and 7058 samples of normal current were collected. Total samples were 14058, each of them having 10.8 ms in length.

Table 5.2: Different operating conditions of PV arc fault data.

Test No.	**Description**	**Min I_{arc} (A)**	I_{mpp} **(A)**	**Sep. Rate (mm/s)**	V_{mpp} **(V)**	**Voc (V)**	**Gap (mm)**	**Arc samples**	**Nromal Samples**	**Total**
1	Arcing test	2.5	3	2.5	312	480	0.8	3500	3500	7000
2	Arcing test	3	4	5	318	490	0.8	3529	3529	7058
							Total samples	7029	7029	**14058**

5.1.3 Data Visualization, Preprocessing and Analysis

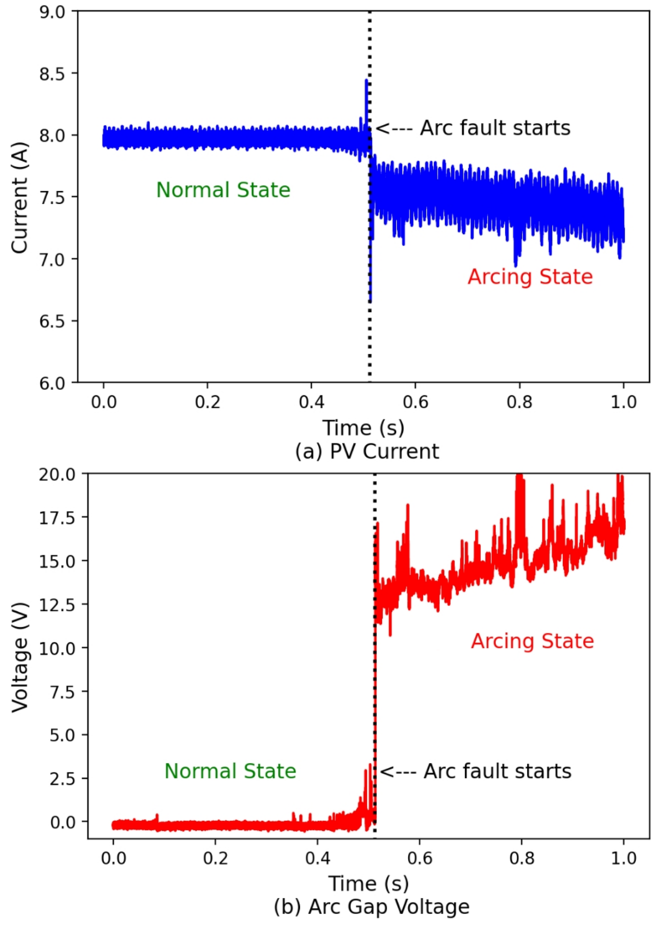


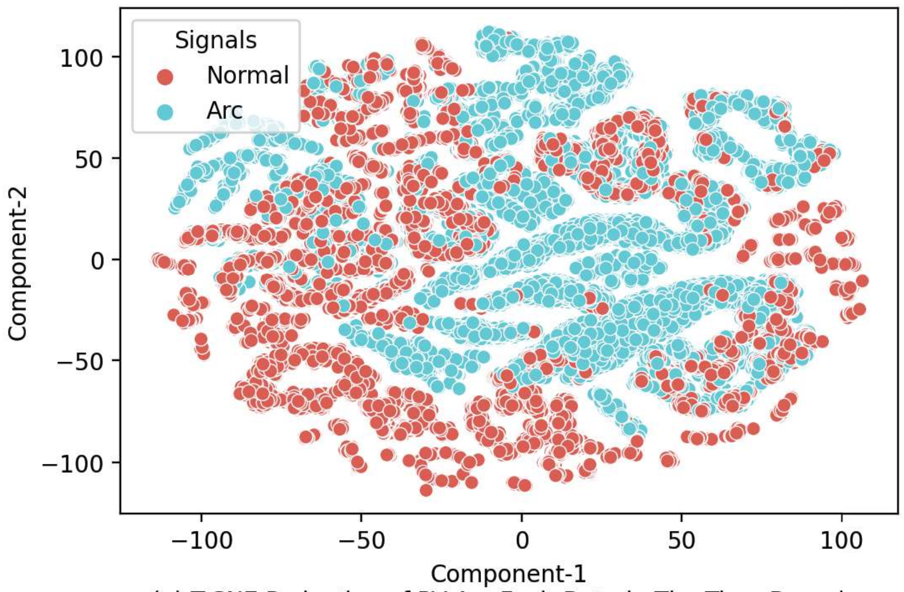
Figure 5.4: Graph of (a) PV system current and (b) arc voltage across the arc gap.

During data collection, both the arc current and arc voltage information were captured and recorded using an oscilloscope. A zero voltage across the arc gap indicated no arc. When there was an arc present in the current, the arc gap voltage increased a considerable amount (about 5 to 30 V). Fig. 5.4 depicts the normal and arcing current of 8 A, 490 V (Vmpp) condition of the PV system. Before the arcing event, the arc gap current is zero. When arcing occurs at 0.52 s, the arc gap voltage is increased above 12.5 V. Each segment of recorded data was 10 s in length. Sample length and sampling frequency play vital roles in preserving feature information. A large sampling frequency and a larger sample size even though hold good feature information, they make the input size large and make the computation higher for data preprocessing as well as in the computation inside the neural network. Nevertheless, the sample length should be sufficient to be able to be tested in a short time for the identification of a possible arcing fault. A larger sample size may lead to missing an important sample in the real-time testing environment which could be an arcing fault. Different research have used different length of samples ranging from 2ms to 50ms [30, 32]. A shorter sample size such as 2 ms may yield poor accuracy. Considering the computational time, feature information, and accuracy in arc fault detection, a sample length of 10 ms is considered in this research. A total of 14058 samples were collected using the aforementioned experimental setup.

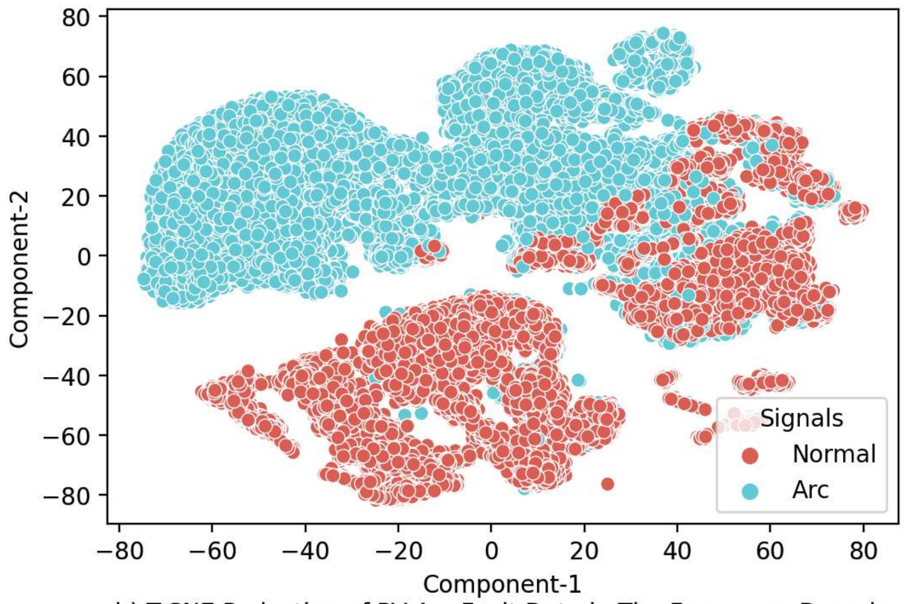
Before preprocessing the data, they are visualized and analyzed using the popular t-SNE (T-distributed Stochastic Neighbor Embedding) plot. T-SNE, a non-deterministic tool based on the stochastic neighbor embedding method [124], is a nonlinear dimensionality reduction technique used to visualize high-dimensional data in two or three-dimensional space. Fig. 5.5 (a) shows the t-SNE projection of the training database in the time domain. The plot reveals that the data samples are so scattered that a simpler decision boundary cannot effectively separate arc and normal samples.

To address this, the fast Fourier transform (FFT) is applied to the data, and the t-SNE visualization of the training data is shown in Fig. 5.5 (b). The FFT spectrum arranges features of the data more conveniently, extracting useful features for classification. However, many arc samples still lie within the normal region and vice versa. Therefore, instead of using raw samples for the model, FFT data is used for the training and testing of the proposed model. It is important to note that after applying FFT, each sample is normalized using the Min-Max normalization technique following the equation given below.

$$X_{norm}(i) = \frac{X(i) - X_{min}(i)}{X_{min}(i) - X_{Max}(i)} \tag{5.1}$$



(a) T-SNE Projection of PV Arc Fault Data in The Time Domain.



b) T-SNE Projection of PV Arc Fault Data in The Frequency Domain.

Figure 5.5: t-SNE plots of arc fault data in the (a) time domain and (b) frequency domain. Feature extraction is more effective in the frequency domain compared to the time domain.

where $X_{norm}(i)$ indicates the normalized data of i^{th} sample. $X_{min}(i)$ and $X_{max}(i)$, respectively, denote the smallest and largest values of the same data sample.

The final data sample has a frequency spectrum of 0 to 125 kHz. The whole database is split at the ratio of 70 : 10: 20 to get the training, validation, and testing samples respectively. The training dataset is used to train the algorithm. The validation dataset is used as an unknown dataset to the model while training is ongoing. It also helps to determine how efficiently the algorithm is learning the features from the training data. At the same time, it also helps to find a suitable termination point before the model is overfitting on the training data. Testing samples are used to test the algorithm after the training is complete. Arc fault detection is a binary classification problem- arc and normal signal. Therefore, for binary classification, the arcing samples are labeled as 1 and the normal current samples are labeled as 0.

One random sample from each of the 3 A, 5 A, and 8 A loads (arc and normal) is chosen from the database for visualization. Their FFT spectrums are plotted in Fig. 2.1. It is noticed that even though within 0 to 40 kHz frequency range most of the samples showed differences in the frequency spectrum, for the later portion some of them overlapped. Sometimes, due to various noise conditions, an arc sample may look like a normal sample or vice versa. In Fig. 5.6, a normal sample of 5 A load is mostly overlapped with the arcing sample of 8A. Therefore, classification of arc fault using only the hand-adjusted threshold of the FFT spectrum may lead to poor accuracy and hence false triggering may occur.

5.2 Algorithm Development using Lightweight CNN Approach

Based on the analysis in Section 5.1.3, it is evident that Fast Fourier Transformation (FFT) generates good initial features from the current signal. Nevertheless, the data in the frequency spectrum is complex, making it difficult to segregate arc samples from normal samples using a simple hand-adjusted threshold with good accuracy. It requires a special algorithm that can extract more concrete features from the frequency domain current signal and classify it more accurately. CNN [125] is an artificial intelligence-based deep learning algorithm that, when designed properly, can automatically extract important feature information from an input signal. At the same time, from the extracted features, it can distinguish and classify the input signals at its output layer with high accuracy. In this

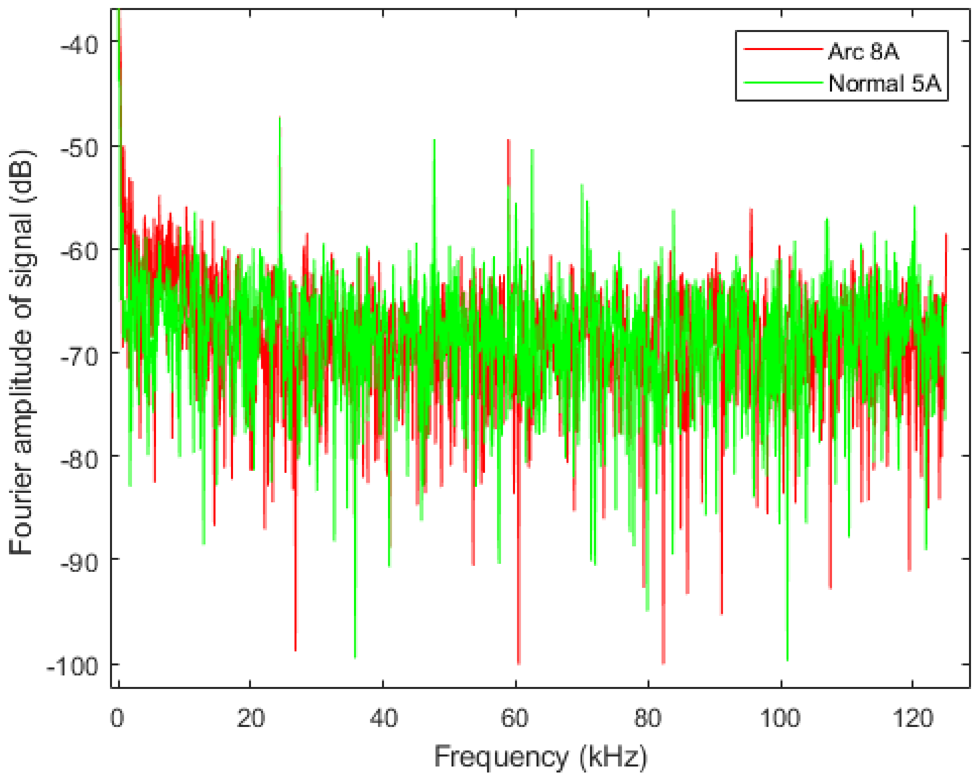


Figure 5.6: Overlapping FFT spectrum of an arc and a normal current.

case, the supervised learning method is used where, during the training of the algorithm, pre-labeled data samples are fed to the algorithm to learn feature vectors from them. The learned model is then tested with a separate test dataset, which is non-inclusive to the training set. The CNN-based method also helps to avoid hand-adjusted threshold-related problems.

In this research, the frequency domain features extracted from the FFT signals are used to feed a convolutional neural network to further extract features and finally classify the arc and non-arc signals. However, the use of traditional CNN blocks in designing an algorithm incurs a lot of computation, which can hinder the implementation of the model in commercial MCUs. Inspired by the efficient network architecture of EffNet-ArcNet [18], a lightweight and efficient CNN-based model has been built. The data fed to the network is a one-dimensional vector of the FFT spectrum. The image of the FFT spectrum is not considered as input to the model, as it adds more computation compared to a 1D vector as input.

5.2.1 Model Building Block

Due to various factors such as competitive costs, combined multitasking, and interactive run-times, there are strict limitations on the size of AI-based models when they are used for industrial applications. Thus, if the computational cost can be reduced, it could allow the redistribution of computational power to critical areas where it is required. Alternatively, it can enable the use of deeper or wider networks to complete tasks with increased capacity.

Instead of using conventional CNN blocks to build a model, a structurally efficient CNN building block is used, as shown in Fig. 5.7. The conventional CNN block has been replaced by a pointwise (1x1) convolution with half of the channels (ch/2). Then, a depthwise (dw) convolution of kernel size 5x1 is applied, followed by a 1D Max Pooling (MP) layer of size 2x1. Finally, a sub-layer consisting of the full channels (ch) with kernel size 2x1 followed by a 1D stride of 2 is applied.

A Max Pooling layer is applied to the depthwise layer instead of the spatial convolution layer, allowing the network to encode the data before a drastic reduction. Since early pooling yields cheaper consecutive layers, separable pooling has been used to take advantage of early pooling. Instead of conventional convolution, depthwise convolution has been utilized, which, when combined with the pooling strategy, achieves a significant advantage in computation.

5.2.2 Network Architecture of PADNet

Using the building block discussed in Section 5.2.1, the PV arc fault detection model has been designed. The network architecture of the proposed PADNet model is based on multiple building blocks of the model, as shown in Fig. 5.7. The complete network architecture of the PADNet model is illustrated graphically in Fig. 5.8. It has three basic building blocks of PADNet (as shown in Fig. 5.7) with 32 filters in each block. Finally, three fully connected (FC) layers with 16, 8, and 2 neurons are annexed. The last layer with 2 neurons is the output layer, having two outputs (arc and normal). The basic building block replaces the conventional CNN layer and makes the model lightweight and efficient.

The data flow of the proposed model at different stages of the network is described in Table 5.3. The data flow of a baseline CNN model corresponding to PADNet layers is also compared in that table. Except in the first layer of the Baseline CNN, other layers have a higher number of trainable parameters compared to the PADNet architecture. It is noteworthy that a maximum bottleneck factor of 2 has been used in the architecture (shown in red color). Additionally, a bottleneck factor of 1.5 (blue) is observed in some cases throughout the structure. A large bottleneck factor tends to reduce accuracy. Unlike other lightweight network architectures, such as ShuffleNet, MobileNet, or SqueezeNet, where a higher bottleneck factor ranging from 4 to 8 is observed [35, 36, 126], a moderate bottleneck factor (within 2) has been maintained. Furthermore, the pointwise spatial convolution layer has been applied in the very first layer, reducing the computation by approximately 30% in the subsequent layers.

The activation function in all convolution and fully connected layers except the output layer is Rectified Linear Unit (ReLU). In the output layer, the 'sigmoid' activation function has been used.

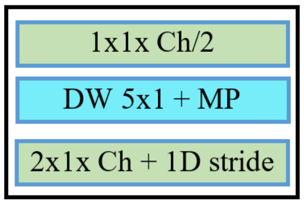


Figure 5.7: PADNet layer building block replacing traditional CNN layers. 'DW' stands for depthwise convolution, 'MP' for max pooling, and 'Ch' for output channels. Strides are set to 2.

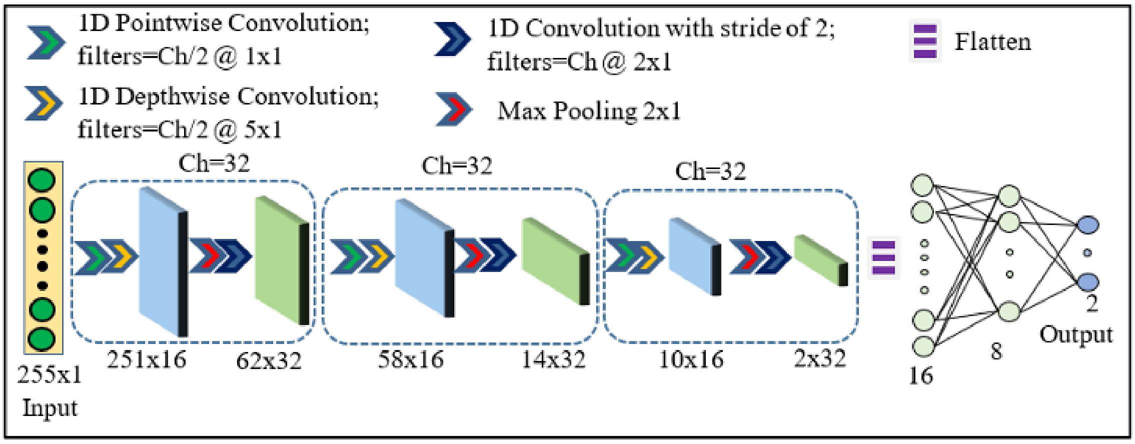

1D Pointwise Convolution; filters=Ch/2 @ 1x1
1D Depthwise Convolution; filters=Ch/2 @ 5x1
1D Convolution with stride of 2; filters=Ch @ 2x1
Max Pooling 2x1
Flatten
Ch=32
Ch=32
Ch=32
255x1 Input
251x16
62x32
58x16
14x32
10x16
2x32
16
8
2
Output

Figure 5.8: Network architecture of PADNet with three consecutive PADNet building blocks each containing 32 filters, followed by three fully connected layers with 16, 8, and 2 neurons respectively. The output layer has 2 neurons.

Table 5.3: Data flow of PADNet model compared to conventional baseline CNN model. FC (16, 8, 2) refers to fully connected layers with 16, 8, and 2 neurons, respectively. The last FC layer has 2 neurons for 2 output classes (arc and normal current). Depthwise convolution and max pooling layers are denoted by 'dw' and 'mp'.

Baseline CNN		PADNet (Ours)	
Layers	Params	Layers	Params
5x1x32 + mp of 2	192	1x1x16	32
		dw 5x1 + mp of 2	352
		2x1x32 + stride of 2	1056
5x1x32 + mp of 2	5152	1x1x16	528
		dw 5x1 + mp of 2	352
		2x1x32 + stride of 2	1056
5x1x32 + mp of 2	5152	1x1x16	528
		dw 5x1 + mp of 2	352
		2x1x32 + stride of 2	1056
FC (16, 8, 2)	77994	FC (16, 8, 2)	1194
Total Params	88,490		6,506

5.3 Experimental Setup, Results and Analysis

This section summarizes the experimental result of the PADNet model, model optimization, and analyzes the misclassified samples.

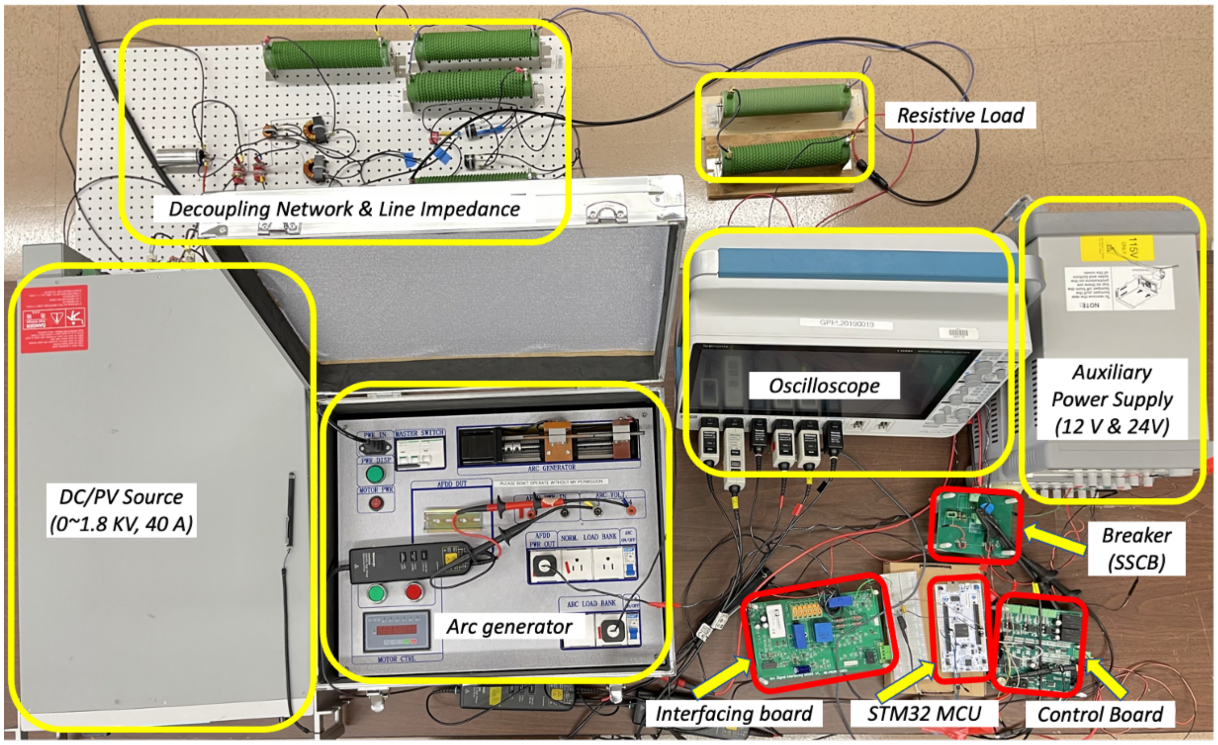

Decoupling Network & Line Impedance
Resistive Load
DC/PV Source (0~1.8 KV, 40 A)
Arc generator
Oscilloscope
Auxiliary Power Supply (12 V & 24V)
Breaker (SSCB)
Interfacing board
STM32 MCU
Control Board

Figure 5.9: Experimental setup for PV arc fault data collection and real-time testing.

5.3.1 Experimental Setup and Results of The Proposed Model

The PADNet algorithm is developed using the TensorFlow Keras platform. The proposed model is trained using all the train samples for 80 epochs with a batch size of 300. Adam optimizer with a learning rate of 0.001 has been used for training. During training, the training accuracy and validation accuracy plots of the model are checked for any possible overfitting. Fig. 5.10 shows the training and validation accuracy plots. From the learning curves, it is observed that the training accuracy continues to improve till the end. However, the validation accuracy increased to a certain point and then became flat without getting worse. The two curves closely followed each other. Hence the model is not overfitted.

To compare the result of PADNet, a baseline CNN model was trained using conventional CNN blocks but with a similar number of channels in each corresponding layer. In this case, the model was trained for 45 epochs with a batch size of 300. The Adam optimizer with an adaptive learning rate was used during training. When the traditional CNN block was used, the model showed a maximum accuracy of 98.89% in classifying arc and normal current. However, the PADNet algorithm showed an increased accuracy of 99.02% compared to the conventional model. In terms of model parameters, PADNet has only 14.698 K trainable parameters, which is an 83.39% reduction compared to the traditional CNN-based model.

The actual size of the models also indicates that the efficient PADNet model has a smaller size. When both models were implemented without any compression, the CNN model required a Flash Memory size of 367.1 KB, which is larger than that of the PADNet model, given that the RAM requirement is similar. On the other hand, because PADNet is constructed using a lightweight and efficient model architecture, it took only 83.08 KB of Flash and 85.62 KB of RAM to run in the MCU (without any model compression). The results are summarized in Table 5.4. The actual experimental platform of the PV arc fault detection system is shown in Fig. 5.9. Different components of the experimental setup are properly labeled in the diagram.

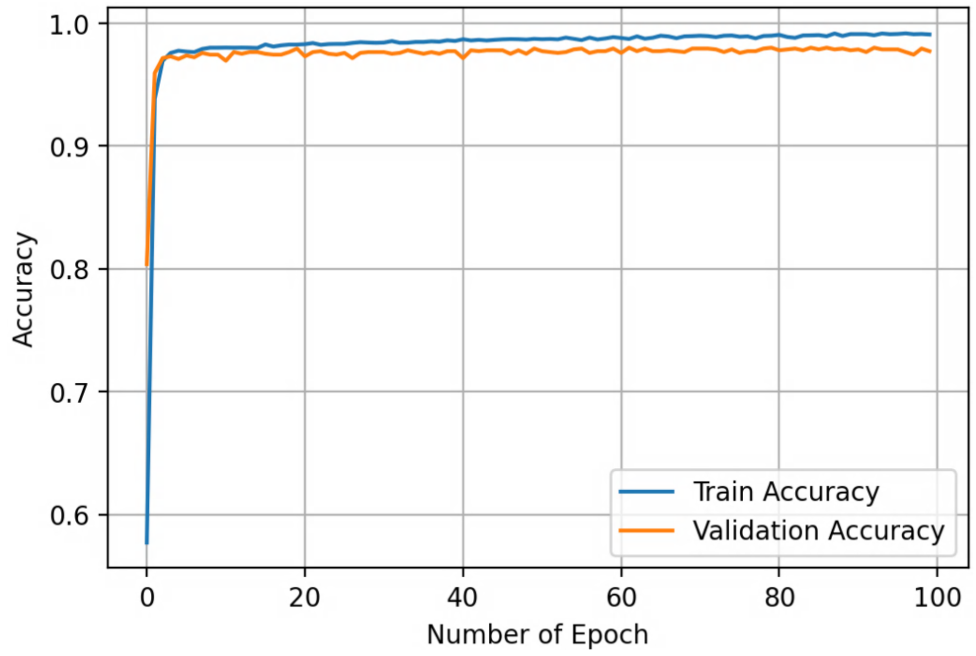


Figure 5.10: Training and validation accuracy plots of PADNet model.

Table 5.4: Results of PADNet model

Model	**Accuracy (%)**	**Params**	**Actual Size**	**kFLOPs**	**Factor**	**Reqd. Flash[2](KB)**	**Reqd. RAM[2](KB)**	**Inference Time (ms)**
CNN	98.89	88.490 K	1097 KB	10144.6	1	367.1	81.99	77.613
PADNet	98.15	14.698 K	290 KB	2464.84	0.24	83.08	85.62	28.13
CNN[1] (Optimized)	98.89	-	353 KB	-	-	366.60	164.75	57.14
PADNet[1] (Optimized)	98.15	-	73 KB	-	-	83.49	163.59	12.2 (25)[3]

[1] Optimized using TensorFlow Lite (TF-Lite) optimization tool.

[2] RAM and Flash memory required by STM32H743ZI2 MCU.

[3] Total fault clearing time using STM32H743ZI2 MCU and SSCB.

Table 5.5: Precision, recall, and overall accuracy of PADNet model.

Actual Class		Predicted Class		Total
		Normal	Arc	
	Normal	1380	4	1384
	Arc	48	1379	1427
	Total	1428	1383	2811
		Precision	99.71%	
		Recall	96.64%	
	Overall accuracy		98.15%	

From the test results of PADNet, a confusion matrix was produced as shown in Table 5.5. It indicates that out of 1384 normal samples 1380 samples were correctly classified. The classification accuracy of the normal sample is 99.71%. Only 4 samples were misclassified as an arc. Conversely, out of 1427 arcing samples, 3.3% samples (48) were misclassified as normal samples. Only arc classification accuracy is 96.7%. In machine learning-based classification problems, precision and recall are considered important performance metrics. A model is considered more reliable when its values are closer to 1. The precision and recall of the PADNet model are 99.71% and 96.64%, respectively. Those values are very high, which indicates that the model yields good performance. It also means that the false positive, as well as the false negative rates of the model, are very low. Overall classification accuracy is 98.15%.

5.3.2 4-Fold Cross Validation

Cross-validation is a statistical technique used to evaluate machine learning models by training several models on a subset of the whole data and testing with the complementary subset of the data. A k-fold cross-validation involves training the algorithm with all but one of the k subsets and evaluating on the subset that is not used for training. This method is repeated k times, with each iteration using a different subset of the data as the test dataset. Cross-validation is used to detect model overfitting and to check the robustness of the model.

A 4-fold cross-validation using the PADNet model has been performed. The whole dataset is grouped into four subsets. Each time, three subsets are used for training the algorithm and the fourth subset is used for testing. The arc fault detection accuracies are recorded in each iteration by evaluating the model on the corresponding test subset. The 4-fold cross-validation results are tabulated in Table 5.6. After 4-fold cross-validation, the average accuracy of 98.07% is determined, which is close to the reported accuracy (98.15%) of the original PADNet model.

Table 5.6: 4-fold cross validation of PADNet model

k-Fold Iteration	**Model Accuracy (%)**
1	98.09
2	98.05
3	97.99
4	98.14
Average	98.07

5.3.3 Real-time Testing of AFCI using Microcontrollers and SSCB

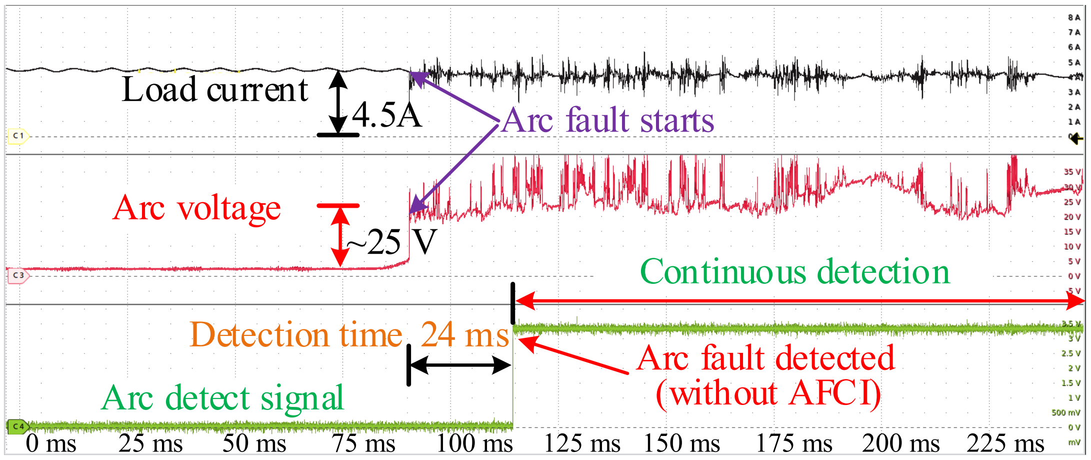


Figure 5.11: Real-time arc fault detection (without AFCI; C4 = 10 μF).

The proposed PADNet model can be used for commercial applications using embedded microprocessors (STM32 microcontrollers). This is because the model is lightweight

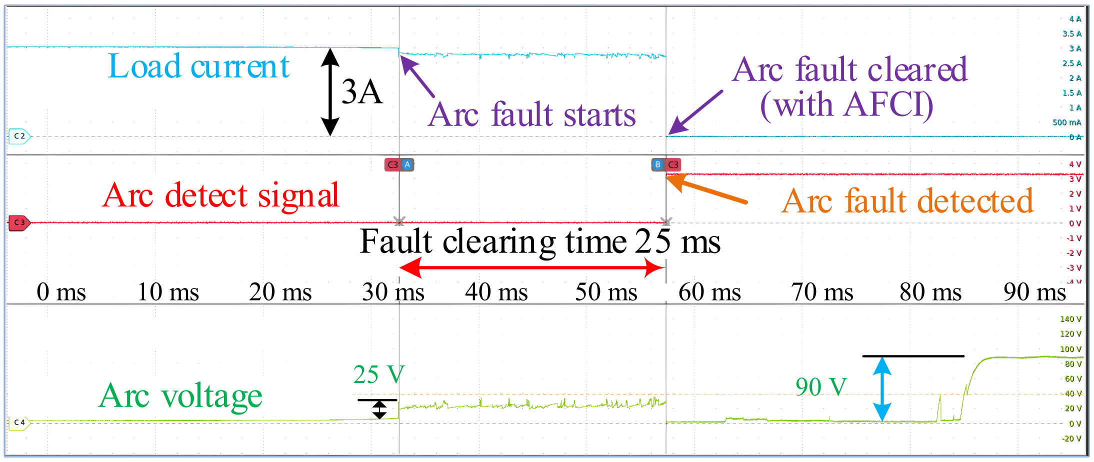


Figure 5.12: Real-time testing of AI-based AFCI (fault clearing time; C4 = 150 nF).

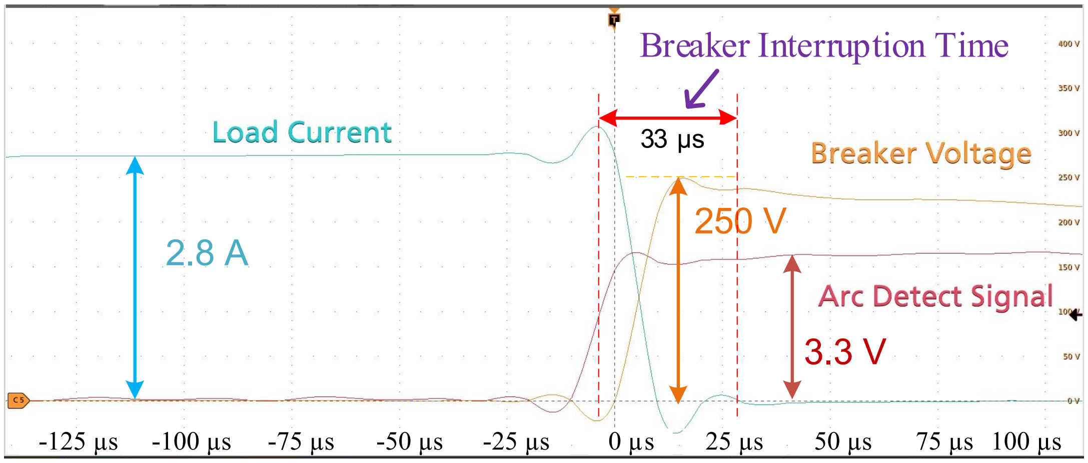


Figure 5.13: Breaker waveforms for AFCI operation.

and has less computational burden. The network architecture is built using an efficient model building technique of Deep Learning. Furthermore, instead of using a 2D image as input, a 1D vector of the FFT spectrum with a frequency range of 40 kHz has been used. Each sample having a 100 kHz frequency spectrum could be huge. Therefore, an FFT spectrum of 40 kHz has been used before feeding the input to the AI model. The STM32 NUCLEO-H743ZI2 development board has been used for embedded implementation. It has an STM 32-bit Arm Cortex microcontroller unit STM32H743ZI2 on the board. The STM32H743ZI2 MCU has a flash memory size of 2 MB and a RAM of 1 MB. Before embedding the model in the STM32H743ZI2 MCU, it has been optimized using the

TensorFlow Lite optimization tool for reduced binary size and latency. The PADNet model has been tested in the ST microcontroller for all the test samples, achieving an arc fault clearing time of only 25 ms. This time includes the data acquisition time of 12.8 ms and inference time of 12.2 ms. This experiment demonstrated the feasibility of the proposed model to embed in a resource-limited commercial embedded microcontroller.

The performance of the proposed AI-based AFCI has been extensively tested under various operating conditions for normal operation and arc fault. Fig. 5.11 demonstrates the real-time detection of an arc fault using the proposed PADNet model without the use of an AFCI or an SSCB. The value of the capacitor C4 was set to 10 μF in this part of the experiment. This test evaluates the algorithm's capability to continuously detect arc fault occurrences. The oscilloscope capture includes three critical waveforms: the load current (black); the arc gap voltage (red), which exhibits significant voltage (25 V) indicating the occurrence of arc fault; and the arc detect signal (green), which activates within 24 milliseconds of the fault initiation and remains active, indicating continuous fault detection. The results highlight the PADNet model's effectiveness in real-time arc fault detection, ensuring continuous monitoring and providing a crucial safety measure.

The snapshot of the fault-clearing time during real-time operation is shown in Fig. 5.12. The AFCI was tested under a loading condition of 3.10 A with an input voltage of 330 V (DC). The value of the capacitor C4 was set to 150 ηF in this experiment. The total time required to clear the arc fault was 25 ms. Notably, the breaker took only 33 μs to break the circuit, as depicted in Fig. 5.13. It is important to note that the AFCI was tested using both resistive loads and a grid-tied PV inverter.

5.3.4 Visualization and Analysis of Misclassified Samples

The proposed PADNet model has achieved more than 98% arc fault detection accuracy, but some samples are misclassified. To understand why samples are misclassified, a deeper analysis of these samples has been conducted. A gradient class activation map (grad-CAM) tool and a statistical analysis method were used to analyze a few misclassified samples [127]. In deep models, as the network goes deeper, activations become more complex and abstract, with deeper layers encoding very high-level feature vectors. Final CNN layers often show that many filters are not activated, as the model has already extracted the necessary feature vectors for classification. Using this concept, grad-CAM images of the last CNN layer of PADNet were evaluated.

Fig. 5.14 shows the output of the last CNN layer for three samples. There are 32 filters in the last CNN layer, as indicated by the red square in the figure. Fig. 5.14 (a) is the activation filter map (grad-CAM) of a sample that is originally an arc sample but is misclassified as normal. Fig. 5.14 (b) is the correctly classified normal sample, and Fig. 5.14 (c) is the activation map for the correctly classified arc sample. A closer look at the figures reveals that the grad-CAM output of Fig. 5.14 (b) and (c) shows considerable differences in the activation of filters as the two samples are of two opposite classes (normal and arc). However, Fig. 5.14 (a) and (b) show more similarity in the location of the activated filters. Even though Fig. 5.14 (a) is originally an arc sample, it has many similarities to a normal sample as shown in Fig. 5.14 (b). Therefore, it was misclassified as a normal sample by the model.

Furthermore, the FFT spectrum of the first two samples mentioned in Fig. 5.14 was checked and found to be almost overlapped throughout the entire range (Fig. 5.15). Pearson's correlation between the two samples was determined and found to have 88% similarity. Therefore, it can be concluded that there are a few samples in the database with similar feature vectors to those of the opposite class, causing misclassification. Additionally, there might be other samples that lie inside the region of the opposite class in the feature map, resulting in misclassification. It was evident from the t-SNE plots shown in Fig. 5.5 that the data distribution is so complex that a very complex decision boundary is required to classify them accurately, which in turn requires a more complex algorithm and more computation. Examining this is beyond the scope of this paper.

When misclassification happens, the arc sample looks like a normal sample and vice versa. The Pearson's correlation of a misclassified sample is compared in Fig. 5.15, where an actual normal sample is compared with an arc sample classified as normal.

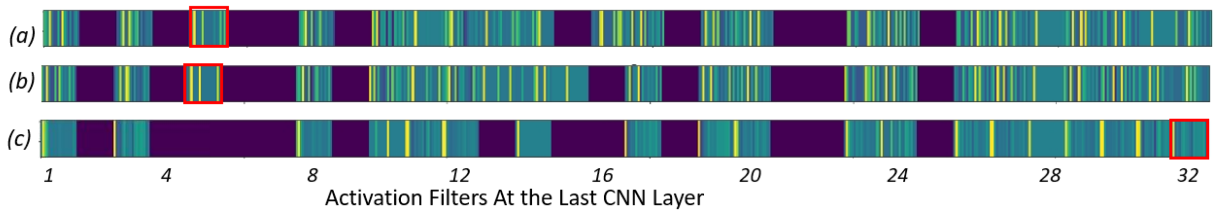


Figure 5.14: Grad-CAM output of the last CNN layer for (a) an arc sample misclassified as normal (top), (b) a correctly classified normal sample (middle), and (c) a correctly classified arc sample (bottom). Each red square represents an individual filter. There are 32 filters in the last CNN layer. Darker regions indicate inactive filters, while colored regions indicate activated filters.

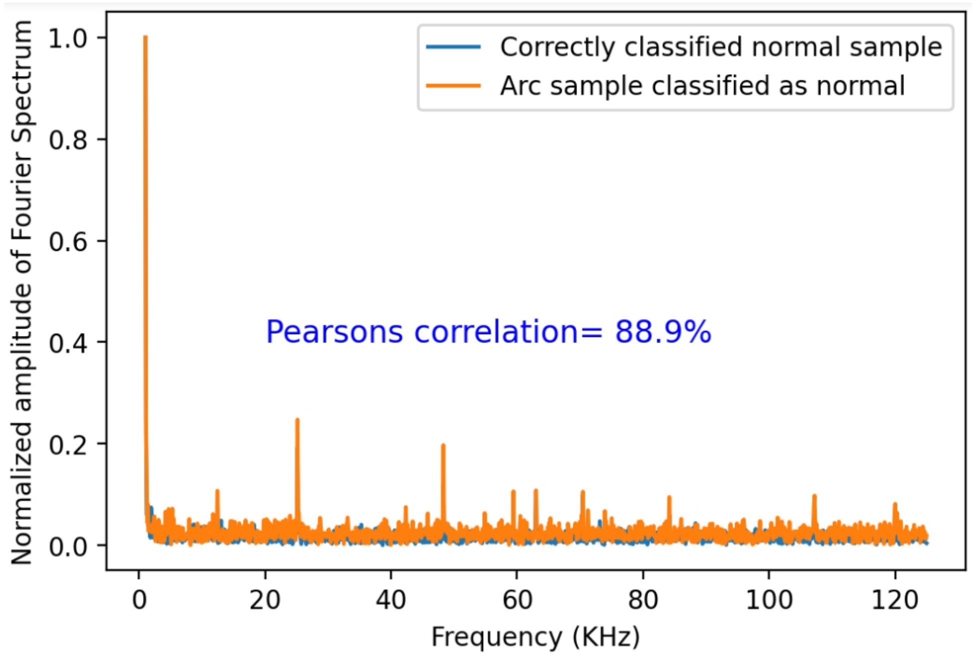


Figure 5.15: Comparison of a Misclassified Normal Sample with an Actual Normal Sample Which is Classified Correctly. The Pearson's Correlation of the two samples is 0.889.

5.4 Comparison of PADNet with Different Methods

In order to demonstrate the performance of the proposed PADNet model, it has been compared to other PV arc fault detection algorithms. The methods have been

Table 5.7: Comparison of PADNet with other methods

Algorithm	**PADNet (Ours)**	**Cai et al. [64]**	**Lu et al. [67]**	**Aziz et al. [128]**	**Miao et al. [61]**
Feature Extraction Method	FFT + Lightweight CNN	PSO-SVM	DA-DCGAN	2D CNN	WPD + RF
Accuracy (%)	98.15	99.20	99.4	70.53	86.36
Test Platform	STM32H743ZI2 MCU	i7-7700 CPU, RTX2070SUPER GPU and 8GRAM	NI-CompactRIO-9030	High performance computing platform	High end computer
Sample length	12.8 ms	13 ms	20 ms	-	10 ms
Sampling frequency	100 kHZ	250 kHz	20 kHz	500 kHz	250 kHz
Inference time	12.2 ms	138 ms	60 ms	4.3 ms	10 ms
Are different fault locations considered?	Yes	Yes	Yes	No	Yes
Arc fault interruption in real-time?	Yes (SSCB based)	No	No	No	No

compared based on different criteria, including classification algorithm, accuracy, platform of verification, detection time, fault location, and sampling frequency of the database, as shown in Table 5.7.

Reference [64] used variational modal decomposition to extract features and then used particle swarm optimization and support vector machine algorithm for classification. Using a high-end computing machine the inference time is reported as 138 ms per sample. In [67], domain adaptation with a deep convolutional generative adversarial network (DA-DCGAN) based method has been used for arc fault detection. The sampling frequency is very low (20 kHz). Compared to PADNet it has higher detection runtime. CNN in combination with Long Short Term Memory (LSTM) based detection algorithm has been proposed in C. However, a high-end computing platform has been used to report the detection runtime. Reference D has used the wavelet packet decomposition (WPD) based method, however, the test accuracy is very low (86.36%). Compared to other methods, PADNet is lightweight as well as it has high arc fault detection accuracy. The proposed method is also tested using commercial MCUs for performance evaluation and showed a runtime of only 23.38 ms per sample. Also, compared to traditional CNN-based algorithms, the model used lightweight CNN architecture which yields a faster inference time and smaller binary size.

5.5 Summary

This research presents a real-time closed-loop system for an AFCI designed for PV (DC) systems. An artificial intelligence-based algorithm has been employed for arc fault detection and a solid-state circuit breaker to interrupt the circuit during arc fault events. An experimental platform was established following the UL1699B-2018 standard to systematically collect PV arc fault data.

A lightweight and efficient algorithm, PADNet, has been proposed for series DC arc fault detection based on the FFT spectrum of the current signal and an efficient CNN model. The model exhibits remarkable accuracy in arc fault classification with over 80% less computational burden than conventional CNN block-based models. Additionally, it requires only 22.84 kB of RAM and 50.07 kB of Flash memory, making it suitable for implementation in resource-limited edge computing devices like the STM32 MCU.

PADNet undergoes optimization using the TF-Lite tool and is tested on the STM32 MCU to assess its performance. The model achieves an average inference time per sample of only 12.2 ms. Further validation in an embedded system for real-time performance evaluation, utilizing an STM32 microcontroller (NUCLEO-H743ZI2) and a solid-state circuit breaker, demonstrated an arc fault clearing time of 25 ms. The STM32 microcontroller, with its comparatively lower computational power, memory size, and RAM (1 MB), proves to be a cost-effective solution. The performance evaluation demonstrates the feasibility of the PADNet model for real-time arc fault protection in photovoltaic systems, enabling implementation in resource-limited commercial MCUs with rapid fault-clearing times.

The main objectives of this study were to utilize a lightweight architecture for the CNN-based model and achieve fast interruption with reasonably high accuracy for DC arc fault interruption using a solid-state circuit breaker.

Although a test bed has been developed to collect PV arc fault data, real-world applications may still exhibit some differences compared to lab-collected data. The lab environment may not include external environmental noise that can affect real PV systems. Additionally, the algorithm has only been tested with load currents ranging from 3 to 5 A. High-current arc faults may present more subtle differences than low-current arc faults. Furthermore, adding more power electronics equipment to the system may alter the noise level, potentially reducing the accuracy of the models developed based on lab-collected data.

To address this issue, a domain adaptation method can be adopted, where a smaller number of arc and normal samples from real-world PV systems are added to the training of this algorithm. This would help the model to learn arc features over a wider range and make it more effective in all environments. This calls for further research.

Future research directions include conducting extensive tests under various loading and environmental conditions to assess the system's performance and generalizability. Insights gained from these tests will be used to refine the proposed CNN model, leading to a more robust and reliable system for real-world application.

CHAPTER 6: OPTIMIZATION AND GENERALIZATION OF THE DC AFCI

Ensuring the robustness and reliability of DC AFCIs is essential for maintaining the safety and efficiency of photovoltaic (PV) systems, especially under diverse operating conditions. Building on previous research, this chapter aims to expand the dataset to include information from various operating environments, shadow occlusions, and inverter startup conditions. The objective is to enhance the model's robustness.

To achieve this, a lightweight architecture is employed, along with model compression techniques using a knowledge distillation-based teacher-student model. This strategy ensures that the model remains robust while being efficient enough for deployment on resource constrained edge devices.

The key contributions of this research are:

1. **Enhanced Database:** The dataset has been expanded to include diverse operating conditions, environmental factors, shadow occlusions, and inverter startup scenarios. This extensive dataset helps the model generalize better and maintain high performance across different situations.

2. **Lightweight Architecture:** A lightweight design using pointwise and depthwise convolutions instead of traditional CNN blocks has been proposed. These techniques minimize computational load while maintaining high detection accuracy.

3. **Knowledge Distillation-Based Model Compression:** A teacher-student model approach for compression has been used. The larger teacher model guides the training of a smaller, more efficient student model. This technique retains the accuracy of the original model while significantly reducing its size and computational demands.

4. **Improved Robustness Strategy:** A systematic strategy to enhance the model's robustness has been developed. This includes noise handling techniques, data augmentation, and advanced regularization methods to improve generalization.

5. **Real-Time Implementation:** The optimized model has been tested on a commercial microcontroller unit (MCU), the STM32H743ZI2. Real-time performance has been evaluated under various conditions, demonstrating the model's robustness and efficiency. Inference time per sample was significantly reduced while maintaining high detection accuracy. The enhanced model has been integrated with a solid-state circuit breaker to ensure rapid circuit interruption during arc fault events. This integration demonstrates the practical applicability of the solution in real-world PV systems.

6. **Online Accuracy and Consecutive Arc Cycle Detection:** The model's online accuracy in detecting arc faults has been demonstrated, showcasing its ability to maintain high detection rates in real-time scenarios. The robust algorithm's effectiveness has been further validated by detecting consecutive arc cycles in a single arc event, proving its reliability in continuous monitoring and detection.

7. **Power Electronic Noise Immunity Testing:** The algorithm has been tested by connecting the DC output to an inverter and then to the AC grid. This ensured that power electronic noise did not affect the algorithm's performance, confirming its robustness in noisy environments. This step was crucial, as previous tests focused mainly on resistive loads.

This chapter highlights significant advancements in model optimization and generalization for robust DC AFCIs, paving the way for more reliable and efficient arc fault detection systems in photovoltaic applications.

6.1 Database Enhancement

Collecting high-quality data is very important for developing a reliable DC AFCI. To build an effective model, an extensive PV arc fault database has been constructed by simulating various real-world conditions using a Solar Array Simulator (SAS) module. The schematic architecture of the arc fault data collection and test setup is provided in Fig. 6.1 which includes an inverter and AC grid (208 V, 2-phase). The experimental setup is provided in Fig. 6.2.

A system has been set up to generate arc faults with both fixed and variable arc gaps by moving the movable electrode. This setup has allowed the capture of the nuanced behavior of arc faults in photovoltaic systems under different conditions. The loads have been varied, inverter startup conditions have been simulated, and shadow occlusions as well as sudden load changes have been introduced to ensure that the model could be trained with a wide range of scenarios.

The dataset includes a substantial number of samples with current ranging from 3 A to 8.5 A while the open circuit voltage ranged from 480 V to 600 V (see Table 6.1). The database consists of a total of 102,000 normal samples and 52,000 arc samples (see Table 6.2). This balanced representation helps in training and testing the model effectively. Data has been collected at 250 KS/s; however, for model simplicity, a sampling rate of 51.2 kS/s has been used, with each sample lasting 10 ms, which is sufficient to capture the detailed characteristics of arc faults. The input to the model was FFT data.

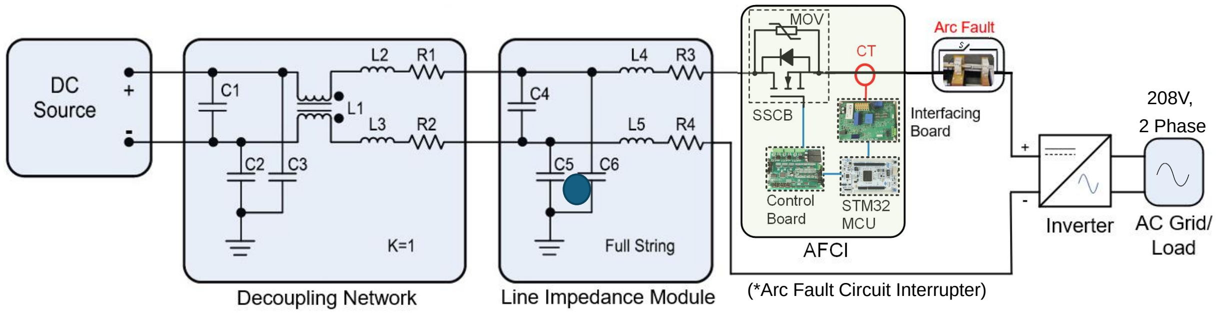


Figure 6.1: Schematic diagram of PV arc fault data collection test bench.

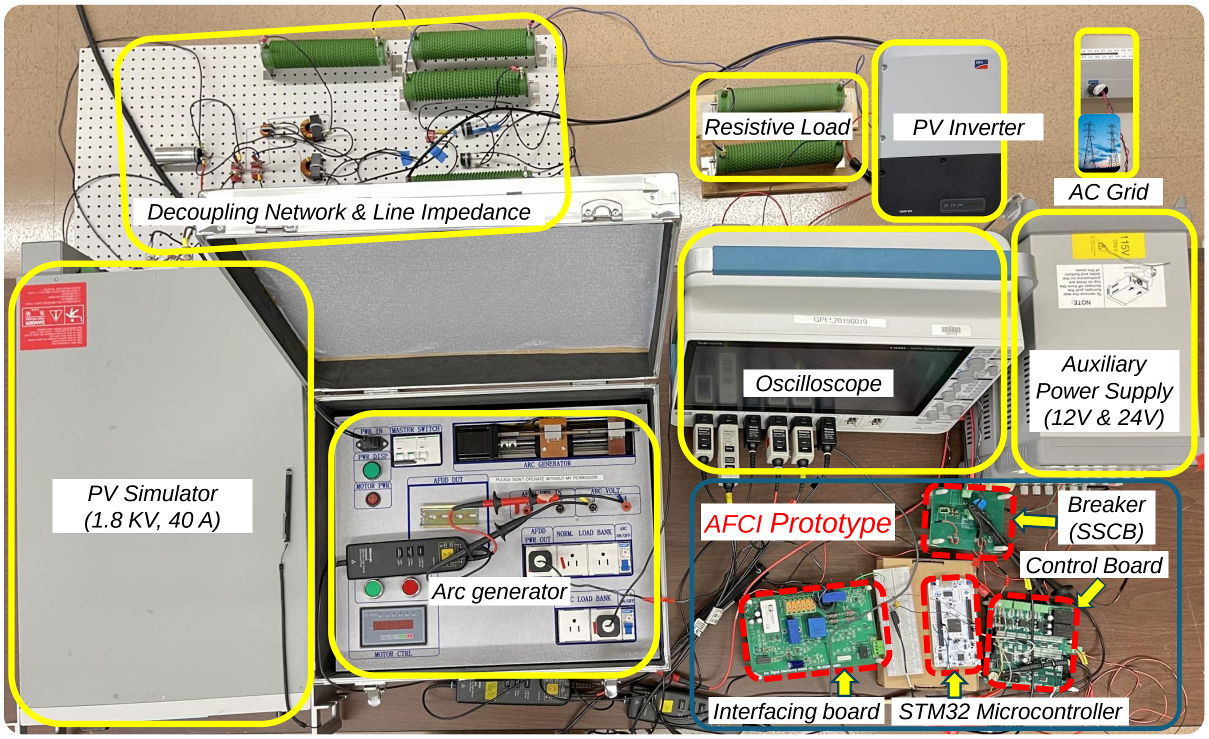


Figure 6.2: Experimental setup for PV arc fault data collection and real-time testing.

Table 6.1: Data description of PV arc fault database.

Sl. No.	Imp (A)	Min Iarc (A)	Vmp (V)	Isc (A)	Voc (V)
1	3	2.5	400	4	490
2	4	3.5	480	5	550
3	5	4.5	500	6	600
4	6	5.5	520	7.5	600
5	7	6.5	520	8.5	600
6	8.5	8	540	10	600

To further ensure the quality of the data, samples have been projected into the frequency domain using techniques like t-SNE (t-Distributed Stochastic Neighbor Embedding). This approach helped visualize the variations and understand the underlying data distribution patterns (see Fig. 6.3). The results indicate that the data distribution is more complex due to the inclusion of diverse data. The data clustering is also worse than before, which means a simpler algorithm may not be able to classify the arc faults properly and efficiently. The database includes at least eight different loading conditions along with other critical environmental conditions, providing a robust foundation for the model to generalize and perform well in diverse environments.

Table 6.2: Total samples in the database

Arc Samples	**Normal Samples**	**Total**
50,000	52,000	102,000

Our data collection process was carefully designed to encompass a wide range of scenarios, resulting in a comprehensive and high-quality dataset. This extensive data is key to the model's ability to generalize and accurately detect arc faults in real-world PV systems.

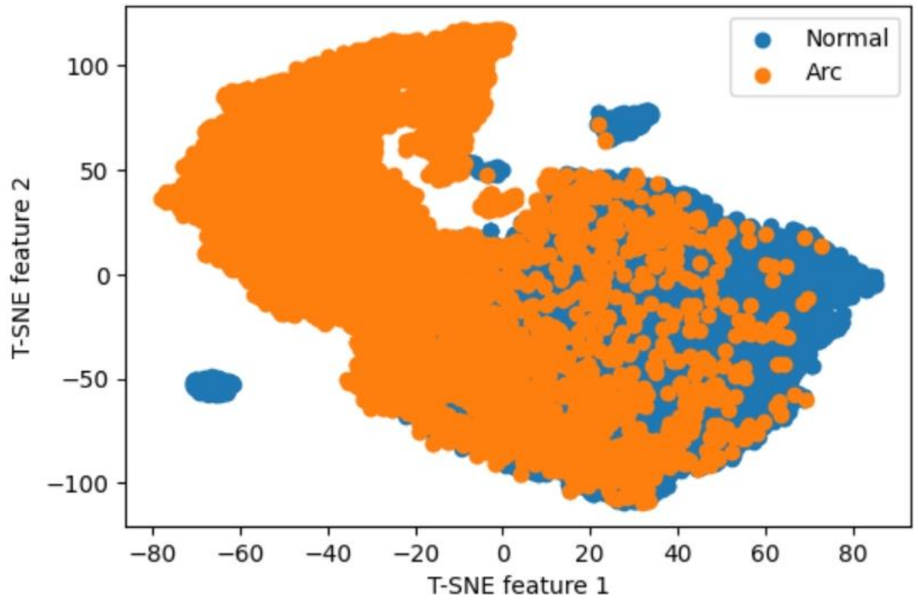


Figure 6.3: t-SNE projection of PV arc fault data in the frequency domain.

6.2 Network Architecture and Methodology of the Proposed Model

The PADNet-Lite architecture is designed to be lightweight yet highly effective in detecting DC arc faults in photovoltaic systems. This section breaks down the key components and techniques used to achieve this balance between performance and efficiency.

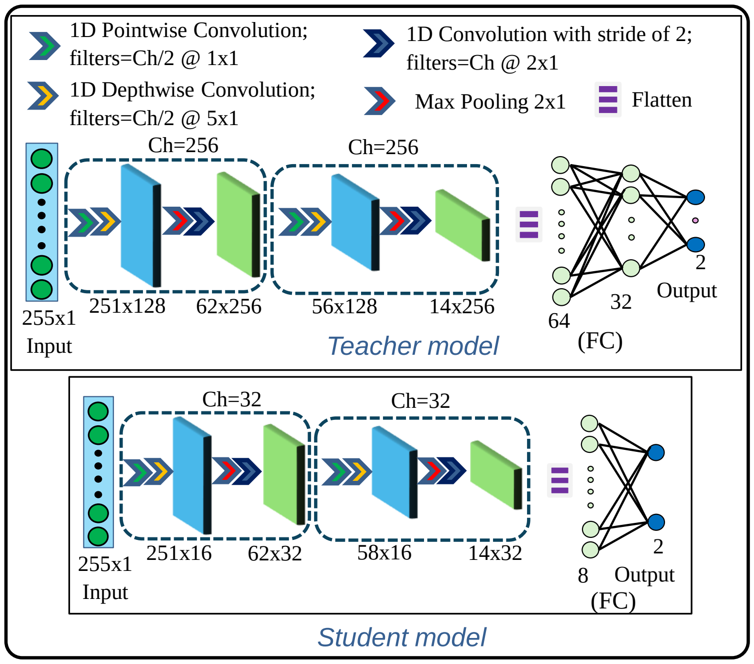


Figure 6.4: Network architecture of the proposed knowledge distillation-based PADNet-Lite model.

6.2.1 Lightweight CNN Design

Our proposed model PADNet-Lite (Lightweight PV Arc Fault Detection Network) combines advanced convolution techniques with knowledge distillation to create a model that is both lightweight and highly accurate. Its robust training process and efficient architecture make it suitable for deployment in real-world PV systems to detect DC arc faults effectively. The model architecture of the teacher-student knowledge distillation method is depicted in Fig. 6.4 To create a model that is both powerful and efficient, PADNet-Lite leverages pointwise and depthwise convolutions instead of traditional CNN blocks.

- Pointwise Convolutions: These involve applying a 1x1 convolution to each pixel, which helps reduce the number of parameters and computations by limiting the filter size. This is particularly useful in extracting features from the input data while keeping the model size small.

- Depthwise Convolutions: These convolutions apply a single filter to each input channel separately, which significantly reduces the computational load compared to applying multiple filters across all channels. By combining depthwise and pointwise convolutions, PADNet-Lite maintains high accuracy in fault detection while being computationally efficient enough to run on resource-limited edge devices.

- Knowledge Distillation: Knowledge distillation is a technique used to compress the model while preserving its performance. This involves training a smaller student model using the guidance of a larger, more complex teacher model. The teacher model, with its larger capacity, is trained first to achieve high accuracy. The knowledge gained by this model is then transferred to a smaller student model. The student model learns to mimic the teacher model's outputs, benefiting from its insights without the need for the same computational resources. This approach ensures that the student model remains lightweight and efficient, making it suitable for deployment on edge devices with limited computational power. Despite its reduced size, the student model retains much of the accuracy of the larger teacher model.

6.2.2 Model Training

The training process for PADNet-Lite involves several key parameters and strategies to ensure robust performance and prevent overfitting.

The teacher model consists of 430.88K parameters, while the student model has only 4.27K parameters. This significant reduction in model size is achieved through knowledge distillation, where the larger teacher model transfers its knowledge to the smaller student model.

For training, the teacher model was trained with a batch size of 200 for 40 epochs, whereas the student model used a batch size of 32 for 50 epochs. Early stopping was implemented at 39 epochs for the teacher model and 35 epochs for the student model to

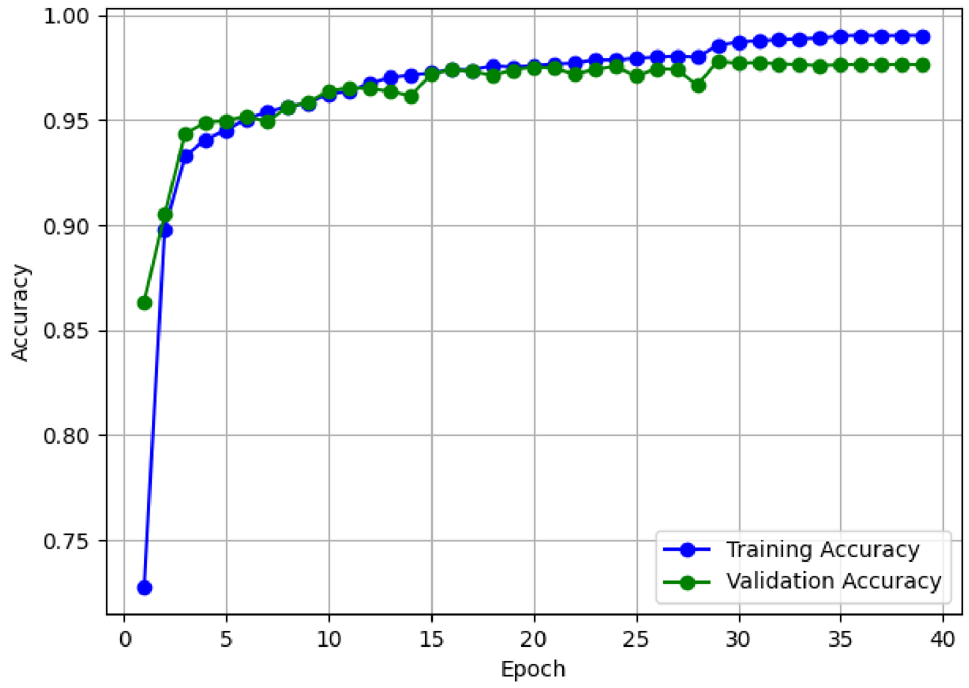


Figure 6.5: Training and validation accuracy plots of teacher model.

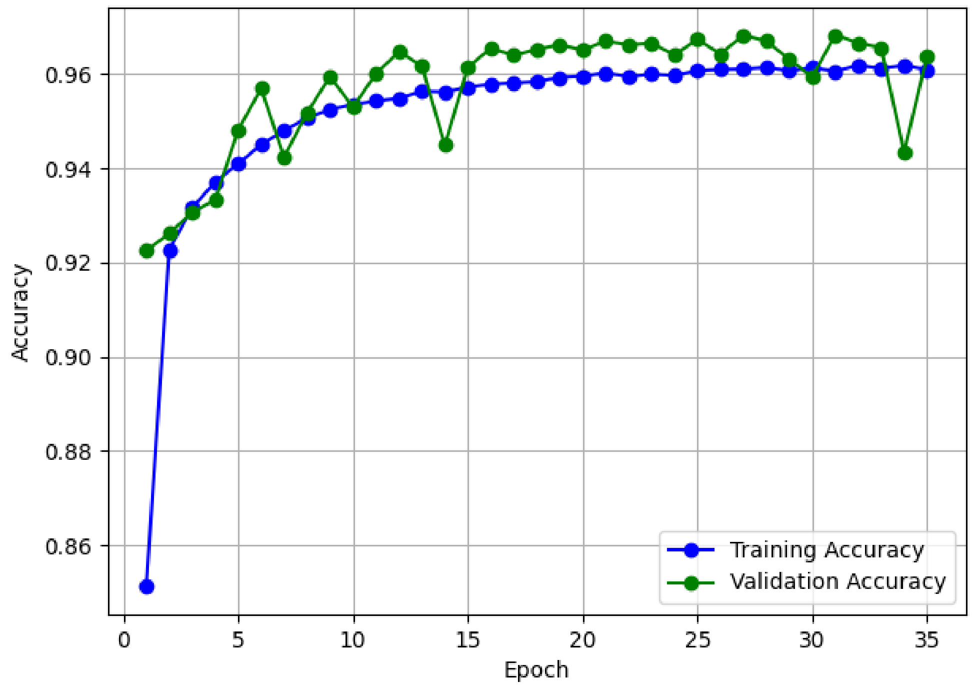


Figure 6.6: Training and validation accuracy plots of student model.

prevent overfitting by halting training when the model's performance on a validation set stops improving.

Both models utilized an adaptive learning rate starting at 0.001. For the student model, L1 regularization was applied to penalize large weights and encourage simplicity. Additionally, a dropout layer with a rate of 0.5 was added to the final convolutional layer of the student model to further prevent overfitting by reducing the dependency on any single feature.

During the knowledge distillation process, a temperature (T) of 5 and an alpha value of 0.3 were used to control the softening of the probability outputs and balance the influence of the teacher model's predictions and the ground truth data during the student's training.

The effectiveness of these strategies is evident in the training and validation accuracy plots for both models (see Fig. 6.5 & Fig. 6.6). The curves closely follow each other, indicating that overfitting was successfully mitigated. For the student model, the training accuracy quickly rises and stabilizes around 96%, with the validation accuracy closely mirroring this trend. This alignment between training and validation accuracy curves confirms that the model generalizes well to new, unseen data.

6.3 Experimental Results and Discussion

The experimental results of the PADNet-Lite model are presented in a comprehensive comparison with both the teacher model and a baseline student model trained from scratch, as shown in Table 6.6. The aim is to highlight the performance, accuracy, and efficiency of the student model compared to the teacher model and the baseline.

6.3.1 Results of PADNet-Lite Model

The teacher model, with its 430.88K parameters, achieved the highest accuracy of 97.45%. This model, however, is significantly larger, with a model size of 5,135 KB, making it impractical for deployment on resource-constrained edge devices. The student model, derived through knowledge distillation, has only 4.27K parameters and a much smaller model size of 52 KB. Despite this reduction in size, the student model achieved an impressive accuracy of 96.52%, demonstrating that it retained most of the performance benefits of the teacher model while being significantly more efficient (see Table 6.6).

Table 6.3: Confusion matrix of student model.

		Predicted		
		Normal	**Arc**	**Total**
Actual	**Normal**	10167 (98.07%)	200 (1.93%)	10367
	Arc	509 (5.07%)	9524 (94.93%)	10033
	Total			10400

Type I error (false positive)	1.93%
Type II error (false negative)	5.07%
Overall accuracy	**96.52%**

Table 6.4: Confusion matrix of teacher model.

		Predicted		
		Normal	**Arc**	**Total**
Actual	**Normal**	10207 (98.46%)	160 (1.54%)	10367
	Arc	360 (3.59%)	9673 (96.41%)	10033
	Total			10400

Table 6.5: Confusion matrix of student scratch model.

		Predicted		
		Normal	**Arc**	**Total**
Actual	**Normal**	10062 (97.06%)	305 (2.94%)	10367
	Arc	1097 (10.93%)	8936 (89.07%)	10033
	Total			10400

In comparison, the student model trained from scratch (without knowledge distillation) also has 4.27K parameters but achieved a lower accuracy of 93.13%. This indicates that the

Table 6.6: Comparison of teacher, student, and student scratch models.

Model	**Teacher**	**Student**	**Student Scratch**
Parameters	430.88 K	4.27 K	4.27 K
Accuracy	97.45%	96.52%	93.13%
Loss	0.0806	0.0385	0.2203
Model size (kB)	5,135	52	122

knowledge distillation process effectively transferred critical knowledge from the teacher model to the student model, enhancing its performance beyond what could be achieved through traditional training alone.

The confusion matrices (Table 6.4, Table 6.3, and Table 6.5) provide a detailed breakdown of each model's performance in classifying normal and arc fault conditions. The teacher model correctly identified 10,207 out of 10,367 normal samples (98.46%) and 9,673 out of 10,033 arc samples (96.41%). It misclassified 160 normal samples as arcs (1.54%) and 360 arc samples as normal (3.59%). The student model showed a slightly lower performance but still impressive, correctly classifying 10,167 out of 10,367 normal samples (98.07%) and 9,524 out of 10,033 arc samples (94.93%). The misclassification rates were 1.93% for normal samples and 5.07% for arc samples. This performance is very close to that of the teacher model, underscoring the effectiveness of the knowledge distillation process in maintaining high accuracy with a significantly smaller model. The student model trained from scratch, however, showed a noticeable drop in performance, correctly identifying 10,062 out of 10,367 normal samples (97.06%) and 8,936 out of 10,033 arc samples (89.07%). The misclassification rates were higher, at 2.94% for normal samples and 10.93% for arc samples, demonstrating that this model struggled more with accurate classification compared to the student model derived from knowledge distillation.

The key takeaway from these results are that the teacher model, while highly accurate, is too large for practical use in edge devices. The student model, created through knowledge distillation, provides a highly efficient and almost equally accurate alternative, making it suitable for real-world applications. The baseline student model trained from scratch was not able to match the performance of the distillation-derived student model, highlighting the value of the knowledge distillation technique. These results demonstrate that the PADNet-Lite student model offers a compelling balance of high accuracy and efficiency, making it a practical solution for real-time DC arc fault detection in photovoltaic systems.

6.3.2 Real-Time Implementation Details

6.3.2.1 AFCI Testing During Regular Operation

For real-time arc fault detection and circuit interruption experiments, the STM32H743ZI2 microcontroller unit (MCU) and a SiC MOSFET-based solid-state circuit breaker (SSCB) were utilized. The STM32H743ZI2 is particularly suited for this application due to its balanced performance and cost-effectiveness. It features a Cortex M7 processor running at 480 MHz, with 1 MB of RAM and 2 MB of flash memory. This configuration provides sufficient computational power and memory to handle the arc fault detection algorithm while maintaining low power consumption. The choice of this MCU was influenced by its affordability, costing only about $25, making it an ideal choice for cost-sensitive AFCI solutions in photovoltaic systems. The STM32H743ZI2 offers a compelling balance of processing capability and cost when compared to other platforms like the Jetson Nano and Raspberry Pi 4B. Although powerful, the Jetson Nano and Raspberry Pi 4B are more expensive compared to the STM32H743ZI2.

The algorithm for arc fault detection was tested under two conditions: without AFCI and with AFCI. One condition demonstrates timely detection, and the other shows fast interruption using the SSCB. The real-time implementation details are illustrated in Fig. 6.7. Under conditions without the AFCI (Imp = 5A, Vmp = 500V, and Voc = 600V), the arc fault starts and the arc voltage rises to around 30V. The arc detection time is approximately 23 ms, but without the AFCI, the arcing continues as there is no mechanism to interrupt the circuit. As long as the arc fault continues, the arc detect signal remains

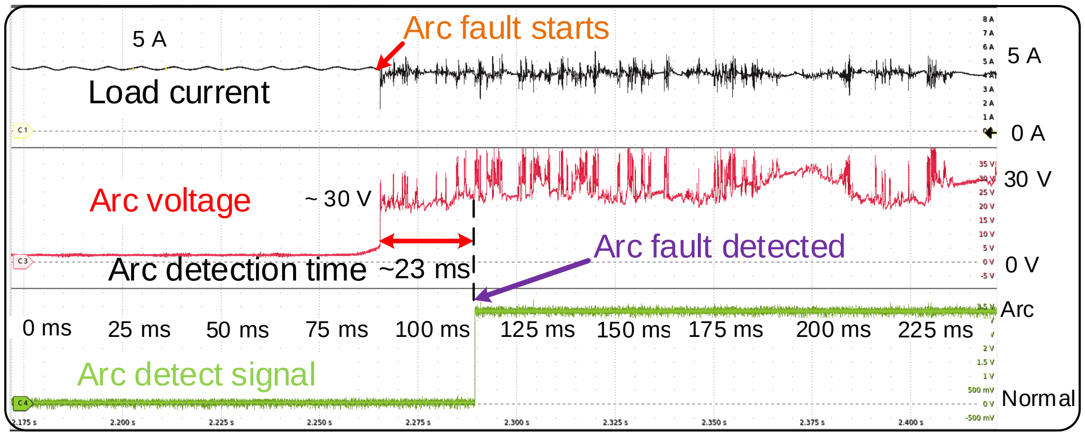


Figure 6.7: Real-time online arc fault detection using STM32 MCU.

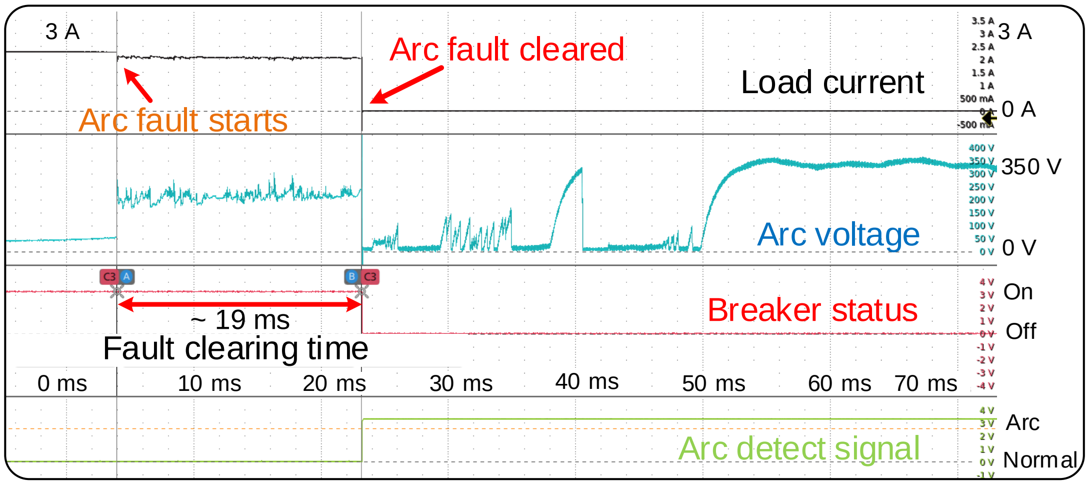


Figure 6.8: Real-time arc fault circuit interruption using STM32 MCU & SSCB.

high, indicating that the algorithm is continuously detecting arc faults during arcing time.

In contrast, under conditions with the AFCI (Imp = 3 A, Vmp = 400 V, Voc = 450 V), the arc fault is detected and the SSCB quickly interrupts the circuit (see Fig. 6.8). The fault clearing time is around 19 ms, effectively stopping the arcing and protecting the system. The use of SiC MOSFETs in the SSCB provides a rapid and reliable method for breaking the circuit upon detection of an arc fault. This fast response time is crucial for preventing damage and ensuring the safety of the photovoltaic system.

These real-time tests demonstrate the effectiveness of the PADNet-Lite model when implemented in the STM32H743ZI2 MCU in conjunction with the SiC MOSFET-based

SSCB for rapid arc fault detection and interruption. The setup successfully detects and clears arc faults much faster than traditional methods, with the AFCI-equipped system showing a fault clearing time of just 19 ms. The STM32H743ZI2's balance of low cost, sufficient computational power, and robust community support make it an ideal choice for real-time arc fault detection in photovoltaic systems, ensuring both safety and efficiency.

6.3.2.2 AFCI Testing During Regular Operation

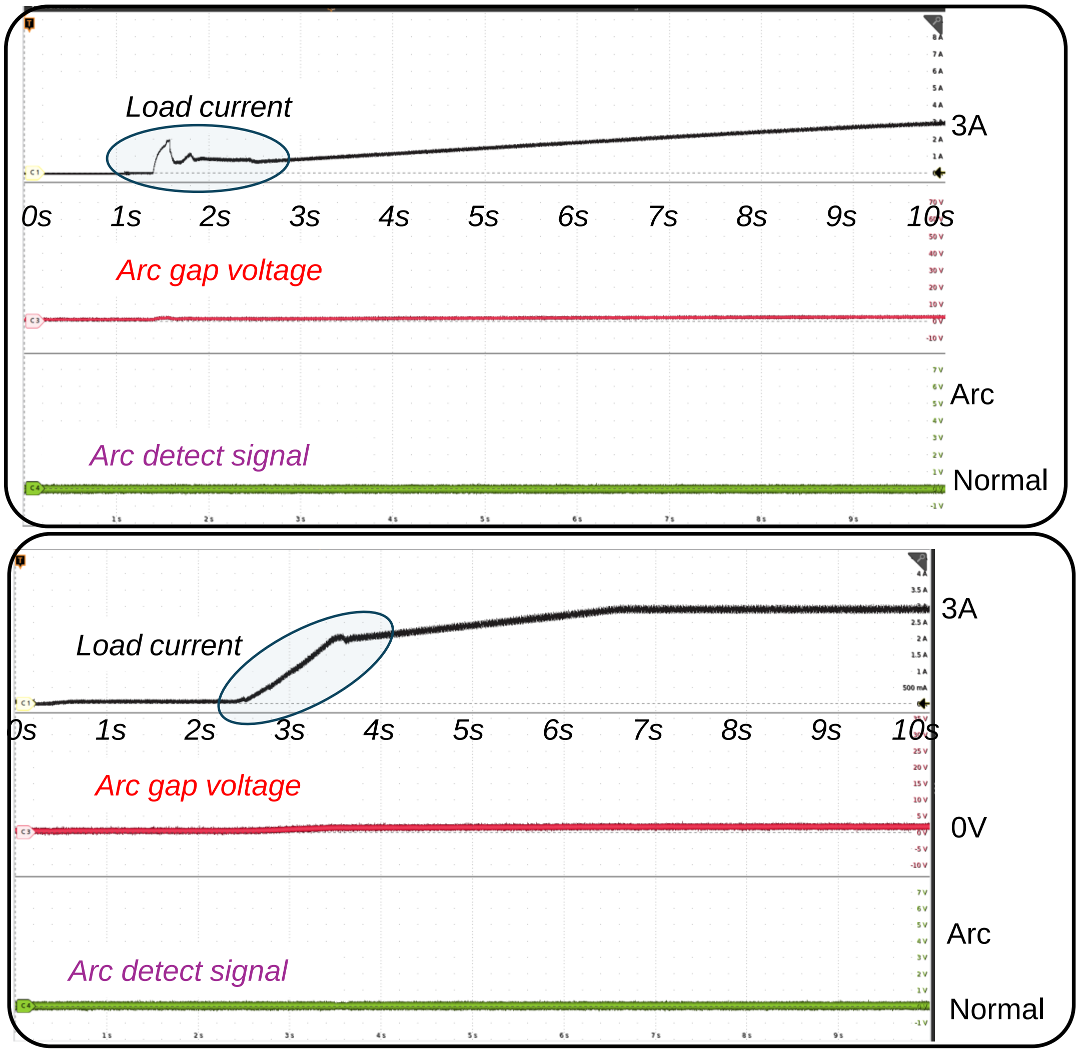


Figure 6.9: Real-time AFCI performance evaluation during inverter startup.

The performance of the arc fault detection system was evaluated under inverter startup conditions to assess its robustness against the dynamic behaviors typically seen during the startup phase of photovoltaic systems. During startup, inverters exhibit transient behaviors where currents and voltages fluctuate before settling into a steady state. A few of these experiments and results are depicted in Figure 6.9.

Scenario 1: In this scenario, the load current gradually increases from 0 A to 3 A within the first 10 seconds. Throughout this period, the arc gap voltage remains stable near zero, and the arc detect signal does not trigger, indicating any detection of arc faults. The transient fluctuations in the load current at around 1.5 seconds do not affect the arc gap voltage or trigger any false alarms, showing the algorithm's ability to differentiate normal startup behavior from actual faults.

Scenario 2: Similar to the first scenario, the load current ramps up to 3A over 10 seconds. However, there are noticeable transient fluctuations in the load current between the 2 and 3-second marks. Despite these transients, the arc gap voltage stays constant, and the arc detect signal remains inactive. This indicates that the system correctly identifies these transient behaviors as normal and does not falsely detect an arc fault.

The results from these two scenarios demonstrate that the arc fault detection system is capable of distinguishing between normal inverter startup transients and actual arc faults. This ability is crucial for preventing false positives and ensuring reliable operation in real-world photovoltaic systems.

6.3.2.3 Shadow Occlusion and Change in Irradiance Test

The arc fault detection system was evaluated under conditions of changing irradiance and shadow occlusion to test its robustness in varying environmental conditions. These tests simulate real-world scenarios where photovoltaic systems experience fluctuations in sunlight due to moving clouds or other obstructions.

In these experiments, three scenarios involving changes in load current and irradiance were tested, as illustrated in Fig. 6.10, Fig. 6.11, and Fig: 6.12.

- Scenario 1: Shadow Occlusion: In this scenario, the load current changes from 0A

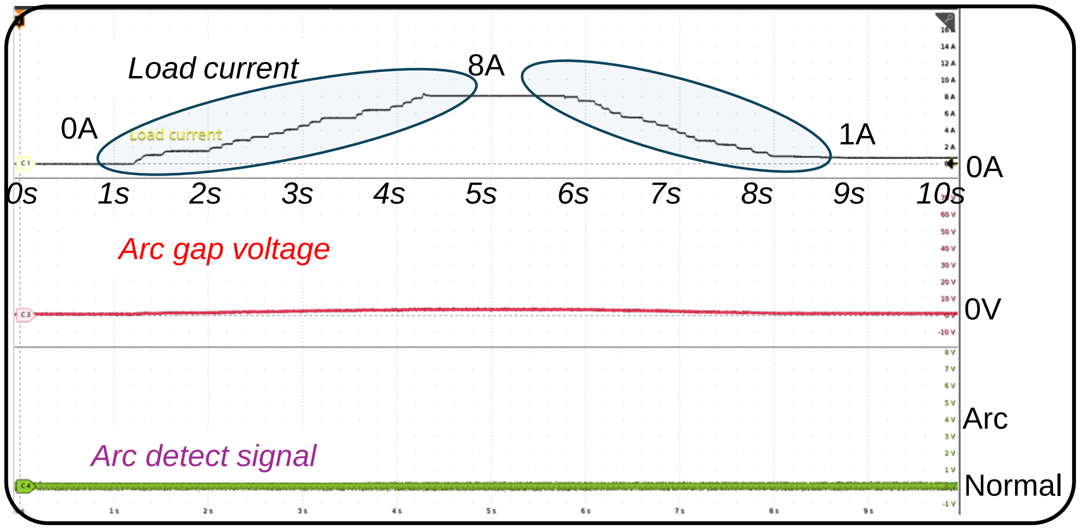


Figure 6.10: Real-time performance evaluation of the proposed AFCI during shadow occlusion.

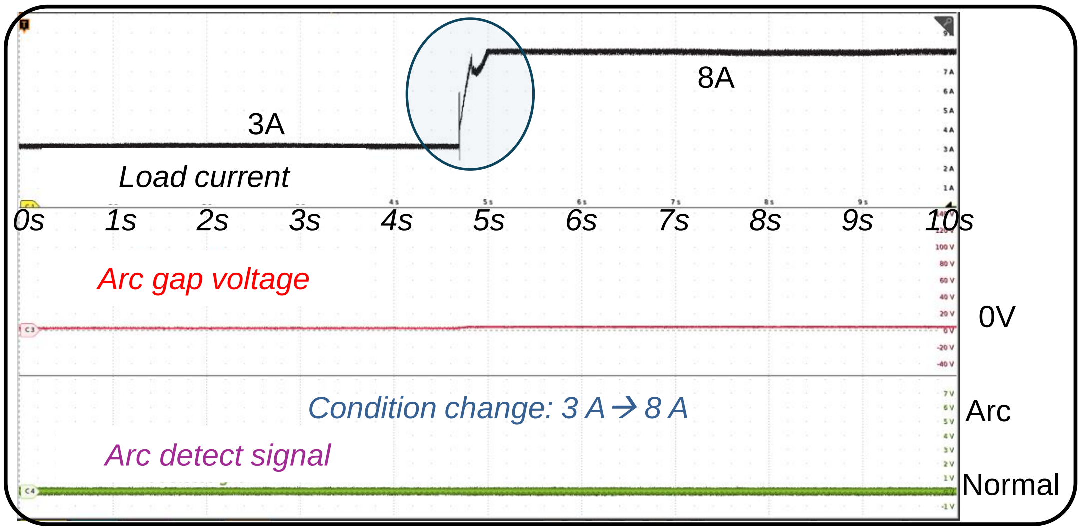


Figure 6.11: Sudden load change (from 3 A to 8 A) test of AFCI.

to 8A and then drops to 1A within the 10-second period. Despite these significant changes in load current, the arc gap voltage remains stable, and the arc detect signal does not trigger. This indicates that the system accurately identifies these changes as normal variations in irradiance rather than arc faults.

- Scenario 2: Load Change (Jumping Up): The load current starts at 3A and jumps

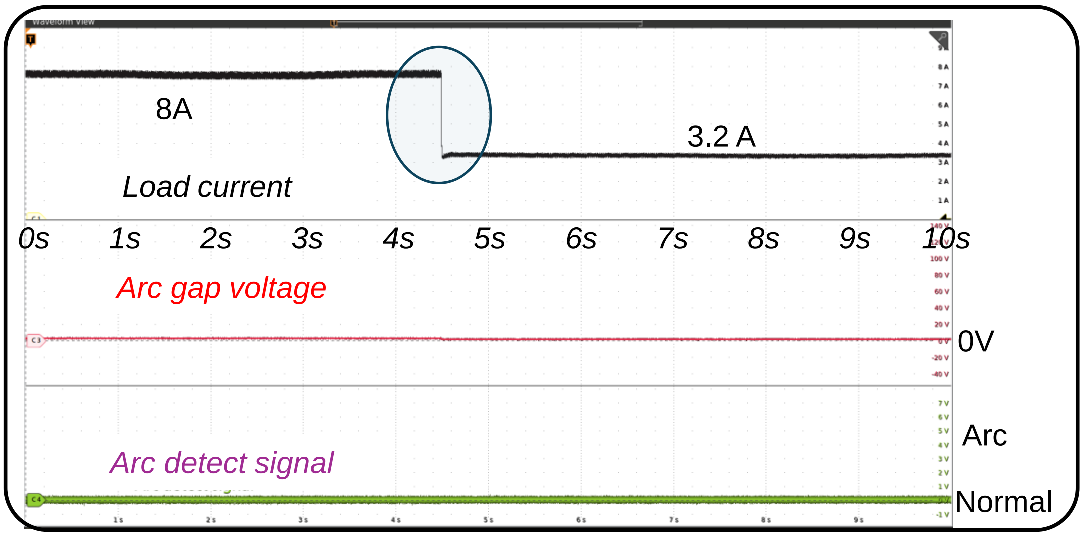


Figure 6.12: Sudden load change (from 8 A to 3 A) test of AFCI.

to 8A at the 5-second mark. This sudden change in load current is typical of a cloud passing over the PV array, temporarily increasing the irradiance. The arc gap voltage remains unchanged, and the arc detect signal stays low, confirming that the system does not misclassify this scenario as an arc fault.

- Scenario 3: Load Change (Jumping Down): In this shadow occlusion test, the load current initially at 8A drops sharply to 3.2A at the 5-second mark. The arc gap voltage remains consistent throughout, and the arc detect signal does not activate. This scenario further validates the system's ability to handle rapid changes in irradiance without producing false positives.

The results demonstrate the arc fault detection system's robustness and reliability under varying environmental conditions. The system successfully maintained a classification accuracy of 100% with zero false positives, even during significant changes in load current and irradiance. This capability is crucial for real-world applications, ensuring that the system can accurately differentiate between normal environmental variations and actual arc faults.

6.3.2.4 Testing at Different Loading Conditions

The arc fault detection system was tested under various current (Imp) and voltage (Vmp) settings to evaluate its performance across different operating conditions. These tests aimed to simulate real-world scenarios where photovoltaic systems might experience varying loads and voltage levels. The results of these tests are illustrated in Fig. 6.13. Scenario 1 (4A, 480V): The load current was set to 4A and the voltage to 480V. An arc fault started around the 5-second mark, and the arc gap voltage increased accordingly. The system successfully detected the arc fault with no false positives during normal operation, indicating high accuracy.

Scenario 2 (5A, 500V): With the load current increased to 5A and the voltage to 500V, the arc fault detection again began around the 5-second mark. The system accurately detected the arc fault, maintaining its reliability with no false positives.

Scenario 3 (6A, 520V): In this test, the load current was 6A and the voltage 520V. The system detected the arc fault but showed a few false negatives towards the end of the 10-second period. This suggests a slight decrease in detection accuracy at higher currents and voltages.

Scenario 4 (7A, 520V): At a load current of 7A and voltage of 520V, the system detected the arc fault but with some false negatives as the arcing continued. Despite these inaccuracies, the system effectively broke the circuit at the first occurrence of an arc fault, mitigating potential hazards.

The arc fault detection system demonstrated reliable performance across various current and voltage settings, with minimal false positives. However, the likelihood of false negatives increased at higher current and voltage levels. The system's capability to quickly interrupt the circuit upon initial arc fault detection ensures enhanced safety and stability for photovoltaic systems.

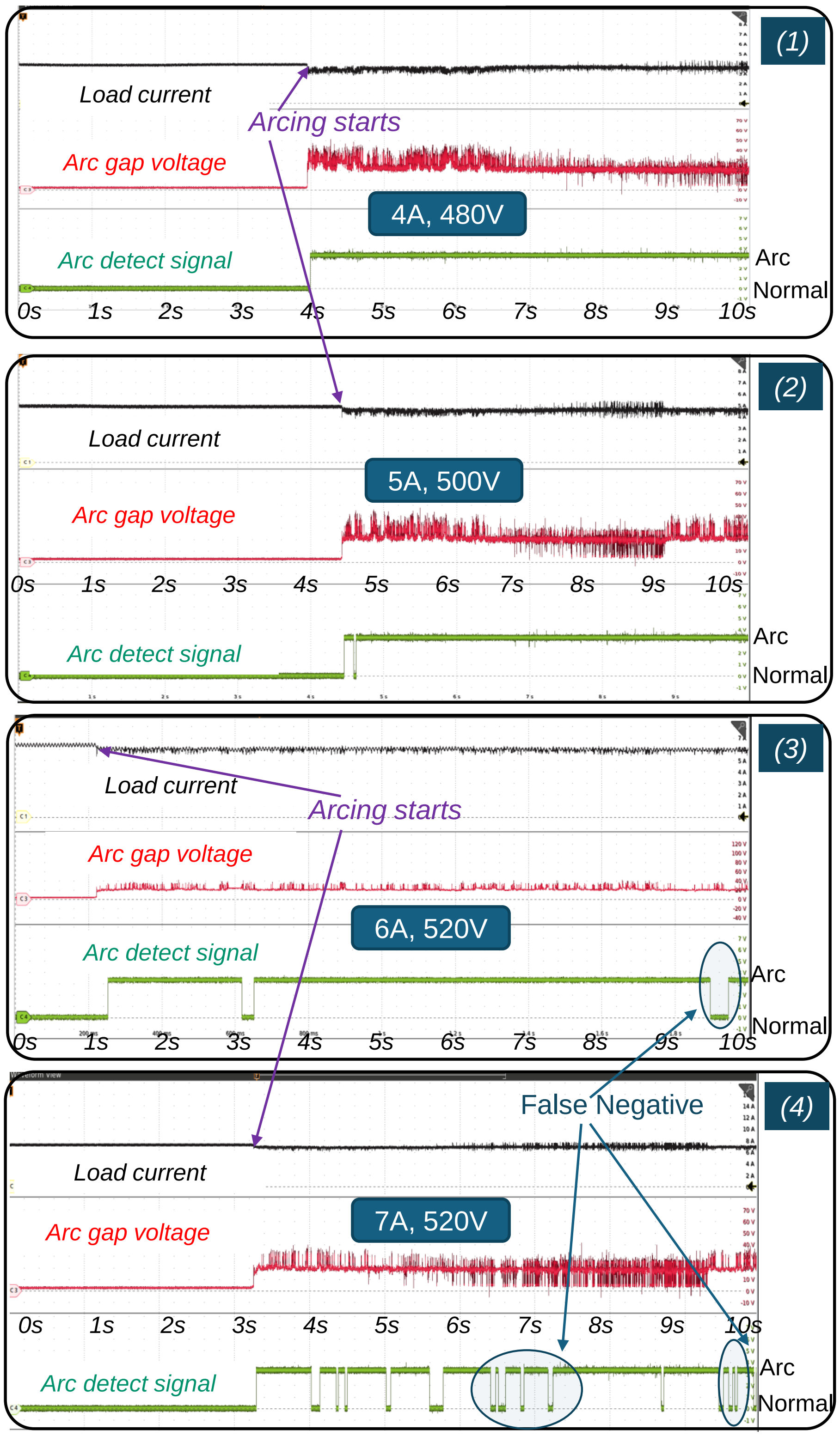


Figure 6.13: Real-time arc fault detection test under diverse loading conditions.

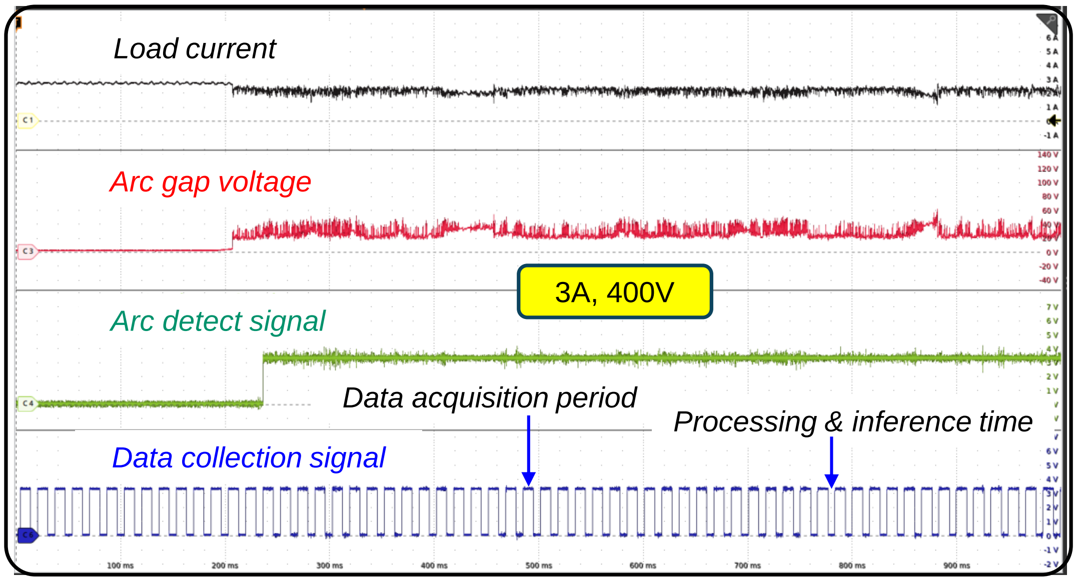


Figure 6.14: Real-time online arc fault detection accuracy test under diverse loading conditions.

Table 6.7: Online accuracy testing of the proposed PADNet-Lite model

Current	Duration	Samples	Accurate Detection
3A (mixed)	1s	60	59
4A (normal)	1s	60	60
6A (mixed)	10s	600	596
6A (mixed)	2s	120	118
7A (mixed)	2s	120	115
8A (mixed)	10s	600	583
Total =		**1560**	**1531**

6.3.2.5 Real-time Online Accuracy Testing

The real-time arc fault detection accuracy of the proposed system was evaluated under diverse loading conditions to ensure its robustness and reliability in different operating scenarios. The test setup involved monitoring various parameters such as load current, arc gap voltage, arc detect signal, and data collection signal, as shown in Fig. 6.14.

In the depicted test scenario, the load current is set to 3A with a voltage of 400V. The data acquisition (DAQ) period and the processing and inference time are clearly marked.

The DAQ period is indicated by the LED signal being high, and the LED signal being low represents the processing and inference time. This method allowed for precise calculation of accuracy based on ground truth data.

The system's online accuracy was tested under various loading conditions, summarized in Table 6.7. The conditions ranged from mixed scenarios to normal operation, with currents varying from 3A to 8A and durations from 1 to 10 seconds. The results show the number of samples collected and the accurate detections made by the system. In total, 1560 samples were collected across all conditions, with 1531 samples accurately detected, resulting in an overall detection accuracy of approximately 98.14%. This high level of accuracy demonstrates the system's effectiveness in real-time arc fault detection across various loading conditions, confirming its suitability for use in photovoltaic systems.

6.3.3 Reliability Enhancement of PV AFCI

To enhance the reliability of the arc fault detection system, a strategy was adopted where five consecutive arc cycles are considered as one arc fault event. This approach addresses issues associated with fast tripping due to detection within a single cycle, which can be prone to inaccuracies. By waiting for multiple consecutive cycles, the system ensures more robust detection of arc faults. As illustrated in Fig. 6.15, this method improves the reliability of the detection process.

The average fault detection time was found to be between 95 to 106 milliseconds. This slight delay ensures that the system captures the arc signature more accurately. Sometimes, the first cycle of an arc event may go undetected due to partial capture of the arc signature, but the inclusion of subsequent cycles enhances overall detection accuracy. This approach significantly reduces false trips and enhances the dependability of the arc fault detection system in real-world photovoltaic applications.

Additionally, Fig. 6.16 demonstrates the real-time AFCI testing with the five-cycle arc approach. The results show that the arc fault is cleared within 95 milliseconds, confirming

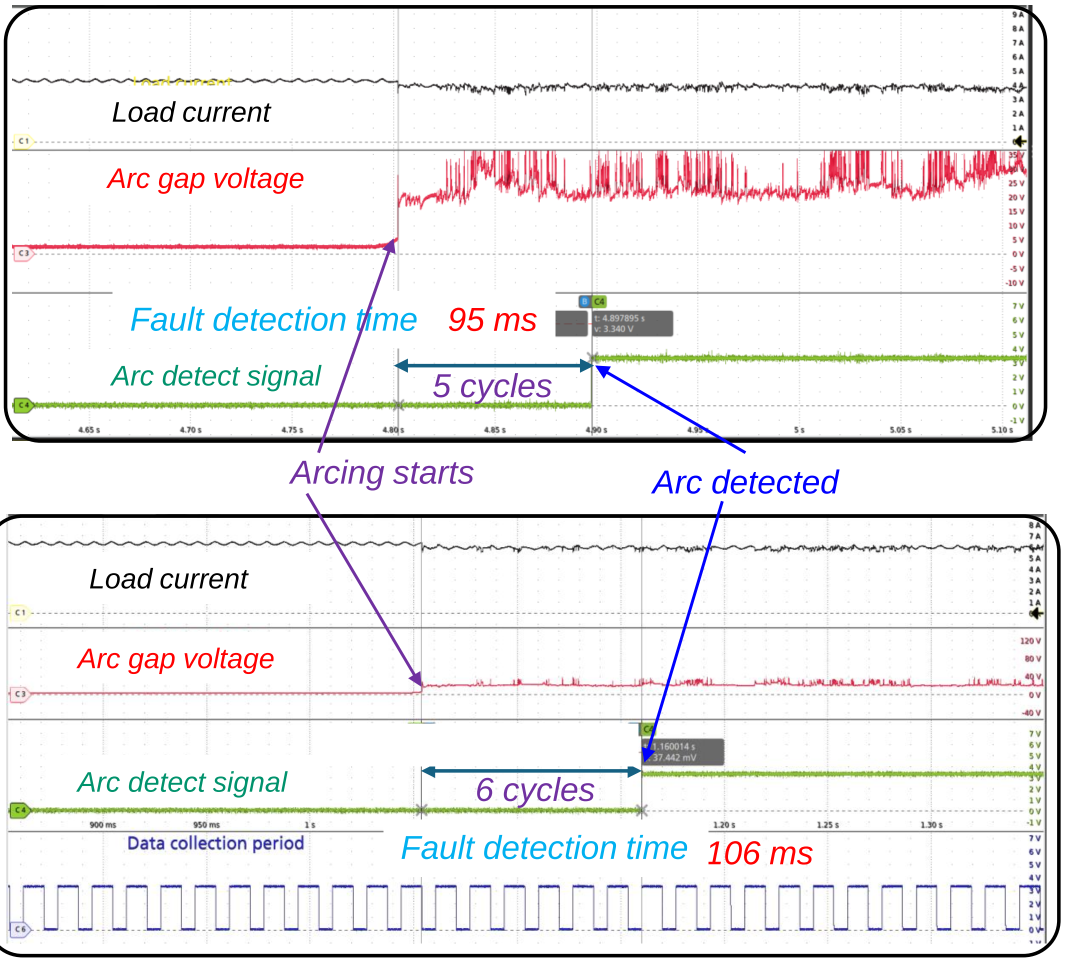


Figure 6.15: Reliability enhancement using 5 cycles of arc per arc event.

the system's ability to respond quickly and accurately. This method is less likely to result in false tripping, thereby enhancing the overall safety and reliability of the photovoltaic system.

6.4 Performance Comparison with Existing Methods

The final evaluation of the robust AI-based AFCI involved a comprehensive performance comparison of the proposed arc fault detection method with existing state-of-the-art models. The comparison focused on accuracy and runtime, providing a clear picture of the effectiveness and efficiency of the proposed model.

As shown in Fig. 6.17, the proposed model achieved an accuracy of 98.14% with the

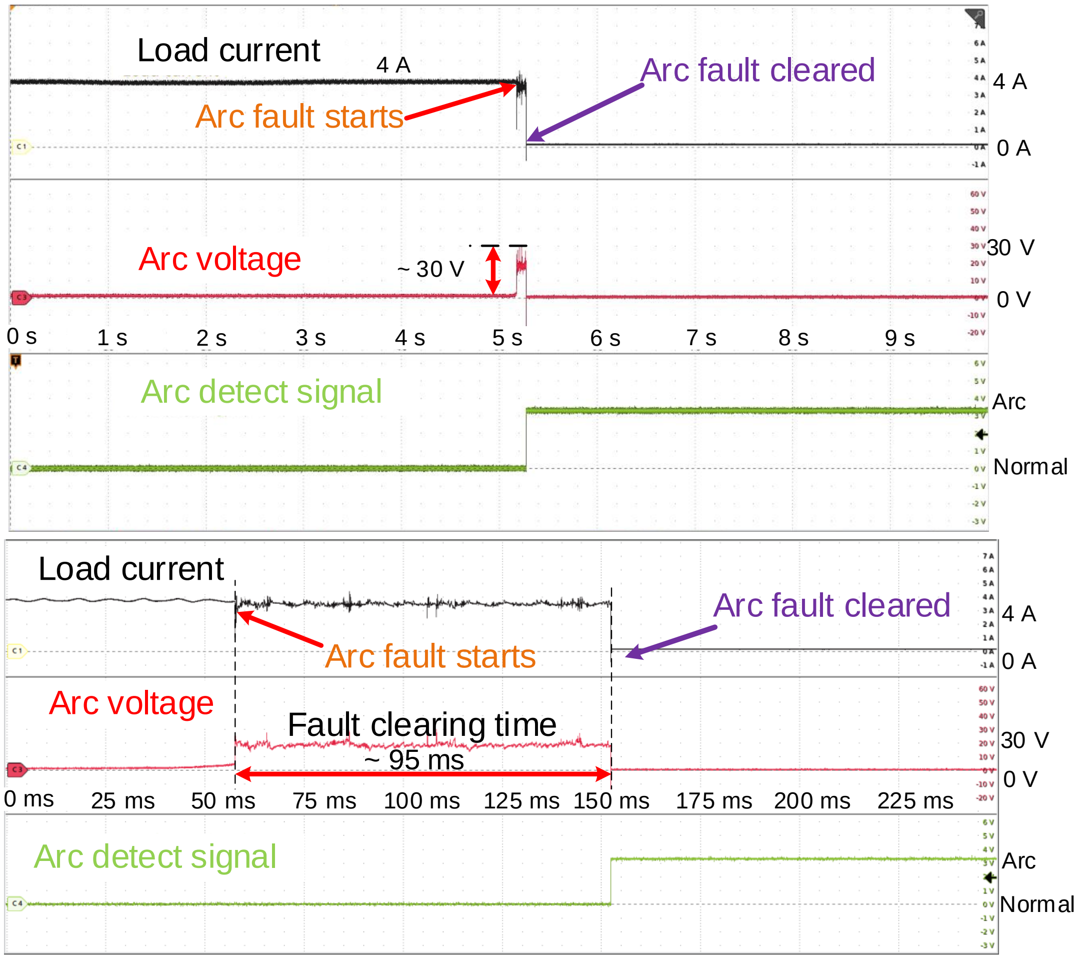


Figure 6.16: AFCI testing for interruption time using 5 cycles of arcs per arc event.

fastest inference time of 19 milliseconds. This performance is particularly noteworthy given the low-cost STM32H743ZI2 microcontroller unit (MCU) and the diverse conditions under which the model was tested. A detailed comparison, presented in Table 6.8, lists various models, their parameter counts, and the platforms they were tested on. The proposed method stands out due to its low parameter size (4.2K) and its deployment on the STM32H743ZI2, combined with the solid-state circuit breaker.

Several key findings from the comparison include models tested on various platforms. The model tested on the Udoo X86 MCU achieved 99.9% accuracy but had a longer runtime of 89 milliseconds. Another model using the TMS320F280049 with a small window size of

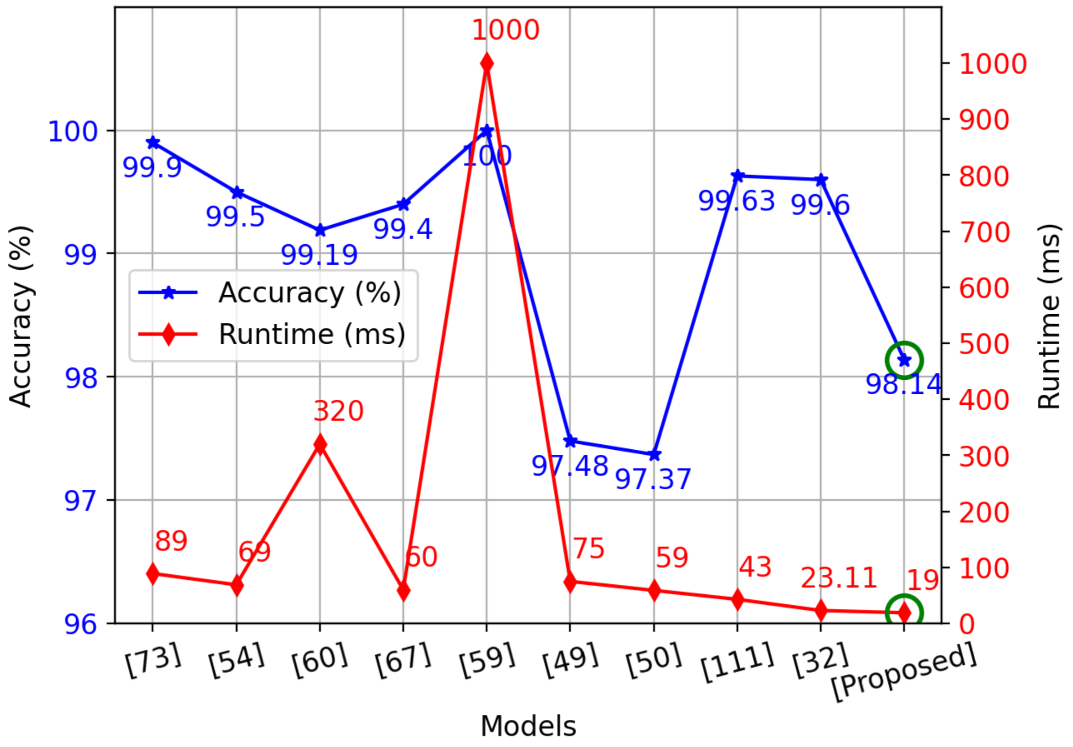


Figure 6.17: Performance comparison: proposed vs. state-of-the-art methods.

2 milliseconds achieved 99.5% accuracy and a runtime of 69 milliseconds. The model tested on the NI PXI3-8821 reached 100% accuracy but with a significantly high runtime of 320 milliseconds. A model with 3.6 million parameters showed 99.19% accuracy and a runtime of 60 milliseconds. Another model with 514.54K parameters achieved 99.4% accuracy and a runtime of 100 milliseconds. A model using an I7-10750 CPU had 354.37K parameters, 97.48% accuracy, and a runtime of 75 milliseconds. The Compact RIO9030 platform, with 103.35K parameters, achieved 97.37% accuracy and a runtime of 59 milliseconds. The i.MX RT1064 platform with 208.9K parameters achieved 99.6% accuracy and a runtime of 43 milliseconds. The Jetson Nano platform, with 332.8K parameters, achieved 98.14% accuracy and a runtime of 23.11 milliseconds.

Our proposed AFCI, tested on the STM32H743ZI2, demonstrated the best balance between high accuracy and low runtime, making it the most efficient model for real-time arc fault detection. This ensures both performance and cost-effectiveness, highlighting its

Table 6.8: Comparison of different state-of-the-art methods and platforms.

Method	**Remarks**	**Platform**
[73]	Params not found	Udoo X86 MCU
[54]	Small window size 2 ms	TMS320F280049
[60]	Not run in MCU	NI PXI3-8821
[67]	Params 3.6 M	Not found
[59]	Params 514.54 K	Not found
[49]	Params 354.37 K	I7-10750 CPU
[50]	Params 103.35 K	Compact RIO9030
[111]	Params 208.9 K	i.MX RT1064
[32]	Params 332.8 K	Jenson Nano
PADNet-Lite (proposed)	Params 4.2 K; SSCB	STM32H743ZI2

suitability for practical applications in photovoltaic systems.

CHAPTER 7: CONCLUSIONS AND FUTURE RESEARCH

7.1 Conclusion

The proposed research developed efficient AC and DC arc fault detection algorithms and provided real-time hardware verifications. Traditional CNN-based algorithms eliminate the drawbacks of manually adjusted threshold problems of conventional frequency domain methods but exhibit significant computational burdens for commercial MCUs. However, a lightweight CNN algorithm with proper optimization can be effective for real-time arc fault protection using low-end MCUs. Additionally, integrating SSCB provides fast interruption of arc faults, thereby enhancing power system safety and reliability. The conclusive remarks of the proposed methods are as follows:

- **Advanced CNN-based Arc Fault Detection for AC Systems:** The development of a CNN-based algorithm for arc fault detection represents a significant advancement in electrical safety technology. Traditional methods rely on manual threshold setting, which can be error-prone and less adaptable to varying environmental conditions. The CNN-based approach automates the detection process, significantly reducing the possibility of human error and enhancing system reliability. By achieving an accuracy rate of 99.47%, this method sets a new standard for fault detection efficacy. Additionally, the research identified 10 kHz as the balanced sampling frequency for AC arc fault detection, balancing system responsiveness and computational efficiency. This finding provides a crucial guideline for system design, ensuring effective arc fault detection while optimizing system resource utilization.

- **Lightweight CNN Architecture for Resource-Constrained Environments:** A key contribution of this research is the development of a lightweight CNN-based algorithm specifically designed for AC arc fault detection on platforms with limited computational resources. Achieving an impressive detection accuracy of 99.31% and an ultra-fast runtime of only 0.20 milliseconds per sample, this algorithm challenges the conventional trade-offs between performance and computational demand. Its minimalistic design conserves system resources and ensures seamless integration into commercially available microcontroller units (MCUs), enhancing their capabilities in critical arc fault detection applications.

- **Cost-Effective AI-Driven AFCI for DC Systems:** Another significant contribution of this research is the proposal and implementation of an AI-driven AFCI for DC systems. Utilizing a combination of an STM32 microcontroller unit (MCU) and a Silicon Carbide (SiC) MOSFET-based Solid State Circuit Breaker (SSCB), this system not only detects arc faults with high accuracy (98.14%) but also interrupts faulty circuits faster than any previously recorded systems (within 19 milliseconds). This rapid response capability is critical in preventing electrical fire hazards due to DC arc faults. The cost-effectiveness of the proposed AFCI system, coupled with its high performance, marks a significant advancement in protective devices for DC power systems, particularly in renewable energy applications like photovoltaic systems.

These contributions collectively represent substantial advancements in the field of electrical safety, particularly in detecting and managing arc faults in both AC and DC systems. They reflect a move towards more intelligent, efficient, and reliable safety mechanisms that can adapt to the demands of modern electrical systems and are poised to make a significant impact on the industry's approach to managing electrical fire hazards due to arc faults.

7.2 Future Research Directions

The research presented in this dissertation lays a robust foundation for arc fault detection in both AC and DC systems. However, there are several avenues for future research that can build upon these findings and further enhance the effectiveness and applicability of the developed technologies.

1. **Incorporating Real PV Systems**: Future work should involve integrating the proposed AFCI system into actual photovoltaic systems to verify its functionality and effectiveness in real-world scenarios. This step is essential to assess the system's performance under varying environmental conditions and operational stresses, providing a more comprehensive validation of the model.

2. **Testing at Higher Voltages and Load Conditions**: Extending the testing to higher voltages and diverse load conditions will validate the robustness and reliability of the model. Evaluating the detection algorithm and the SSCB under extreme conditions ensures the system's durability and effectiveness, confirming its suitability for broader applications.

3. **Exploring Advanced AI Techniques**: Further research could explore the integration of advanced AI and machine learning techniques, such as reinforcement learning or hybrid models, to enhance the detection accuracy and response time of the AFCI system.

4. **Enhancing Data Augmentation Methods**: Improving data augmentation techniques can provide a more diverse training dataset, which is crucial for enhancing the model's generalization capabilities. Simulating various fault scenarios and environmental conditions will create a robust training dataset, improving the system's overall performance.

5. **Developing Adaptive Algorithms**: Future research could focus on developing adaptive algorithms that can adjust their parameters in real-time based on the operating conditions. This will ensure optimal performance of the AFCI system under different scenarios, making it more responsive and reliable.

These future research directions build on the strong foundation established in this dissertation, aiming to further enhance the capabilities and applicability of arc fault detection technologies in photovoltaic and other electrical systems. By pursuing these avenues, the ongoing development and refinement of these systems can be ensured, contributing to greater safety and reliability in electrical infrastructure.

7.3 Published Papers & Provisional Patents

The following list is a summary of my main publications.

1. Y. Wang, L. Hou, K. C. Paul, Y. Ban, C. Chen, and T. Zhao, "ArcNet: Series AC Arc Fault Detection Based on Raw Current and Convolutional Neural Network," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 77-86, 2021.

2. Y. Wang, J. Zhou, K. C. Paul, T. Zhao, and D. Sheng, "Explainability Approach-Based Series Arc Fault Detection Method for Photovoltaic Systems," *IEEE Access*, vol. 12, pp. 45 530-45 542, 2024.

3. K. C. Paul, T. Zhao, C. Chen, Y. Ban, and Y. Wang, "Efficient-ArcNet: Series AC Arc Fault Detection using Lightweight Convolutional Neural Network," in *2021 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 1327-1333, IEEE, 2021.

4. K. C. Paul, L. Schweizer, T. Zhao, C. Chen, and Y. Wang, "Series AC Arc Fault Detection Using Decision Tree-Based Machine Learning Algorithm and Raw Current," in *2022 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 1-8, 2022.

5. K. C. Paul, J. Zhou, S. Chen, C. Chen, and T. Zhao, "Efficient and Lightweight Convolutional Neural Network-based Series DC Arc Fault Protection for PV Systems," in *2024 IEEE Holm Conference on Electrical Contacts*, 2024 (accepted).

6. S. Chen, K. C. Paul, and T. Zhao, "Enhancing Arc Fault Detection Performance through Data Augmentation with Artificial Intelligence Technology: An Approach to Time Series Dataset Enlargement," in *2024 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2024 (accepted).

Provisional patents are enlisted below:

1. T. Zhao, K. Paul, and Y. Wang, "ArcNet-Lite: Lightweight Deep Neural Network for Real-Time Series AC Arc Fault Detection," US Patent Application No. 63/531,659, 2023.
2. T. Zhao, K. Paul, and S. Chen, "Lightweight Neural Network-based Series DC Arc Fault Circuit Interrupter for PV Systems," US Patent Application No. 63/626,673, 2024.

Journals under review are enlisted below:

1. K. C. Paul, C. Chen, Y. Wang, and T. Zhao, "LArcNet: Lightweight Deep Neural Network for Real-Time Series AC Arc Fault Detection," submitted to *IEEE Open Journal of Industry Applications*, April 2024.
2. K. C. Paul, D. Waldmann, C. Chen, and T. Zhao, "Artificial Intelligence for DC Arc Fault Detection: A Comprehensive Review," submitted to *IEEE Access*, July 2024.

REFERENCES

[1] G. D. Gregory and G. W. Scott, "The arc-fault circuit interrupter, an emerging product," in *1998 IEEE Industrial and Commercial Power Systems Technical Conference. Conference Record. Papers Presented at the 1998 Annual Meeting (Cat. No. 98CH36202)*, pp. 48–55, IEEE, 1998.

[2] T. Covington, "House Fire Statistics and Facts in 2020," Apr 2021.

[3] I. E. Commission *et al.*, "General requirements for arc fault detection devices; IEC 62606," *International Electrotechnical Commission: Geneva, Switzerland*, 2017.

[4] UL1699, "Standard for Safety Arc Fault Circuit-Interrupter." Underwriters Laboratories Inc., 2017.

[5] NEC, "National Electrical Code Handbook, 2014 ed.," *National Fire Protection Association*, 2014.

[6] A. Mellit, G. M. Tina, and S. A. Kalogirou, "Fault detection and diagnosis methods for photovoltaic systems: A review," *Renewable and Sustainable Energy Reviews*, vol. 91, pp. 1–17, 2018.

[7] S. McCalmont, "Low cost arc fault detection and protection for PV systems: January 30, 2012-september 30, 2013," tech. rep., National Renewable Energy Lab.(NREL), Golden, CO (United States), 2013.

[8] SEIA, "Solar Industry Research Data," 2023.

[9] SEIA, "U.S. Solar Market Insight," 2023.

[10] V. Linga, "EIA projects that renewable generation will supply 44% of U.S. electricity by 2050." `https://tinyurl.com/jm8eeza4`, Mar 2022. [Accessed 24-May-2024].

[11] Y. Wang, L. Hou, K. C. Paul, Y. Ban, C. Chen, and T. Zhao, "ArcNet: Series AC Arc Fault Detection Based on Raw Current and Convolutional Neural Network," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 77–86, 2021.

[12] Y. Wang, F. Zhang, S. Zhang, and G. Yang, "A novel diagnostic algorithm for AC series arcing based on correlation analysis of high-frequency component of wavelet," *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering*, 2017.

[13] D. O. Anggriawan, A. E. Rheinanda, M. K. Khafidli, E. Prasetyono, and N. A. Windarko, "Series Arc Fault Breaker in Low Voltage Using Microcontroller Based on Fast Fourier Transform," *EMITTER International Journal of Engineering Technology*, vol. 9, no. 2, pp. 239–251, 2021.

[14] P. Qi, S. Jovanovic, J. Lezama, and P. Schweitzer, “Discrete wavelet transform optimal parameters estimation for arc fault detection in low-voltage residential power networks,” *Electric power systems research*, vol. 143, pp. 130–139, 2017.

[15] M. K. Khafidli, E. Prasetyono, D. O. Anggriawan, A. Tjahjono, and M. H. R. A. Syafii, “Implementation AC series arc fault recognition using mikrokontroller based on fast Fourier transform,” in *2018 International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, pp. 31–36, IEEE, 2018.

[16] S. Lu, B. Phung, and D. Zhang, “A comprehensive review on DC arc faults and their diagnosis methods in photovoltaic systems,” *Renewable and Sustainable Energy Reviews*, vol. 89, pp. 88–98, 2018.

[17] K. Xia, Z. He, Y. Yuan, Y. Wang, and P. Xu, “An arc fault detection system for the household photovoltaic inverter according to the DC bus currents,” in *2015 18th International Conference on Electrical Machines and Systems (ICEMS)*, pp. 1687–1690, IEEE, 2015.

[18] K. C. Paul, T. Zhao, C. Chen, Y. Ban, and Y. Wang, “Efficient-ArcNet: Series AC Arc Fault Detection using Lightweight Convolutional Neural Network,” in *2021 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 1327–1333, IEEE, 2021.

[19] Y. Meng, Q. Yang, S. Chen, Q. Wang, and X. Li, “Multi-branch AC arc fault detection based on ICEEMDAN and LightGBM algorithm,” *Electric Power Systems Research*, vol. 220, p. 109286, 2023.

[20] Y. Wang, F. Zhang, X. Zhang, and S. Zhang, “Series AC Arc Fault Detection Method Based on Hybrid Time and Frequency Analysis and Fully Connected Neural Network,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6210–6219, 2018.

[21] W. Li, Y. Liu, Y. Li, and F. Guo, “Series Arc Fault Diagnosis and Line Selection Method Based on Recurrent Neural Network,” *IEEE Access*, vol. 8, pp. 177815–177822, 2020.

[22] J. Jiang, W. Li, Z. Wen, Y. Bie, H. Schwarz, and C. Zhang, “Series Arc Fault Detection Based on Random Forest and Deep Neural Network,” *IEEE Sensors Journal*, 2021.

[23] M. A. Abdulrachman, E. Prasetyono, D. O. Anggriawan, and A. Tjahjono, “Smart detection of AC series arc fault on home voltage line based on fast Fourier transform and artificial neural network,” in *2019 International Electronics Symposium (IES)*, pp. 439–445, IEEE, 2019.

[24] J. E. Siegel, S. Pratt, Y. Sun, and S. E. Sarma, “Real-time Deep Neural Networks for internet-enabled arc-fault detection,” *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 35–42, 2018.

[25] X. Han, D. Li, L. Huang, H. Huang, J. Yang, Y. Zhang, X. Wu, and Q. Lu, "Series Arc Fault Detection Method Based on Category Recognition and Artificial Neural Network," *Electronics*, vol. 9, no. 9, p. 1367, 2020.

[26] N. Qu, J. Chen, J. Zuo, and J. Liu, "PSO-SOM neural network algorithm for series arc fault detection," *Advances in Mathematical Physics*, vol. 2020, 2020.

[27] N. Qu, J. Zuo, J. Chen, and Z. Li, "Series arc fault detection of indoor power distribution system based on LVQ-NN and PSO-SVM," *IEEE Access*, vol. 7, pp. 184020–184028, 2019.

[28] K. C. Paul, L. Schweizer, T. Zhao, C. Chen, and Y. Wang, "Series AC Arc Fault Detection Using Decision Tree-Based Machine Learning Algorithm and Raw Current," in *2022 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 1–8, 2022.

[29] X. Ning, D. Sheng, J. Zhou, Y. Liu, Y. Wang, H. Zhang, and X. Lei, "Arc_EffNet: A Novel Series Arc Fault Detection Method Based on Lightweight Neural Network," *Electronics*, vol. 12, no. 22, p. 4617, 2023.

[30] Y. Wang, C. Bai, X. Qian, W. Liu, C. Zhu, and L. Ge, "A DC Series Arc Fault Detection Method Based on a Lightweight Convolutional Neural Network Used in Photovoltaic System," *Energies*, vol. 15, no. 8, p. 2877, 2022.

[31] D. R. Akbar, E. Prasetyono, and D. O. Anggriawan, "Identification of Series DC Arc Fault Location in MPPT Photovoltaic System Based on Fast Fourier Transform and Artificial Neural Network," in *2022 International Electronics Symposium (IES)*, pp. 26–33, IEEE, 2022.

[32] Y. Sung, G. Yoon, J.-H. Bae, and S. Chae, "TL–LED arc Net: Transfer Learning Method for Low-Energy Series DC Arc-Fault Detection in Photovoltaic Systems," *IEEE Access*, 2022.

[33] A. H. Omran, D. M. Said, S. M. Hussin, N. Ahmad, and H. Samet, "A novel intelligent detection schema of series arc fault in photovoltaic (PV) system based convolutional neural network," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 8, no. 3, pp. 1641–1653, 2020.

[34] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[35] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

[36] I. Freeman, L. Roese-Koerner, and A. Kummert, "Effnet: An Efficient Structure for Convolutional Neural Networks," in *2018 25th ieee international conference on image processing (icip)*, pp. 6–10, IEEE, 2018.

[37] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[38] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[39] G. Artale, A. Cataliotti, V. Cosentino, and G. Privitera, "Experimental characterization of series arc faults in AC and DC electrical circuits," in *2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, pp. 1015–1020, IEEE, 2014.

[40] V. V. Terzija, M. Popov, V. Stanojevic, and Z. Radojevic, "EMTP simulation and spectral domain features of a long arc in free air," in *CIRED 2005-18th International Conference and Exhibition on Electricity Distribution*, pp. 1–4, IET, 2005.

[41] J. Johnson, B. Pahl, C. Luebke, T. Pier, T. Miller, J. Strauch, S. Kuszmaul, and W. Bower, "Photovoltaic DC arc fault detector testing at Sandia National Laboratories," in *2011 37th IEEE Photovoltaic Specialists Conference*, pp. 003614–003619, IEEE, 2011.

[42] J.-C. Gu, D.-S. Lai, J.-M. Wang, J.-J. Huang, and M.-T. Yang, "Design of a DC series arc fault detector for photovoltaic system protection," *IEEE Transactions on Industry Applications*, vol. 55, no. 3, pp. 2464–2471, 2019.

[43] M. Ahmadi, H. Samet, and T. Ghanbari, "A new method for detecting series arc fault in photovoltaic systems based on the blind-source separation," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 6, pp. 5041–5049, 2019.

[44] UL1699B, "Standard for Safety Arc Fault Circuit-Interrupter." Underwriters Laboratories Inc., 2018.

[45] C. Strobl and P. Meckler, "Arc faults in photovoltaic systems," *2010 Proceedings of the 56th IEEE Holm Conference on Electrical Contacts*, pp. 1–7, 2010.

[46] D. A. Dini, P. W. Brazis, and K.-H. Yen, "Development of arc-fault circuit-interrupter requirements for photovoltaic systems," in *2011 37th IEEE Photovoltaic specialists conference*, pp. 001790–001794, IEEE, 2011.

[47] V. Le and X. Yao, "Ensemble machine learning based adaptive arc fault detection for DC distribution systems," in *2019 IEEE Applied Power Electronics Conference and Exposition (APEC)*, pp. 1984–1989, IEEE, 2019.

[48] P. Schweitzer, R. Chu, Y. Jiang, C. Bonnet, Y. Berviller, S. Weber, E. Tisserand, and A. Chabert, "An algorithm for the detection of DC series-arc faults using a Convolution Neural Network," in *2022 IEEE 67th Holm Conference on Electrical Contacts (HLM)*, pp. 1–4, IEEE, 2022.

[49] X. Chen, W. Gao, C. Hong, and Y. Tu, “A novel series arc fault detection method for photovoltaic system based on multi-input neural network,” *International Journal of Electrical Power & Energy Systems*, vol. 140, p. 108018, 2022.

[50] S. Lu, R. Ma, T. Sirojan, B. Phung, and D. Zhang, “Lightweight transfer nets and adversarial data augmentation for photovoltaic series arc fault detection with limited fault data,” *International Journal of Electrical Power & Energy Systems*, vol. 130, p. 107035, 2021.

[51] J. Yan, Q. Li, and S. Duan, “A Simplified Current Feature Extraction and Deployment Method for DC series Arc Fault Detection,” *IEEE Transactions on Industrial Electronics*, 2023.

[52] Z. Yi and A. H. Etemadi, “Fault detection for photovoltaic systems based on multi-resolution signal decomposition and fuzzy inference systems,” *IEEE Transactions on Smart Grid*, vol. 8, no. 3, pp. 1274–1283, 2016.

[53] L. Wang, E. Lodhi, P. Yang, H. Qiu, W. U. Rehman, Z. Lodhi, T. S. Tamir, and M. A. Khan, “Adaptive local mean decomposition and multiscale-fuzzy entropy-based algorithms for the detection of DC series arc faults in PV systems,” *Energies*, vol. 15, no. 10, p. 3608, 2022.

[54] V. Le, X. Yao, C. Miller, and B.-H. Tsao, “Series DC arc fault detection based on ensemble machine learning,” *IEEE Transactions on Power Electronics*, vol. 35, no. 8, pp. 7826–7839, 2020.

[55] J. Yang and Y. Wang, “Identification and detection of DC arc fault in photovoltaic power generation system,” in *2020 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, pp. 440–444, IEEE, 2020.

[56] R. D. Telford, S. Galloway, B. Stephen, and I. Elders, “Diagnosis of series DC arc faults-A machine learning approach,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1598–1609, 2016.

[57] Q. Lu, S. Duan, Q. Li, T. Peng, and J. Yan, “Design of DCArc Fault Online Detectors in Photovoltaic Systems Based on Neural Network,” in *2022 IEEE 5th International Electrical and Energy Conference (CIEEC)*, pp. 2442–2447, IEEE, 2022.

[58] L. Xing, Y. Wen, S. Xiao, D. Zhang, and J. Zhang, “A deep learning approach for series DC arc fault diagnosing and real-time circuit behavior predicting,” *IEEE Transactions on Electromagnetic Compatibility*, vol. 64, no. 2, pp. 569–579, 2021.

[59] T. Li, Z. Jiao, L. Wang, and Y. Mu, “A method of DC arc detection in all-electric aircraft,” *Energies*, vol. 13, no. 16, p. 4190, 2020.

[60] Z. Yin, L. Wang, B. Zhang, L. Meng, and Y. Zhang, “An Integrated DC Series Arc Fault Detection Method for Different Operating Conditions,” *IEEE Transactions on Industrial Electronics*, vol. 68, no. 12, pp. 12720–12729, 2020.

[61] W. Miao, Q. Xu, K. Lam, P. W. Pong, and H. V. Poor, "DC arc-fault detection based on empirical mode decomposition of arc signatures and support vector machine," *IEEE Sensors Journal*, vol. 21, no. 5, pp. 7024–7033, 2020.

[62] H. Lala and S. Karmakar, "Application of Empirical Mode Decomposition and Support Vector Machine for the Classification of Arc Fault in Distribution Line," in *Advances in Electronics, Communication and Computing*, pp. 385–393, Springer, 2021.

[63] T. Shang, W. Wang, J. Peng, B. Xu, H. Gao, and G. Zhai, "Series arc fault identification based on complete ensemble empirical mode decomposition with adaptive noise and convolutional neural network," *International Journal of Metrology and Quality Engineering*, vol. 13, p. 11, 2022.

[64] X. Cai and R.-J. Wai, "Intelligent DC Arc-Fault Detection of Solar PV Power Generation System via Optimized VMD-Based Signal Processing and PSO–SVM Classifier," *IEEE Journal of Photovoltaics*, 2022.

[65] M. Rabla, E. Tisserand, P. Schweitzer, and J. Lezama, "Arc fault analysis and localisation by cross-correlation in 270 V DC," in *2013 IEEE 59th Holm Conference on Electrical Contacts (Holm 2013)*, pp. 1–6, IEEE, 2013.

[66] Q. Xiong, S. Ji, L. Zhu, L. Zhong, and Y. Liu, "A novel DC arc fault detection method based on electromagnetic radiation signal," *IEEE Transactions on Plasma Science*, vol. 45, no. 3, pp. 472–478, 2017.

[67] S. Lu, T. Sirojan, B. Phung, D. Zhang, and E. Ambikairajah, "DA-DCGAN: An effective methodology for DC series arc fault diagnosis in photovoltaic systems," *IEEE Access*, vol. 7, pp. 45831–45840, 2019.

[68] L. Zhao, Y. Zhou, K.-L. Chen, S.-H. Rau, and W.-J. Lee, "High-speed arcing fault detection: Using the light spectrum," *IEEE Industry Applications Magazine*, vol. 26, no. 3, pp. 29–36, 2020.

[69] L. Zhao, Y. Zhou, K.-L. Chen, S.-H. Rau, and W.-J. Lee, "Using spectrum of the light for high speed arcing fault detection," in *2018 IEEE IAS Electrical Safety Workshop (ESW)*, pp. 1–8, IEEE, 2018.

[70] S. Lu, A. Sahoo, R. Ma, and B. Phung, "DC series arc fault detection using machine learning in photovoltaic systems: Recent developments and challenges," in *2020 8th International Conference on Condition Monitoring and Diagnosis (CMD)*, pp. 416–421, IEEE, 2020.

[71] Z. Wang, S. Tian, H. Gao, C. Han, and F. Guo, "An On-Line Detection Method and Device of Series Arc Fault Based on Lightweight CNN," *IEEE Transactions on Industrial Informatics*, 2023.

[72] Q. Yu, G. Huang, and Y. Yang, “Low voltage AC series arc fault detection method based on parallel deep convolutional neural network,” in *IOP Conference Series: Materials Science and Engineering*, vol. 490, p. 072020, IOP Publishing, 2019.

[73] V. Le, X. Yao, C. Miller, and T.-B. Hung, “Arc fault detection in DC distribution using semi-supervised ensemble machine learning,” in *2019 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 2939–2945, IEEE, 2019.

[74] M. Xie, X. Zhang, Y. Dong, and W. Li, “Arc fault detection for DC solid state power controllers,” in *2014 IEEE Conference and Expo Transportation Electrification Asia-Pacific (ITEC Asia-Pacific)*, pp. 1–6, IEEE, 2014.

[75] S. Y. Guo, J. L. Jones III, and A. S. Dooley, “DC arc detection and prevention circuit and method,” Jan. 27 2004. US Patent 6,683,766.

[76] B. Grichting, J. Goette, and M. Jacomet, “Cascaded fuzzy logic based arc fault detection in photovoltaic applications,” in *2015 International Conference on Clean Electrical Power (ICCEP)*, pp. 178–183, IEEE, 2015.

[77] H.-L. Dang, S. Kwak, and S. Choi, “Different domains based machine and deep learning diagnosis for DC series arc failure,” *IEEE Access*, vol. 9, pp. 166249–166261, 2021.

[78] J.-C. Kim and S.-S. Kwak, “Frequency-domain characteristics of series DC arcs in photovoltaic systems with voltage-source inverters,” *Applied Sciences*, vol. 10, no. 22, p. 8042, 2020.

[79] J. Johnson and K. Armijo, “Parametric study of PV arc-fault generation methods and analysis of conducted DC spectrum,” in *2014 IEEE 40th Photovoltaic Specialist Conference (PVSC)*, pp. 3543–3548, IEEE, 2014.

[80] N. L. Georgijevic, M. V. Jankovic, S. Srdic, and Z. Radakovic, “The detection of series arc fault in photovoltaic systems based on the arc current entropy,” *IEEE Transactions on Power Electronics*, vol. 31, no. 8, pp. 5917–5930, 2015.

[81] C.-G. Cho, J.-B. Ahn, J.-H. Lee, H.-J. Ryoo, K.-D. Lee, and Y.-J. Kim, “DC Series Arc Fault Characteristic Comparision Of A Fast Fourier Transform Result,” in *2020 8th International Conference on Condition Monitoring and Diagnosis (CMD)*, pp. 218–221, IEEE, 2020.

[82] M. H. R. A. Syafi’i, E. Prasetyono, M. K. Khafidli, D. O. Anggriawan, and A. Tjahjono, “Real time series DC arc fault detection based on fast Fourier transform,” in *2018 International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, pp. 25–30, IEEE, 2018.

[83] R. Balamurugan, F. Al-Janahi, O. Bouhali, S. Shukri, K. Abdulmawjood, and R. S. Balog, “Fourier Transform and Short-Time Fourier Transform decomposition

for photovoltaic arc fault detection," in *2020 47th IEEE Photovoltaic Specialists Conference (PVSC)*, pp. 2737–2742, IEEE, 2020.

[84] G.-S. Seo, H. Bae, B.-H. Cho, and K.-C. Lee, "Arc protection scheme for DC distribution systems with photovoltaic generation," in *2012 International Conference on Renewable Energy Research and Applications (ICRERA)*, pp. 1–5, IEEE, 2012.

[85] C. Aarstad, T. Taufik, A. Kean, and M. Muscarella, "Development of arc fault interrupter laboratory testing for low voltage DC electricity," in *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pp. 583–588, IEEE, 2016.

[86] J. A. Momoh and R. Button, "Design and analysis of aerospace DC arcing faults using fast Fourier transformation and artificial neural network," in *2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No. 03CH37491)*, vol. 2, pp. 788–793, IEEE, 2003.

[87] X. Liu, D. Huang, T. Jing, and Y. Zhang, "Detection of AC Arc Faults of Aviation Cables Based on HIW Three-Dimensional Features and CNN-LSTM Neural Network," *IEEE Access*, vol. 10, pp. 106958–106971, 2022.

[88] A. Khamkar and D. D. Patil, "Arc fault and flash signal analysis of DC distribution system sing artificial intelligence," in *2020 International Conference on Renewable Energy Integration into Smart Grids: A Multidisciplinary Approach to Technology Modelling and Simulation (ICREISG)*, pp. 10–15, IEEE, 2020.

[89] S. Lu, *Intelligent DC Series Arc Fault Detection using Deep Learning in Photovoltaic Systems.* PhD thesis, UNSW Faculty of Engineering, 2021.

[90] H. Zhu, Z. Wang, and R. S. Balog, "Real time arc fault detection in PV systems using wavelet decomposition," in *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC)*, pp. 1761–1766, IEEE, 2016.

[91] C. He, L. Mu, and Y. Wang, "The detection of parallel arc fault in photovoltaic systems based on a mixed criterion," *IEEE Journal of Photovoltaics*, vol. 7, no. 6, pp. 1717–1724, 2017.

[92] Z. Wang and R. S. Balog, "Arc fault and flash signal analysis in DC distribution systems using wavelet transformation," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1955–1963, 2015.

[93] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[94] Z. Wang and R. S. Balog, "Arc fault and flash detection in DC photovoltaic arrays using wavelets," in *2013 IEEE 39th Photovoltaic specialists conference (PVSC)*, pp. 1619–1624, IEEE, 2013.

[95] Z. Wang, S. McConnell, R. S. Balog, and J. Johnson, "Arc fault signal detection-fourier transformation vs. wavelet decomposition techniques using synthesized data," in *2014 IEEE 40th Photovoltaic Specialist Conference (PVSC)*, pp. 3239–3244, IEEE, 2014.

[96] J. Yeager, H.-C. Hsieh, S. Baek, and J.-S. Lai, "Series DC Arc Fault Detection Using a Wavelet-Based Filter Bank with Statistical Analysis," in *2022 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 1–6, IEEE, 2022.

[97] H.-P. Park, M. Kim, and S. Chae, "Smart DC Optimizer for DC Series Arc Fault Detection and Extinguishing," *IEEE Transactions on Power Electronics*, vol. 37, no. 9, pp. 10117–10121, 2022.

[98] R. Janarthanan, R. U. Maheshwari, P. K. Shukla, P. K. Shukla, S. Mirjalili, and M. Kumar, "Intelligent Detection of the PV Faults Based on Artificial Neural Network and Type 2 Fuzzy Systems," *Energies*, vol. 14, no. 20, p. 6584, 2021.

[99] H.-L. Dang, J. Kim, S. Kwak, and S. Choi, "Series DC Arc Fault Detection Using Machine Learning Algorithms," *IEEE Access*, vol. 9, pp. 133346–133364, 2021.

[100] Z. Wang and R. S. Balog, "Arc fault and flash detection in photovoltaic systems using wavelet transform and support vector machines," in *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC)*, pp. 3275–3280, IEEE, 2016.

[101] M. M. Badr, A. S. Abdel-Khalik, M. S. Hamad, R. A. Hamdy, E. Hamdan, S. Ahmed, and N. A. Elmalhy, "Intelligent fault identification strategy of photovoltaic array based on ensemble self-training learning," *Solar Energy*, vol. 249, pp. 122–138, 2023.

[102] W. Gao and R.-J. Wai, "Series Arc Fault Detection of Grid-Connected PV System via SVD Denoising and IEWT-TWSVM," *IEEE Journal of Photovoltaics*, vol. 11, no. 6, pp. 1493–1510, 2021.

[103] J. Beyerer, M. Richter, and M. Nagel, *Pattern recognition: introduction, features, classifiers and principles.* Walter de Gruyter GmbH & Co KG, 2017.

[104] Y. Zhang, H. C. Chen, Z. Li, C. Jia, Y. Du, K. Zhang, and H. Wei, "Lightweight AC Arc Fault Detection Method By Integration of Event-Based Load Classification," *IEEE Transactions on Industrial Electronics*, 2023.

[105] S. Ma and L. Guan, "Arc-fault recognition based on BP neural network," in *2011 Third International Conference on Measuring Technology and Mechatronics Automation*, vol. 1, pp. 584–586, IEEE, 2011.

[106] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT press, 2012.

[107] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[108] S. Zhang, N. Qu, T. Zheng, and C. Hu, "Series arc fault detection based on wavelet compression reconstruction data enhancement and deep residual network," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–9, 2022.

[109] Z. Qi, X. Qi, W. Gao, and W. Ying, "Arc-fault detection using one-dimension convolution neural network," in *2022 5th World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM)*, pp. 488–495, IEEE, 2022.

[110] R. Jiang, G. Bao, Q. Hong, and C. D. Booth, "Machine learning approach to detect arc faults based on regular coupling features," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 2761–2771, 2022.

[111] Y. Wang, J. Zhou, K. C. Paul, T. Zhao, and D. Sheng, "Explainability Approach-Based Series Arc Fault Detection Method for Photovoltaic Systems," *IEEE Access*, 2024.

[112] A. Chabert, M. C. Bakkay, P. Schweitzer, S. Weber, and J. Andrea, "Ac series arc-fault detection with a transformer neural network," 2022.

[113] K. C. Paul, L. Schweizer, T. Zhao, C. Chen, and Y. Wang, "Series AC Arc Fault Detection Using Decision Tree-Based Machine Learning Algorithm and Raw Current," in *2022 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 1–8, IEEE, 2022.

[114] Y. Zhao, L. Yang, B. Lehman, J.-F. de Palma, J. Mosesian, and R. Lyons, "Decision tree-based fault detection and classification in solar photovoltaic arrays," in *2012 Twenty-Seventh Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*, pp. 93–99, IEEE, 2012.

[115] S. Chen, H. Wu, Y. Meng, Y. Wang, X. Li, and C. Zhang, "Reliable Detection Method of Variable Series Arc Fault in Building Integrated Photovoltaic Systems Based on Nonstationary Time Series Analysis," *IEEE Sensors Journal*, vol. 23, no. 8, pp. 8654–8664, 2023.

[116] IEC62606, "General requirements for arc fault detection devices," *IEC62606*, 2013.

[117] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[118] Y. Liu, G. Yang, and H. Wang, "Series Arc Fault Detection under Vibration Condition Based on NMMB," *Sensors*, vol. 24, no. 3, p. 959, 2024.

[119] Wikipedia, "Knowledge distillation," Jan 2023.

[120] Google, "TensorFlow Lite: ML for Mobile and Edge Devices."

[121] Y. Wang, F. Zhang, and S. Zhang, "A new methodology for identifying arc fault by sparse representation and neural network," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 11, pp. 2526–2537, 2018.

[122] J. Jiang, W. Li, Z. Wen, Y. Bie, H. Schwarz, and C. Zhang, "Series Arc Fault Detection Based on Random Forest and Deep Neural Network," *IEEE Sensors Journal*, vol. 21, no. 15, pp. 17171–17179, 2021.

[123] A. Tang, Z. Wang, S. Tian, H. Gao, Y. Gao, and F. Guo, "Series Arc Fault Identification Method Based on Lightweight Convolutional Neural Network," *IEEE Access*, 2024.

[124] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[125] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[126] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and$<$ 0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[127] A. Chakravarthy, "Visualizing intermediate activations of a CNN trained on the mnist dataset," Apr 2021.

[128] F. Aziz, A. U. Haq, S. Ahmad, Y. Mahmoud, M. Jalal, and U. Ali, "A novel convolutional neural network-based approach for fault classification in photovoltaic arrays," *IEEE Access*, vol. 8, pp. 41889–41904, 2020.